Max Salminen

A METADATA MODEL FOR HYBRID DATA PRODUCTS ON A MULTILATERAL DATA MARKETPLACE

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY
2018

ABSTRACT

Salminen, Max
A metadata model for hybrid data products on a multilateral data marketplace
Jyväskylä: University of Jyväskylä, 2018, 81p.
Information systems science, Master's Thesis

Multilateral data marketplaces provide a platform where organizations and individuals can buy, trade, sell, and combine data into hybrid data products. On such marketplaces, multiple different vendors can offer equivalent products in terms of functionality, but different in attributes such as pricing, quality and licensing; attributes that could be contained in metadata. Furthermore, hybrid data products introduce an additional challenge: How can metadata regarding the origin and attributes of data be contained?

In this thesis, we propose a metadata model based on W3C PROV that is able to contain both provenance and relevant metadata related to data products. In addition, the functionality of the metadata model is demonstrated and evaluated through a prototype implementation.

Keywords: Data marketplaces, DaaS, Metadata, W3C PROV

TIIVISTELMÄ

Salminen, Max
Metadatamalli hybrididatatuotteille multilateraalisella datamarkkinapaikalla
Jyväskylä: Jyväskylän yliopisto, 2018, 81s.
Tietojärjestelmätiede, pro gradu -tutkielma

Monitahoiset datamarkkinapaikat tarjoavat alustan, jolla organisaatiot ja yksityishenkilöt voivat ostaa, vaihtaa, myydä ja yhdistää datatuotetteita hybrididatatuotteiksi. Datatuotteilla voi kuitenkin olla useita tarjoajia, jotka saattavat erota toisistaan laadullisten ominaisuuksien tai lisenssiehtojen suhteen. Hybrididatatuotteen muodostamisessa toteutuneen tapahtumaketjun, provenienssin, seuraaminen voisi mahdollistaa hybrididatatuotteen muodostaneiden datatuotteiden ominaisuuksien todentamisen.

Tässä tutkielmassa muodostetaan W3C PROV-määritelmään perustuva metadatamalli, joka mahdollistaa sekä provenienssin, että hybrididatatuotteisiin liittyvän yleisen metadatan seuraamisen. Tutkielmassa kehitettyä metadata-mallia hyödynnetään prototyypissä, jota käytetään myös mittaaman metadata-mallin vaikutusta suorituskykyyn.

Avainsanat: Datamarkkinapaikat, Data palvelumuotona, Metadata, W3C PROV

FIGURES

TABLES

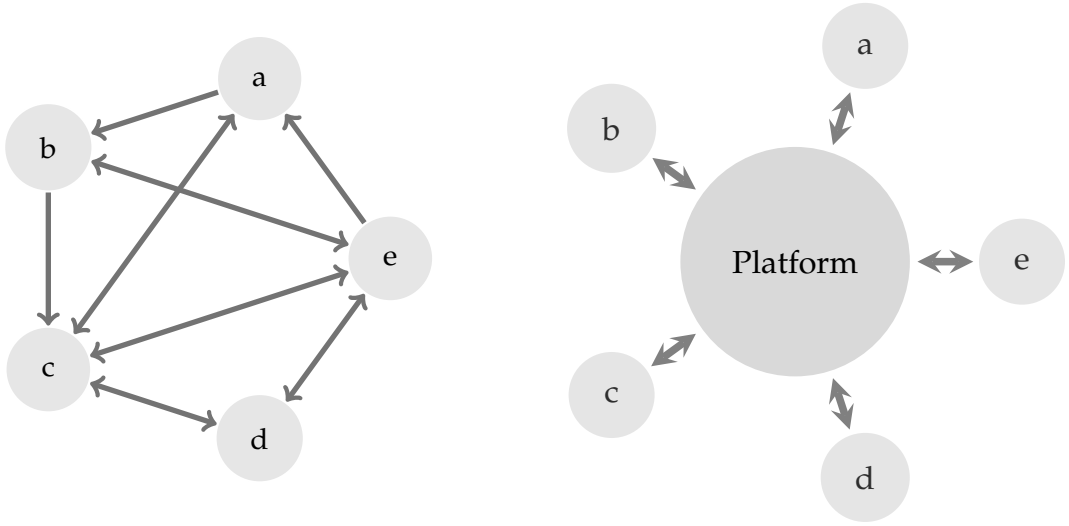
CONTENTS

1 INTRODUCTION

Data from different sources can have inter-dependencies that can be used to discover correlations and other surprising insights. Air pollution metrics require combining data from traffic, weather conditions, and industrial emissions. Similarly a restaurant recommendation algorithm would also need data from multiple sources: restaurant locations, customer reviews and user preferences (Du, Huang, Chen, Xie, Liang, Lv, and Ma 2016). However, an organization might not always possess the required data assets or know-how to implement such algorithms themselves. Data marketplaces address this issue by providing a platform where organizations and individuals can trade data. Trading data can create synergetic benefits for all participants that operate on the platform while also enabling new innovative business models (Schomm, Stahl, and Vossen 2013; Muschalle, Stahl, Löser, and Vossen 2012). Sharing data assets could also highly enhance analytical capabilities of organizations (Arafati, Dagher, Fung, and Hung 2014). The more varied and vast the data assets, the better chance there is to find interesting correlations that could lead to creation of new value. To mention a few examples, industrial IoT data from infrastructure companies could be used by a maintenance company to better understand when and where to dispatch workers using predictive analytics. Car companies could share diagnostic data from vehicles to accelerate development on self-driving cars. Finally, somewhat controversially, advertising firms can combine data from multiple sources to create more accurate advertising profiles on consumers.

There are multiple approaches to trading data between organizations. At simplest, organizations could expose ports on their databases or simply transmit the data for a single, non-generalizable purpose. Although such approach of *ad hoc* data trading is viable for the most basic use cases, the approach becomes unsustainable at scale. Additionally, such approach provides no solid method of tracking the origin and provenance of data and other metadata related to it. Data marketplaces provide a mechanism for trading data between organizations allowing data to be be repurposed for new purposes with minimal use of management and development resources. The difference between the two approaches is visualized in Figure 1. When executed correctly, a data trading platform can lower costs of data management (Arafati et al. 2014). By utilizing capabilities provided by cloud computing, the platform can scale freely to save resources when there is little load on the system, and scale up when there is higher demand. Similarly, cloud services provide high-capacity storage as a service and application service provisioning that further simplify the management of IT infrastructure (Stahl, Schomm, Vossen, and Vomfell 2016). Furthermore, sharing a data marketplace platform with other organizations reduces the need for other organizations to redundantly create similar infrastructures.

Managing a data marketplace platform has multiple and diverse challenges (Koutroumpis, Leiponen, and Thomas 2017). In this thesis, the focus is on providing a metadata model for hybrid data products. A metadata model provides



(a) Ad hoc data sharing (b) Data sharing on a platform

Figure 1: Ad hoc data trading in comparison to trading facilitated by a centralized platform. Nodes (e.g. a, b, c) represent organizations, and arrows represent interactions

a data structure that contains information about the data itself. When multiple data products are combined on a marketplace, a **hybrid data product** is created. To track the provenance of data, metadata related to the original data sources must somehow be included in the derivative data set.

The issues with tracking provenance of hybrid data products is highlighted by Koutroumpis et al. (2017). In a situation where there are multiple data vendors with multiple different licences, all derivative hybrid data products must comply with the terms of every data product from which the hybrid data product consists of. Assume a scenario with following data providers: **Data provider A** offers a free and open data product, **data provider B** provides a proprietary data product with contractual term that requires royalties from derivate data products, **data provider C** provides hybrid data product that combines data from **A** and **B**, and lastly **data provider D** provides a hybrid data product that uses data from **C** in addition to **D**'s own proprietary data. Both **C** and **D** have to be sure that their hybrid data products are compliant with the contractual terms of both **A** and **B** (In addition to **C** in the case of **D**). Lastly, all data product sales from **C** and **D** require part of the cost being transferred to **B** in the form of royalties.

There exists many concepts in literature that relate to the data marketplaces: A **Data marketplace** is a platform where data can be sold, bought or traded (Muschalle et al. 2012). A **Data as a Service (DaaS)** is a Cloud Computing service that provides data. The data is most typically accessed via internet through a set of endpoints that allow various types of operations on the data, such as querying and filtering. A set of these endpoints form an Application Programming Interface (API) that can be used by clients to build new types of services and applications, including higher level platforms for trading data, such as data marketplaces (Vu, Pham, Truong, Dustdar, and Asal 2012; Muschalle et al. 2012). **Stream data** is data that arrives gradually, typically in the form of events, and is potentially infinite

(Kleppmann 2017). For instance, smart meters send the electricity usage readings of customers daily or even hourly, forming a data stream for each customer. An **Event** is a data record that contains the details of something that happened at some point of time (Kleppmann 2017). An event could be a single reading from a smart meter, an overheating warning from a power system, or even a notification that lights have been flicked on in. **Stream processing** is an approach where data is processed as it arrives with minimal latency, as opposed to gathering the data in arbitrarily sized batches and processing them in bulk (Kleppmann 2017). Stream processing enables reacting to data faster in comparison to batch methods, enabling use cases that require low reaction times, such as fraud detection (Stonebraker, Çetintemel, and Zdonik 2005). **Metadata**, at it's most general form, is data about data. As the definition is very broad, Deelman, Berriman, Chervenak, Corcho, Groth, and Moreau (2010) defines metadata to be "structured data about an object that supports functions associated with the designated object". **Provenance** is a record of events that can be used to determine the history of an object of interest (Moreau, Clifford, Freire, Futrelle, Gil, Groth, Kwasnikowska, Miles, Missier, Myers, Plale, Simmhan, Stephan, and den Bussche 2011).

1.1 Research problem, research questions, and limitations

The research problem was inspired by issues that were encountered in a certain local company. Collaborating with other organizations and finding ways to share and reuse data had an incentive problem: How to motivate other organizations to share their data? This problem could largely be addressed with a platform that provides a way for organizations to share data in exchange for monetary compensation, or in other words, a data marketplace.

One specific use case of such data marketplace is the enrichment of information with data from different sources. If organization A has a dataset that could be combined with a dataset from organization B, how would a data marketplace ensure that both organizations are fairly compensated when the data is bought? Additionally, how would it work with industrial IoT devices that provide data in the form of real-time streams, generating new data every second?

In order to fairly compensate all parties on a data marketplace, the origin of the data and the conditions of use must be known. Solving provenance is an open research problem in the field of data marketplaces (Koutroumpis et al. 2017). The main research question of the thesis is: *"How to track provenience of hybrid data products on a multilateral data marketplace?"*. The question lead into investigating the concept of **data marketplaces** that are concerned with the business models and organizational aspects of such platforms, the concept of **Data as a Service** which provides perspective to technical and architectural concepts of data marketplaces, and finally **provenience** that introduces the challenges and approaches to tracking origin of data. Existing literature is used as a base for a metadata model that can be used to track provenience of hybrid data products and other metadata relevant to data marketplaces.

1.2 Research method

The thesis follows Design Science Research Model by Peffers, Tuunanen, Rothenberger, and Chatterjee (2007). The resulting artifact from the thesis is an instantiation of the metadata model, which is evaluated using a prototype that demonstrates the use of artifact in an example scenario. Using the definition by Peffers, Rothenberger, Tuunanen, and Vaezi (2012), a prototype is an implementation of the artifact that is used to demonstrate its utility using illustrative scenarios. Scenarios are used to apply the artifact in synthetic or real world situations to demonstrate its applicability.

DSRM is built around six steps where each of the steps have an output that is used as an input for the next step. The flow of the research process is demonstrated in Figure 2, while the specific DSRM steps are listed and described in order in the following list (Peffers et al. 2007):

1. **Problem identification and motivation:** In the first DSRM step, the research problem is defined and justified. Clearly defining the problem and the concepts related to it helps digesting the complexity of the problem domain. Justifying why the problem should be solved provides insight to the line of reasoning of the researcher and provides motivation by communicating the value of the solution. This is the focus of the chapter 1, as we introduce the problem of data marketplaces and motivate how provenance might be able to solve it.
2. **Define the objectives for a solution:** The problem definition is used to *infer* objectives that can be used to measure the solution. These objectives can be either qualitative or quantitative. Qualitative objectives are similar to functional requirements found in engineering disciplines by describing how the solution would address the problem through its functionality. Quantitative objectives, on the other hand, reflect non-functional requirements that measure specific metrics, such as speed or performance, and can be directly benchmarked against other solutions. After describing the background theory in chapter 2, chapter 3, and chapter 4, we introduce a concrete list of objectives in the form of requirements at the beginning of chapter 5.
3. **Design and development:** Using the *theory* created in the previous step, the artifact is created and the reproducible steps to recreate it are documented in a manner that fits the type of the artifact. There are many possible artifact types that can be created in the process. To mention a few, an artifact can be a concrete instantiation, a model, a construct, or a method for doing something. A software-based artifact, for instance, could include activities of determining artifact's functionality, forming an architecture, and lastly creating an instantiation based on the architecture. In this thesis, we form our own data model based on PROV and design the components that support its function in the context of data marketplaces after defining the requirements in chapter 5. The data model is described in section 5.2.
4. **Demonstration:** The artifact (*How to knowledge*) and its capability of solving a problem is demonstrated with one or more applications that are applicable to the nature of the artifact. Some of these include case studies, proofs,

simulations, and experiments. The data model created in this thesis is tested using a prototype application, with the process being detailed in section 5.3.

5. **Evaluation:** Lastly, the performance and applicability of the artifact in solving the problem is evaluated using *metrics, analysis and knowledge* gained from the demonstration. As with the earlier steps, the method of evaluation that best fits the nature of artifact should be chosen. The results and insight obtained during design, development and demonstration of the metadata model are discussed in section 5.4.

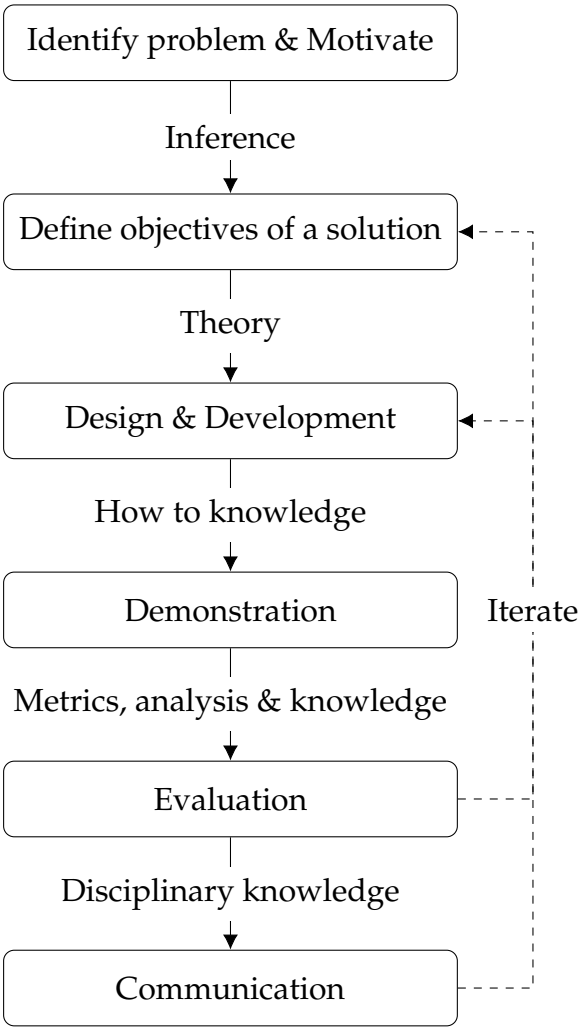


Figure 2: DSRM Process Model by Peffers, Tuunanen, Rothenberger, and Chatterjee (2007).

Although the model appears linear, it provides flexibility by allowing the research to enter the process from later steps, and giving an opportunity to iterate to earlier steps when necessary. By allowing multiple entry points, DSRM can also be applied to research where objectives have already been defined in some other context (Objective-centered solution), or situations where an artifact already exists, but is not used in specific problem context (Design and Development centered approach), or lastly, when the research simply observes a practical solution and

how it works (Client/context initiated solution). In this thesis, however, we follow DSRM from the very first step.

Iteration in the DSRM process can occur at evaluation and communication steps if the nature of the research allows it. A researcher might, for instance, return backwards in order to improve or build on the solution, or find an entirely different approach that solves the problem. The new solution could be benchmarked against the earlier solution to discover whether an improvement was made. Additionally, some researchers might choose to pursue an iterative search process in the design and development phase of the artifact. (Peffers et al. 2007)

2 DATA MARKETPLACES

Finding the appropriate data sources with the right types of data can sometimes be challenging. Either the data is not available in the open, access to it is hidden behind obscure interfaces, or it's simply not in a usable form. In the early years of the web, there were even professionals whose sole task was to search the Web for the desired information and return the results to the client (Schomm et al. 2013). As the technology evolved, so did the methods of information exchange. Websites known as "mashups" that combine information sources from multiple sites became popular, enabling users to combine prices for products from different stores, or trip prices between different airlines (Zhu and Madnick 2009). Data marketplaces take this a step further by making the data itself into a product and by providing a way to trade this information on a single platform. These platforms typically operate in the internet and can be structured in multiple different ways. (Schomm et al. 2013)

In this chapter, we will investigate the concept of Data marketplaces from organizational perspective. The literature used in this chapter was mostly discovered with search engines (`kirjasto.jyu.fi`, `scholar.google.com`) using relevant keywords (e.g. "Data marketplace(s)"). Additional sources were found by following and investigating papers that had cited the literature discovered with search engines.

2.1 Introduction to data marketplaces

To understand data marketplaces, it should first be understood what is meant with the basic economical terms behind the concept of data marketplaces. *Markets* are places where the interactions of buyers and sellers set prices and quantities of goods and services. *Marketplaces* are concrete locations that facilitate markets: Explicit places at an explicit time, where market participants can prepare and execute transactions. By facilitating the interactions between market participants, marketplaces provide the fundamental infrastructure for trading. (Stahl et al. 2016)

In general, the market serves three key functions: Firstly, the market serves as an *institution*: A framework of rules that govern behavior for the participants at the market. Institution assigns roles, expectations, and protocols for trading on the medium. Second, markets provide a *pricing mechanism* for buyers and sellers, the condition of exchange, which operates as an equalizer of supply and demand. If the demand rises above supply, prices rise to compensate. Accordingly, a large surplus with little demand would lead to prices decreasing. Finally, the market defines the process of *transactions*.

According to Richter and Nohr (2002) (as cited by Schomm et al. (2013)), transactions can be broken down to four distinct phases (visualized in Figure 3):

Information phase Market participant seeks information on the goods and create an intention of exchange with bids and offers.

Negotiation phase Negotiate on the goods between buyer and seller. Set contract terms and price for the good, which forms a contract

Transaction phase Fulfillment of contract, where the good is exchanged from seller to buyer

After-sales phase After the contract has been fulfilled, the buyers satisfaction and commitment can be enhanced from the side of seller with customer support.

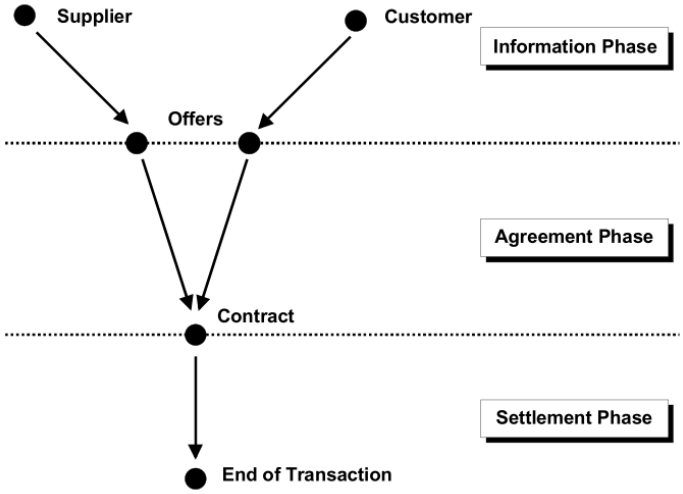


Figure 3: Phases of a market transaction according to Schmid and Lindemann (1998)

With marketplaces also appearing in the Web, Schmid and Lindemann (1998) (as cited by Stahl et al. (2016)) defined the concept of electronic marketplaces to be the an electronic medium that is based on the new digital communication and transaction infrastructure. In comparison to traditional markets, electronic markets have major advantages in terms of high accessibility, low entry barriers, ubiquity, and lower transaction costs. However, ubiquity can also complicate the effort needed to maintain an electronic market as rules, legislation and language might vary between different areas of operation. (Stahl et al. 2016)

For a market to be considered at least partially electronic, at least one phase needs to be done electronically. Some researchers also consider an electronically performed information phase to be the absolute minimum for an electronic marketplace as it enables participation to the market without the conventional limits of location and time. (Schomm et al. 2013)

Data marketplaces extend on the concept of electronic marketplaces by providing a marketplace where the commodity is data. At the most general level, the three main categories of data marketplace users are **data consumers** (buyers) who use the data to their needs, **data providers** (sellers) who give an access to data, and **data marketplace owners** who own the infrastructure on which the trading occurs (Muschalle et al. 2012). Data marketplaces provide a method of accessing variety of information through a single platform that makes it easier for participants trade data. To access the data, data marketplaces must have infrastructure to enable the browsing, uploading, and downloading the data while also facilitating transactions (i.e. buying and selling) between the participants of the platform. The

origin of the data must also be tracked: It must be clear from where the data is public or property of a member who is exchanging data on the platform. (Stahl et al. 2016)

Data is an abstract digitized good which makes estimating it's value difficult. One way to approach the problem of how to valuate data is through commoditization, a process of product standardization. The level of commoditization dictates how much a product diverges from other similar products. In the case of data, a highly commoditized data would be in a standardized format and it could be used with the same tools and applications as similar data from an alternative data provider. Machine-readability of data is one of the main indicators of commoditization. Uncommoditized data, however, would be highly different from other data sets without a specific standardized way to access it. The more commoditized the data product is, the more perfect the market will be in terms of competition and freedom. There are cases where uncommditized data might be a more appropriate choice as perfect markets might not be the goal for those services: Wikipedia is an example of an uncommoditized data market where data is shared for altruistic purposes so that everyone can benefit from it. The sharing and browsing of data occurs through the browser, but is not easily read by machine due to its unstructured form. (Stahl, Schomm, Vomfell, and Vossen 2015)

Data products on the data marketplace are sold by data providers. In order to fulfill the definition of a data provider, at least one of the four prerequisites that must be true Stahl et al. (2015):

- The primary business model of the provider must be providing data.
- Provider owns or makes a platform available for others where data can be browsed, uploaded or downloaded in a machine-readable format. The data on the platform must be hosted by the provider and the origin of data must be detailed.
- Provider offers or sells proprietary data hosted by themselves. Similar to previous item, the origin of data must be traceable and transparent. In addition, data that has been processed must include the original sources and details on the method for achieving the result.
- Providers who offer data processing services in the form of data analysis tools must be web-based and provide storable data as their main offering.

Furthermore, government agencies and providers who host their data for free are typically excluded from this definition. This is due to them offering the data as a side effect of their other operations, and they are typically not interested commoditizing data or finding new business models to transform their data into a business. One example of this is the Open Government movement that seeks to provide access to data from governments to increase transparency and citizen participation. (Stahl et al. 2015)

Pricing on the data marketplaces varies largely based on the overall market situation (Muschalle et al. 2012). According to the survey by Muschalle et al. (2012), there exists multiple different types of pricing models that tie into the business model of the data provider. **Usage based pricing** model is used to put a price for each unit (be at data, bandwidth, or time) that the customer uses. This might include pricing the data by the amount of times that the customer queries the data service, or time and effort that it takes for data provider to produce the data

customer desires. Pricing per usage does, however, has a weakness of losing its attractiveness as the marginal costs to produce the data approaches zero – at the time of the research there didn't exist a single data provider who would offer pure per-usage pricing with queries. **Package pricing** is used to provide fixed amount of service for a single payment. This might limit the usage of data in a certain time window (e.g. 1000 queries in a day) or have some other restrictions for the usage of data. Package pricing is a popular choice and is being offered by many data providers. **Flat fee tariffs** are used to give a full access to the offering for a specified amount of time. Although the pricing model is simple and gives guarantees of continued income for the provider, the pricing model is not very flexible from the perspective of consumer as it requires planning into future. To address the concerns of consumers, the provider might offer more flexible short-term contracts. **Two-part tariffs** extend on the previous model: In addition to the fixed fee, the consumers also pay a per-use fee for each consumed unit. The pricing model can, for example, be used from the side of the provider in a way that the fixed part covers the running costs, while the per-use prices bring in the profit. Some software licensing models also use the model: consumers might pay a fixed fee for the base product, and an additional fee based on the amount of users that use the product. **Freemium** model provides two-tier service the consumers by offering the basic service for free, and adding a price to premium features such as data integration. The pricing model of premium features may be any of the formerly mentioned models. Lastly, one of the pricing approaches is also not asking a price at all. Data can be given for free, but providers often use it as a way to attract customer towards their other offerings that do have a price.

Aside from the pricing models discovered in the survey by Muschalle et al. (2012), there also exists alternative pricing models in literature. Koutris, Upadhyaya, Balazinska, Howe, and Suciu (2015) presents a framework for query-based pricing that can be used to automatically assign a price to more complex and custom ad-hoc queries, allowing data providers to extend their offering beyond a limited set of views on the data. Tang, Wu, Bao, Bressan, and Valduriez (2013) created a generic pricing model that builds on the ideas of Koutris et al. (2015) and assigns a value to each tuple (or a row) on the database, with each query priced based on the set of results included in the query. In addition, another paper by Tang, Amarilli, Senellart, and Bressan (2014) details an algorithm based on the idea of setting the price based on the completeness of the requested data – the consumer would pay less if they asked for a limited sample of the whole query.

2.2 Categories of products and users on data marketplaces

Currently existing data marketplaces provide a base for investigating the taxonomy of data products and data users. Surveys done by Muschalle et al. (2012), Schomm et al. (2013), Stahl, Schomm, and Vossen (2014), and Stahl et al. (2015) investigated the various data markets and formed categorizations based on the findings. The first category is the types of data offerings that data providers provide, listed in Table 1. The data offerings are not mutually exclusive allow-

ing a single provider to have multiple offerings from different categories of data products.

Table 1: Different data offerings according to Schomm, Stahl, and Vossen (2013)

Data product	**Description**
Web crawler	Web crawlers are services that seek and gather data from websites automatically based on pre-specified rules. Web crawlers have two categories: Focused crawlers that are typically bound to one specific area of domain (e.g. blogs or social media) and customizable crawlers that can be configured by the customer to gather data from any arbitrary web source.
Search engine	Search engines are services that provide an interface similar to web search engines such as Google. Search engines query the data sources they are attached to based on a set of keywords that are given by the customer.
Raw data	Raw data is data in an unprocessed form typically in the format of lists or tables.
Complex data	Complex data is data that has been processed or refined in some manner.
Data matching	Data matching is a service for correcting or verifying data that the customer already has. Instead of providing complete data sets, data matching can be used to cross-reference data of the customer to their internal data set to discover any discrepancies. One example use case could be checking validity of customer shipping addresses by comparing it to various address databases.
Data enrichment	Data enrichment provides different methods for increasing the value of data itself by altering it (e.g. adding additional information to the data). Data enrichment has three subcategories: Firstly, data tagging adds additional information to the input data in the form of tags, which is typically used to find details about unstructured text data. Second, sentiment analysis can be used to get data on how people feel about products, services and other matters of interests to the customer of sentiment analysis provider. Lastly, data analysis can be used investigate and enrich input data with insights like future trends and forecasts.
Data marketplace	Data marketplaces are also considered a data offering as they facilitate a service where customers can buy, sell and trade data.

Similarly, there exists multiple different users for the data products listed in Table 1. Some of the users are more technical, using the platforms programmatically through APIs, while others abstract the details behind user interfaces and use

data products in order to discover insights that interest them. In a set of interviews over organizations that use data markets, Muschalle et al. (2012) discovered seven different groups of data marketplace users. These users are listed in Table 2.

Table 2: Different users of data marketplaces according to Muschalle et al. (2012)

Data user	**Description**
Analysts	Analysts try to discover trends and insights from data. This leads them to utilize multiple sources of data ranging from public data sets on the web, search engines for discovery, private internal data from enterprises, and data acquired from other services such as data markets. As analysts seek insight by combining data with exploratory techniques, they also create a demand for data relevant to their needs. Members of this group include business analysts, marketing managers, sales executives, and other roles that benefit from analytical techniques.
Application vendors	Application vendors develop applications that make the use of data from data markets easier based on the requirements from analysts. Applications might provide easy-to-use interfaces that lower the barrier of access and allow a broader group of users to take advantage of data from a data market. Alternatively, they might simply provide procedures to query and aggregate data so that it can be used elsewhere. Some example applications include business analytics applications, customer relationship management applications, and enterprise resource planning systems.
Developers of data associated algorithms	To get the data in a form that is usable by analysts and application vendors, it might have to be transformed or otherwise processed into a desired format. Developers of data associated algorithms create pipelines for various tasks, such as data mining, cleaning, matching, and other purposes. These pipelines could also be integrated to the data marketplace as custom functionality that could be bought similarly to the data on the platform.
Data providers	Data providers store, sell and advertise data. In addition, some data providers might also have data integration offerings similar to the developers of data associated algorithms. There exists commercial and non-commercial providers: Non-commercial providers range from web search engines such as Google and Bing, to free-to-access web archives. Commercial providers include companies like Reuters and Bloomberg who sell financial and geographical data.
Consultants	Consultants act as support for organizations that require assistance with tasks related to the selection, integration, and evaluation of data for analysts and product development.

Licensing and certification entities	Licensing and certification entities are sometimes used by the data providers to ensure that data sets, applications and algorithms on the platform are appropriately licensed, or conform to a certain certification. This is also used to assist the customer in choosing data related products.
Owner of the data marketplace	The entity that owns the market. Owner of the data market is responsible for the technical, ethical, legal, and economical challenges that rise from the users of the platform, technical details of the platform, and legal aspects for areas on which the platform operates on.

2.3 Data marketplace structures

The structure of marketplaces affects how people interact on the platform. In this thesis, we describe the overall market structures based on the findings of Muschalle et al. (2012), and a later classification for more specific marketplaces that was founded on electronic marketplace research by Stahl et al. (2016).

Muschalle et al. (2012) divides the market into three categories: Monopolies, oligopolies and strong competition. Monopolies in data markets imply situations where a data provider has no competition. As there exists no alternative data products, the provider can set prices freely to maximize their profits. This allows the provider to do selective pricing, otherwise known as *price discrimination*, to set different prices for different types of customers to optimize profits earned from each customer. Oligopolies are the next step from monopolies. When one or more competitors exist, monopolistic pricing no longer works as the customers would simply move to the competitors offerings. In a competition for market share, providers might adjust pricing competitively ("races to the bottom") or try to compromise with each other to improve profits. A strong level of competition will shift the market towards the ideal of perfect markets as prices of offerings will approach their marginal cost. Transparency between competing providers is improved as consumers of data will want to compare offerings between the different available providers. Providers themselves will no longer have the market-power to set prices (as nobody would buy their product) and must abide by the trends set by the market. However, this market situation carries risks for the data provider: Even if the gross margins from trading data would be profitable, the overall costs of the provider might not be covered by margins that are too thin. To counter this, a provider would have to either add unique value propositions to their products to stand out from competition and justify larger margins, or alternative cut overall costs to avoid loss of profits.

Stahl et al. (2016) created a classification framework for data marketplaces that identifies six data marketplace Business Models on the scale of orientation. The classification is used to indicate how freely users of the marketplace are allowed to trade with each other on the platform; market-oriented structures allow greater freedom of interaction, while hierarchical structures restrict users of the platform to predefined interactions. In addition, Stahl et al. (2016) identified three ownership structures that can affect the neutrality of the platform: Privately

owned, consortia-based, and independent. Private and consortia-based platforms might have vested interests on the platform that can lead to bias towards the owners themselves. Independent platforms, however, are run without connections to providers or consumers which leads to a more even market situation.

The Figure 4 illustrates potential ways to structure a data marketplace based on the ownership model. Starting from the hierarchical private ownership model, the structures in this category are highly restrictive one-to-many or many-to-one relations, limiting the interactions to either procuring or selling data between third parties and the data provider (or the data buyer) who also owns the platform. Consortium-based marketplaces, however, offer more freedom internally with many-to-many relations between both the owner-providers of the platform and the third parties. Even so, consortium-based platforms are typically collaborative efforts by multiple companies and are typically closed by nature and inaccessible by the public. Stahl et al. (2016) also describes many-to-many platforms where the owners participate by trading and selling their own data to be in the same category as consortia-based marketplaces. Although such a marketplace would not be a consortia-based, it would still behave like one due to the bias towards the owners. Finally, the independent data marketplaces operate closer to the principles of free markets by having little to no restrictions for entry and simply acting as a mediators between the consumers and providers. It should be noted that, according to Stahl et al. (2016), there exists little empirical research on data marketplaces aside from few surveys. However, the model provides a base for observing new marketplaces that emerge from new applications made possible by technologies such as cloud computing.

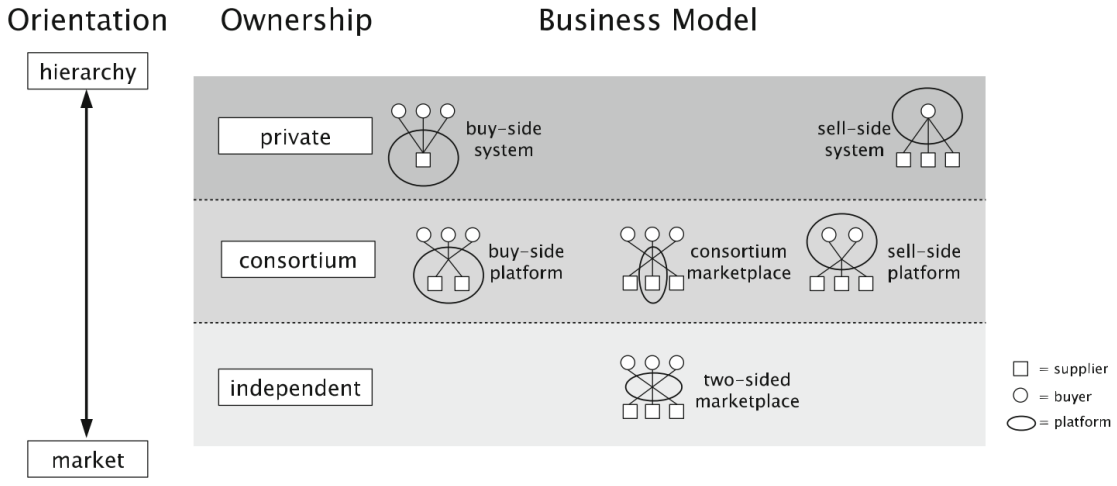


Figure 4: Hierarchy of marketplace structures (Stahl, Schomm, Vossen, and Vomfell 2016)

2.4 Requirements of multilateral data marketplaces

The requirements of data marketplaces can be summarized with findings of Koutroumpis et al. (2017). As the focus of the thesis is on multilateral marketplaces,

we will not discuss requirements for other categories of data marketplaces. For the general structure of the marketplace Koutroumpis et al. (2017) identified five key requirements.

1. *"Thickness" or the liquidity of market* indicates the number of participants on the data marketplace – or a strong enough networking effect – to ensure that there is enough diversity in the offerings and channels of trade. Without a sufficient amount of market participants, the market is unable to reach a critical mass that is required for it to grow in a meaningful manner.
2. *Performance and efficiency* are required to ensure small enough latencies in fulfilling transactions, and to provide enough througput so that the transactions will not slow down as the amount of participants rises. Technological choices and implementation details of the platform are used to address this requirement. Some approaches to performance and efficiency are discussed in chapter 3.
3. *Perception of safeness* affects the degree of trust between market participants. The marketplace should have controls to provide sufficient deterrence for bad actors. Without necessary measures, it might be possible to manipulate the market or otherwise take an advantage over other participants in ways that reduces trust on the marketplace.
4. *Provenance of information* should be provided by the data marketplace. The origin, quality and other attributes of data should be made available to the buyer in order to prevent information asymmetry where the seller of data will know the details and quality of the data better than the buyer. Provenance is investigated with more detail in chapter 4.
5. *Conforming to social and legal restrictions* affects the attraction of the data marketplace from the perspective of the market participants. Trading information with privacy implications, for instance, is a conflicting topic that has many societal and regulative barriers.

The general requirements are fundamental for a data marketplace to stay healthy and maintain growth. To facilitate the transactions on the platform itself, the data marketplace must provide the two functional capabilities. Firstly, a data marketplace must be capable of *matching buyers with sellers*, requiring either a manual mechanism that allows browsing, buying or selling data on-demand, or alternatively an algorithmic matching mechanism that will automatically connect sellers with buyers. Optionally, a functionality of *excludability* can be used to prevent undesirable trades. Second, a data marketplace must be able to *support provenance with metadata* in order to protect data and enable data providers to control their data assets while still allowing for innovative reuses of data.

2.5 Data contracts

One method of supporting the instititution of a data marketplace is through data contracts, providing a framework for marketplace rules and pricing with metadata. Data contracts are a type of service contracts that provide information and guarantees on the nature of the data. The concept is otherwise known as *data agreements*, and the terms are used interchangeably (H. L. Truong, Dustdar, Gotze,

Fleuren, Muller, Tbahriti, Mrissa, and Ghedira 2011). Current research on data contracts is scarce (Koutroumpis et al. 2017) with the exception of H. L. Truong and Dustdar (2009) applying the concept of data contracts in the context of data marketplaces, and then following it with research on different models of data contracts (H. L. Truong et al. 2011; H.-L. Truong, Comerio, Paoli, Gangadharan, and Dustdar 2012). Currently most of the data marketplaces use human-readable data agreements that have limitations in terms of automation and usage in hybrid data products, to which data contracts provide an alternative solution (H. L. Truong et al. 2011).

H.-L. Truong et al. (2012) analyzed properties that are relevant for data contracts in data marketplaces, leading to the formulation of five data contract terms.

- *Data rights* declares the rights that the data provider grants to the consumer of that specific data. This includes the rights of *Derivation* – allowing modifications to the data asset that lead to a creation of a "derivative work", *Collection* – permitting the consumer of data to include the specific data set as a part of a collection of independent data sets, *Reproduction* – giving a right to create temporary or permanent reproductions of the data set, *Attribution* – how the original provider of data set is attributed for the use of data, and finally *Noncommercial use* – whether the right to use data in non-commercial or commercial use is either denied or allowed.
- *Quality of data* metrics may be included in the contract, including specifications such as completeness, reliability, accuracy, consistency and interpretability. The metrics should be based on common agreements that are established on the specific domain of the data set.
- *Regulatory compliance* provides a list of regulations with a set of specifications that describes how the data complies specific regulations. As an example, handling personally identifiable information necessitates strict security measures due to compliance regulations.
- *Pricing model* defines the method of pricing and the cost that the user of data must pay to the data provider.
- *Control and relationship* provides information on contractual obligations, such as warranty, indemnity, liability, and jurisdiction.

These concepts were materialized in a data contract metadata model presented in Table 3.

Table 3: Data contract terms and values in the metadata model by H.-L. Truong, Comerio, Paoli, Gangadharan, and Dustdar 2012

Category	Term representation	Examples
Data rights	*termName*={val_1, val_2, ..., val_n}	*termName*={`Derivation, Collection, Reproduction, Attribution, Noncommercialuse`}, val_i={`Undefined, Null, Allowed, Required, True, False`}
Quality of Data	$val_1 \leq termName \leq val_u$	$val_l, val_u \in [0,1]$ *termName*={`Accuracy, Completeness, Uptodateness`}
Regulatory compliance	*termName*={val_1, val_2, ..., val_n}	e.g. *termName*={`PrivacyCompliance`} *termValue*={`Sarbanes-Oxley (SOX) Act`}
Pricing model	*termName*=(*cost*=val_1, *usagetime*=tal_2, *maximumusage*=tal_3	e.g. *termName*={`MonthlyPayment`}, $val_1 \in \mathbb{R}$ (e.g. `cost=50€`), val_2={(end_t - $start_t$) or `UNLIMITED`} where $start_t, end_t \in datetime$
Control & Relationship	*termName*=*val*	Any key/value string e.g. *termName*={`Liability, LawandJurisdiction`}, *val*={`US, Austria`}

2.6 Summary

In this chapter we introduced the basics of markets. A market provide rules, roles, protocols and pricing mechanisms for market transactions (Stahl et al. 2016). A market transaction has four phases: First the user of the market seeks information on the goods, second the price is negotiated between the buyer and seller, third the transaction is fulfilled and the good is transferred from seller to buyer, and fourth customer support might be provided by the seller after the transaction. (Schomm et al. 2013)

Data marketplaces are electronic marketplaces that operate in the Web, providing infrastructure for trading, browsing, uploading and downloading data (Stahl et al. 2016). Data is an intangible digitized good that makes estimating its value challenging. However, the level of commodization, or the machine readability of the data can help determining its value. Data products on data marketplaces

can be priced in multiple different ways, including *usage based pricing*, *package pricing*, *flat fee tariffs*, *two-part tariffs*, and *freemium models*.

A data marketplace has three general categories of users: *Data consumers*, *data providers*, and *data marketplace owners* (Muschalle et al. 2012). Data providers may have multiple different data product offerings (listed in Table 1) and data marketplaces have multiple different subcategories of users (seen in Table 2).

Data marketplaces can be structured in different ways, affecting the way in which users of the data marketplace can interact with each other. The structure of the data marketplace depends on the business model of the data marketplace. Hierarchical data marketplaces have more strictly defined interactions, while market-oriented data marketplaces place little to no restrictions for user interactions. (Stahl et al. 2016)

The most general requirements for a successful multilateral data marketplace are *market liquidity*, *performance and efficiency*, *perception of safeness*, *provenance of information*, and *conforming to social and legal restrictions* (Koutroumpis et al. 2017). The requirement of performance and efficiency is investigated through data-as-a-Service concepts in chapter 3 and provenance of information is further discussed in chapter 4. Lastly, data contracts provide a framework for data marketplaces that can be used to provide relevant information and guarantees about data products. H.-L. Truong et al. (2012) identified five necessary data contract terms for data marketplaces: *Data rights*, *quality of data*, *regulatory compliance*, *pricing model*, and *control and relationship*.

3 Data-as-a-Service

Data marketplaces provide platforms to trade data, but how is the trading facilitated via internet? This chapter describes what Data-as-a-Service is and how it fits into the current taxonomy of "x-as-a-service" landscape originating from Cloud Computing. In order to understand the data that can move within DaaS platforms, the chapter investigates data formats and the concepts of big data, providing insight on the challenges of data processing on DaaS platforms. In addition to using academic literature found from search engines using relevant keywords, technical literature was used to support some concepts. As much of the technical literature is not freely available, the technical sources were chosen simply based on their available to the author.

3.1 Introduction to Data-as-a-Service

With the introduction of Cloud Computing, abbreviations such as SaaS (Software-as-a-Service), PaaS (Platform-as-a-Service), and IaaS (Infrastructure-as-a-Service) have become well known. Cloud computing moves storage and computation to large data centers with leased services where economies of scale increase efficiency over typical on-premise servers. Service-oriented strategies have a longer history in manufacturing industry, where the approach of finding competitive advantage through service-oriented strategies is known as "servitization". An example of this is provided by Opresnik and Taisch (2015): Hilti International, a drill manufacturing company, created a new product that soon faced competition from another company with a similar product at a lower price. To create a competitive advantage, Hilti International transformed its drill into a service with a "per hole" pricing model. Service-oriented models enable a more economical approach to computing: Services can be scaled up or down depending on the level of workload, and instead of clients doing all the processing separately, services can reduce redundant processing by serving precomputed data to clients or move some computations entirely to the cloud (Dikaiakos, Katsaros, Mehra, Pallis, and Vakali 2009).

Online data marketplaces are DaaS platforms. A DaaS platform can facilitate trading of data through a set of endpoints that communicate via internet. However, not all DaaS are necessarily data marketplaces, as some of them might not have all the prerequisites to be called a data market. For instance, Stahl et al. (2016) suggests that in addition to providing data from machine-readable endpoints, the service provider's primary business model also needs to be centered around providing data and data-related services to be classified a data marketplace.

DaaS platforms seeks to alleviate challenges that come from using data from multiple sources. Different data sources may use different formats, which requires additional engineering effort to integrate together. Standard formats and interfaces are a requirement for enabling economies of scale as they reduce costs of integration for all users of the service (Assunção, Calheiros, Bianchi, Netto,

and Buyya 2015). DaaS platforms solve this by providing a generic API that can be used for multiple independent data sources for common operations (i.e. searching, downloading, uploading and updating), or through a specialized API that operates over the otherwise fragmented data sources (Vu et al. 2012).

There exists various types of DaaS platforms, which tend to differ due to the use cases and requirements of the service. One way to categorize DaaS platforms is the amount of data assets provided on the platform, and the relations between the assets. Based on this categorization, Vu et al. (2012) have observed three abstract types of services:

Generic A generic DaaS platform contains multiple independent data sets that have own APIs and properties. DaaS platforms are centered around data assets and APIs to access them, either using a generic or data asset specific API. Some examples of these include Amazon Data Sets and Microsoft Azure data marketplace.

Specialized Specialized DaaS platforms are either centered around a single, or a limited amount of data assets. The API describes the DaaS platform and its services as a whole, and many operations that can be performed through the API work on the data assets collectively. Additionally, some specialized data assets can be manipulated through their API. A DaaS framework proposed in research such as **Active CTDaaS: A Data Service Framework Based on Transparent IoD in City Traffic** (Du et al. 2016) could be considered an example of a specialized DaaS, although without the capability of modifying the internal structure due to the immutable form of data.

Hybrid A hybrid DaaS platform is an in-between model between Generic and Specialized DaaS approaches where an otherwise generic DaaS platform is used to provide access to data from a Specialized DaaS platform. The data from a Hybrid DaaS could be retrieved using either using a generic API provided by the DaaS platform or the API provided by the underlying Specialized DaaS platform.

The requirements set to DaaS platforms are closely related to requirements set to big data systems. Yin and Kaynak (2015) provides an example of a manufacturing company that gathers sensor data from their machines: A single device alone generates 5000 data samples every 33ms which results in 150 000 samples in a minute, or total of four trillion samples per year. If the company would, for example, want to sell the data from their machines to a maintenance company, the DaaS platform used to facilitate the exchange of data would have to be capable of fulfilling the same requirements as those in Big Data systems.

Building a functional DaaS platform requires use of various technologies. According to Chen, Kreulen, Campbell, and Abrams (2011), there are four categories of technologies that must sufficiently be covered for a DaaS platform to be successful:

Data Modeling tools A DaaS platform will handle data in multiple formats and schemas, which can make it difficult for users to understand and make use of the data. Data Modeling tools allow the data to be modeled and presented consistently, allowing users of the DaaS take advantage of various data sources more efficiently.

Common query language and API A DaaS will be accessed by a variety of users, devices, and applications. For this reason, a single unified API and query language is essential: Firstly, it makes the cost of creating and updating applications that connect to DaaS as low as possible. Second, it separates the internal complexity of the platform away from the users. And third, the concern of generating low-level queries is moved to the DaaS itself, allowing queries to be automatically optimized.

Massive scale data management The scale of data handled by DaaS should be assumed to be massive, requiring massive scale data management. To mention a few examples, issues of scale can be managed with replication of server nodes in order to respond to varying load, partitioning between geographical areas, and by ensuring backups and successful restoration of databases in case of errors.

Data cleansing and processing technologies As DaaS typically provide a common API and query language, it must also have methods of ensuring that the data from a variety of different sources conforms to a similar schema. A range of data cleansing and processing services is therefore required to convert data to the desired schema.

3.2 Properties and processing of data

A DaaS platform must be able to process data with different properties (Chen et al. 2011). In this section, we look at attributes of data, and the constraints placed on handling data as the amount of data grows towards the scale of big data – high-volume, high-velocity and high-variety data that requires innovative approaches to use it properly (Kleppmann 2017).

3.2.1 Data formats and their structures

For data to be readable by computers, it has to conform to some type of schema while preferably staying readable to humans as well. There exists multiple data formats that can be used to transfer and store data on web. The more structured the data structure is, the easier it is to process using automated tools. Assunção et al. (2015) define the following four categories of structuredness of data:

Structured Data that is modeled and follows a certain schema. Easier for computers to process, but not necessarily readable by humans. Li, Ooi, Feng, Wang, and Zhou (2008) mention relational databases as a prime example of a structured data source.

Unstructured Data that has no specified schema or structure (e.g. natural language, video audio etc.). Text documents are a good example of this: Easily read by humans, but a challenge for computers to correctly understand.

Semi-structured Data that might have some structure, but lacks strict guarantees. As an example, many XML- and JSON-documents belong into this category. (Li et al. 2008; Gandomi and Haider 2015).

Mixed A combination of multiple categories of structuredness. For instance, a data format could include both structured fields (e.g. time, location), but

also unstructured data fields (e.g. "description" field writted in natural language).

The two most commonly used data formats are JSON (Javascript Object Notation) and XML (Extensible Markup Language). Both of the formats can contain data in either semi-structured or mixed manner (Kleppmann 2017). In this thesis, the scope will be limited to only briefly introducing the JSON notation.

JSON is a flexible data format, providing support for encoding both key-value objects and lists. Figure 5 shows a structure of a simple JSON document. Key-value objects are surrounded with curly braces and can contain an arbitrary number of key-value pairs. List structures, on the other hand, are indicated with brackets. All text strings are to be surrounded with quotation marks, but numerical values do not have a similar requirement. (Kleppmann 2017)

```
{
  "name": "Elli Esimerkki",
  "address": "Katuosoite 123",
  "age": 52,
  "children": [
    {
      "name": "Erkki Esimerkki"
    },
    {
      "name": "Emma Esimerkki"
    }
  ]
}
```

Figure 5: Example of a JSON document

The key-value property of JSON allow nesting of data in a tree-like structure. For instance, in the example Figure 5, children could have additional children of their own with each of them having their unique details and attributes. The depth of nested data is only limited by the capability and performance of the software reading the JSON document. (Kleppmann 2017)

3.2.2 Properties of big data

With the production and storage of increasing amount of data, the key competitive advantage for many organizations can now be derived from the capability of managing and finding insights from those otherwise overwhelmingly large and complex pools of data. Sensors, finance, accounting and user activity are some of the most major sources of this data, and contribute to the phenomenon also known as "data deluge" (Yin et al. 2015), a downpour of data capable of drowning IT infrastructure that is not properly prepared for it. Similarly, any service that acts as an intermediary for such data at a similar scale has to respond to constraints set by big data.

In this thesis, we will use the definition of big data by Assunção et al. (2015) However, it should be noted that there is not a single universal metric that can be

used to determine whether a certain data set falls into the dimensions of big data as it depends on the size, industry, and location of the organization, and how the definition of big data changes over time (Gandomi et al. 2015). According to some definitions, even exceptionally large, but otherwise mundane files (e.g. a 40MB PowerPoint presentation) could be considered to be big data (Zaslavsky, Perera, and Georgakopoulos 2013).

Variety The amount of different types of data. A large variety of data formats translates into more complex requirements for software used to process the data, as logic for handling different formats must be included (Yin et al. 2015).

Velocity The rate in which the data arrives and is processed, often varying depending on the data source, processing, and network capabilities of the big data systems (Assunção et al. 2015). Processing capability is required in order to transform data into another format or otherwise refine it to some other form, while network limits the total bandwidth that can be used for transferring the raw data (Yin et al. 2015). The highest velocity is **real-time** where a stream of data is received, processed and pushed forward with minimal delays. A step below this is **near time**, which is similar to real-time, but with minor delays. Lastly, **batches** process the data in large chunks, which leads to a noticeable delay. (Assunção et al. 2015)

Volume The total size of data. The larger the data, the more there is overhead in moving data from place to place. This introduces the performance benefits of *data locality*. With larger data sets it might be preferable to process the data as close as possible to the data source in terms of network distance. This maintains the ratio between the time it takes to transfer data, and the time it takes to process the data.

Veracity How well the data can be trusted. As an example, data on subjective opinions of people might not objectively correct (e.g. customer reviews, feedback), but still provide valuable information (Gandomi et al. 2015). In the context of DaaS and data markets, veracity indicates the trustworthiness of specific data sources.

Value The value of data in comparison to its volume (*"value density"*). big data typically has low value density, but has the capability of being transformed to high value with enough volume (Gandomi et al. 2015). However, the metric of value also depends on the organization's capability of finding value from data (Assunção et al. 2015).

3.2.3 Stream Processing

Stream processing is an approach where data is processed instantly as it arrives (Figure 7) as opposed to batch processing where data is processed in chunks or intervals (Figure 6). Traditionally, gathering and processing data from multiple different sources has been facilitated using ETL (Extract, Transform, Load) tools that move data to data warehouses in a batch-type fashion. Stream processing systems provide an alternative to the traditional model (Kleppmann 2017).

In many real-world applications, such as financial transactions and IoT devices, data is created in continuous streams of events. An event represents a single, self-contained object that contains details about something that has

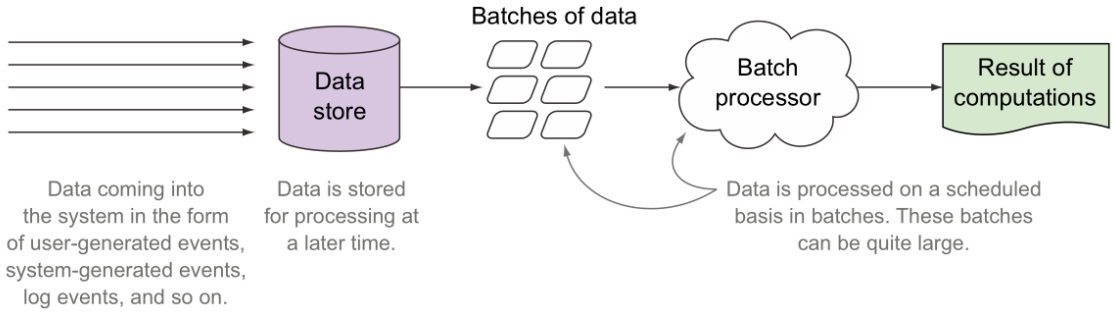


Figure 6: Batch processing (Allen, Jankowski, and Pathirana 2015)

happened at some point of time (Kleppmann 2017). Although such data could also be processed in batches, in some cases it would be more appropriate to process it without delays. Stream processing addresses some drawbacks of processing analytics in batches: When using batches, the data is never processed real-time, which affects the timeliness of the discovered insights. This becomes a problem with "perishable" data: The longer it takes to analyze data points, the less value they provide (Flannagan 2016). Furthermore, stream processing can be used to solve issues of scalability: Handling data at scale for systems such as airline ticket booking or retail systems is difficult because they rely on multiple different and distributed systems that might end up in conflicting states. Stream processing can be used to solve some of the problems as the abstraction of events provides capabilities for resolving those conflicts. (Friedman and Tzoumas 2016).

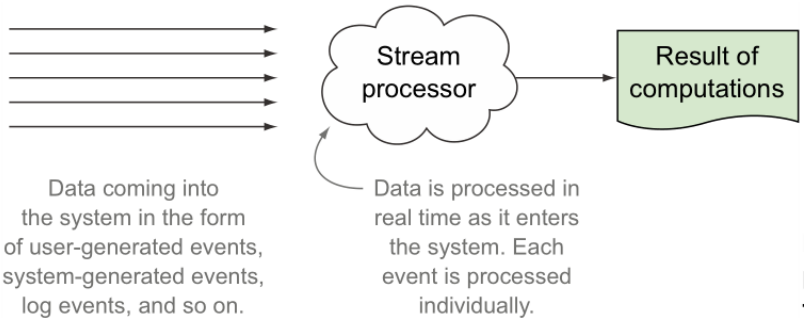


Figure 7: Stream processing (Allen, Jankowski, and Pathirana 2015)

Ideas behind stream processing are partially related to an earlier technology called Complex Event Processing (CEP) Narkhede, Shapira, and Palino (2017). CEP was developed in the 1990s for analyzing streams with certain patterns that were defined with query languages such as SQL. Unlike traditional databases that would respond with a one-time response, the queries would define channels with potentially infinite events. Although the concepts are similar to each other, the main difference is found from the analytics part of stream processing: Instead of discovering specific event chains, stream processing is more focused in statistics and averages. However, features such as defining streams with SQL are being adapted by modern stream processing frameworks. Some of those stream processing frameworks include Apache Samza that introduced SamzaSQL in 2016 (Pathirage, Hyde, Pan, and Plale 2016), Apache Kafka that released their preview

of KSQL in 2017 (Akidau, Chernyak, and Lax 2018), and Apache Flink that features a CEP library as a part of the framework (Friedman et al. 2016). (Kleppmann 2017)

There are three possible ways to use stream data sources: Firstly, The stream can be written into different types of storages (e.g. databases, caches) where it can be used as the application state. This can involve updating a traditional database, or taking an advantage of a technique called event sourcing that provides an alternative for storing state. In a banking application, for instance, the state could include user accounts, balances and transfers. With event sourcing, however, that same state could also be contained as a series of events that could be used to form the state itself on demand. Using the same banking example, a balance of a bank account could be kept as a single column of data in a database that is updated at each transaction. Alternatively, the balance could be calculated every time on the fly by summing together all the events where money is transferred to an account and negating the sum of transactions where money was moved away from the account. Another approach is taking an advantage of both databases and event sources: A database can be used to reflect the current state while also retaining the log of events for other purposes. Second, the events of stream data can be sent directly to users in form of real-time dashboard or notifications. And third, the input stream can be transformed with other streams to form new kinds of output streams; an approach that can also be applied to data marketplaces and hybrid data products. (Friedman et al. 2016; Kleppmann 2017)

According to Stonebraker et al. (2005), there are eight rules that stream processing systems must follow:

Keep the Data Moving The system must be able to handle message processing with minimal latency. This means avoiding bottlenecks in the processing path (e.g. costly storage operations). The messages should be processed "on the fly" as they arrive.

Query using SQL on Streams It's desirable to process real-time data using a high-level language like SQL instead of relying on lower application programming languages (such as Java and C++) as it results in shorter development cycles and lower maintenance costs. As SQL is primarily intended for use with relational databases, it needs additional features to fulfill needs of stream processing systems.

Handle stream imperfections The stream system must be able to handle situations where the stream data might arrive late, out of order, or with some data missing entirely. Although some processing might rely on waiting for all data to arrive, the data might never arrive, leading to a standstill. Therefore every function in the stream system should have a time out period that allows continuing even with incomplete data.

Generate predictable outcomes The output of the stream processing system should be deterministic and repeatable. In other words, providing the same input data in the same order should always lead to the same results. In addition to ensuring the correctness of the system, deterministic and repeatable results are necessary when the system has to recover from a failure and the same application state before the crash has to be replicated.

Integrate stored and streaming data The stream processing system must be capable of combining stored and stream data together. This is commonly used

in applications where messages that happened in the past are used together with current data (e.g. fraud detection). In addition to maintaining history in the format it arrived in, another approach is to maintain signatures of "normal" and unusual data based on past behavior as a summarization of earlier messages. Additionally, stored data is useful during the development of applications, allowing the developer to conduct repeatable experiments with past data before switching to live data. It should be noted that interfacing with external databases can add notable latency to the pipeline, and would conflict with the rule of "Keeping the data moving".

Guarantee data safety and availability For the sake of reliability and fault tolerance, the stream processing system must, by design, be highly available. In a case of hardware failure, the stream processing system should be able to restore itself with little to no delay. Longer downtimes are unacceptable for real-time processing systems, as the system would not be "real-time" for the period of the downtime.

Partition and scale application automatically It should be possible to distribute the stream processing system over multiple machines due to the scalability and price-performance benefits offered by clusters built with commodity hardware. The system should be able to scale automatically and transparently to available machines depending on the system load. Additionally, the stream processing system should support multiple threads to take full advantage of multiple processor cores in modern hardware.

Process and respond instantaneously The processing system must be able to process high volumes of data with very low latency, typically in the range of microseconds to milliseconds on commodity off-the-shelf hardware. This requires that all the system components are designed with performance in mind to prevent bottlenecks. However, the performance of the system might also vary depending on the type of workload, making this rule workload-dependent.

3.3 Summary

Online data marketplaces are Data as a Service (DaaS) platforms that facilitate trading of data via internet. DaaS provides a framework of standard formats and interfaces that enable operating data products through an API. There exists three general categories of DaaS platforms: *Generic* platforms with multiple independent data sets, *specialized* platforms that are centered around a single (or limited amount of) domain specific dataset(s), and *hybrid* platforms that may provide an access to specialized datasets in addition to a generic API. (Vu et al. 2012)

As the data products on a DaaS platform might extend to the scale of big data, a DaaS platform must be capable of handling big data. Technologies required by a DaaS platform include *data modeling tools* for humans to handle the complexity of data, *common query language and API* to access and manipulate the data programmatically, *massive scale data management* to respond to challenges of big data, and *data cleansing and processing technologies* to process, clean, and transform data if necessary. (Chen et al. 2011)

Data handled by a DaaS platform can have different properties. The level of structuredness of data determines how easily it is handled with automated tools. The two most commonly used data formats, JSON and XML, can contain both semi-structured and mixed data. JSON is a flexible data format that can contain data in a tree-like structure using key-value pairs. (Assunção et al. 2015; Kleppmann 2017)

Properties of big data include *variety* that describes the diversity of processed data, *velocity* as the speed or throughput of data, *volume* representing the total size of data, *veracity* determining the trustworthiness of data, and *value* of data in terms of density of valuable information in comparison to noise (Assunção et al. 2015). Data can be processed using *batch or stream processing*, with latter approach being the more viable choice for real-time use cases (Kleppmann 2017; Flannagan 2016). Stream processing systems must *keep data moving*, *be able to use SQL on Streams*, *handle stream imperfections*, *generate predictable outcomes*, *integrate stored and streaming data*, *guarantee data safety and availability*, *partition and scale application automatically*, and *process and respond instantaneously* (Stonebraker et al. 2005).

4 PROVENANCE

The second chapter provided details of data marketplaces and their special requirements. The third chapter introduced the concepts of DaaS, properties of data, and stream processing. In this chapter, we introduce concepts of provenance, investigate how provenance reflects to data, and discover what existing solutions exist for tracking provenance. Some literature used in this chapter was discovered on search engines using relevant keywords, while references to W3C specifications were gathered directly from the source.

The concept of provenance has been used in the context of art to preserve history of artifacts, denoting the lineage of ownership for objects. The provenance record is used by scholars and collectors to verify the origin of art, its authenticity, and its price (Deelman et al. 2010). Likewise, the concept of provenance has been used in digital libraries to document the usage history and life cycle of digital books and documents (Moreau, Freire, Futrelle, McGrath, Myers, and Paulson 2008).

When applied to data, provenance refers to the origin of data in terms of time and place. It answers the questions of how was data created or generated, what intermediate processing steps have occurred to the data, and through which parties has the data passed through (Lim, Moon, and Bertino 2009; Deutch, Frost, and Gilad 2018). The questions are important in order to understand the properties of data that originate from multiple sources. Provenance becomes increasingly important as the complexity and the amount of different data sources rise, as tracking the origin of data would become more and more challenging (Deutch et al. 2018). Provenance is also a useful concept for scientists as it can be used to ensure reproducibility of data analysis. Provenance can indicate how the results of the research were derived, with what parameters, and from which data sets (Moreau et al. 2008; Deelman et al. 2010). Provenance of data has applications in diverse range of different domains. Lim et al. (2009) provides an example of a situation where provenance of data is vital in order to ensure that correct decisions are made in life and death scenarios:

> "A battlefield monitoring system gathers enemy locations from various sensors deployed in vehicles, aircrafts, and satellites and processes the monitoring queries over these streaming data. In this system, we need to assess the collected data since we must make sure that mission critical applications only access highly trustworthy data in order to guarantee accurate decisions by these applications. Since sensors and communication lines have different accuracy and confidence, it is essential to know the provenance of each data for assessing its trustworthiness level."

Being able to track provenance is helpful for ensuring quality of results, enforcing policies and ensuring trust in the system (Deutch et al. 2018). When extended to data marketplaces, data provenance has a crucial role of linking data back to the data providers where the data originated from. The value of the data on

a data marketplace is directly linked to data provenance as it provides information quality and legality of data (Koutroumpis et al. 2017).

Provenance data is contained within metadata. Metadata, as defined in the introduction, is "structured data about an object that supports functions associated with the designated object" (Deelman et al. 2010). In other words, metadata can be used to contain information that is not part of the data itself, but that can be used to support and compliment the operations and processes related to the data. Metadata is organized in a set of attributes that belong to a specific schema. A schema can be an application specific practice or a commonly used standard, that dictates the names of the attributes, the type of data that is expected to be found from those attributes, and when and how should the attributes be used. Deelman et al. (2010) provides some general metadata categories that are found from various metadata schemas used by scientific applications:

- *Logical file metadata*, metadata related to files that, for example, includes logical file names, data types, and creation and modification timestamps.
- *Logical collection metadata*, metadata on collections of files or other data, including data on collection name, collection contents, creator, modifiers, and audits.
- *Logical view metadata*, metadata on the logical files, collections and other views that are used to form a specific presentation to data
- *Authorization metadata*, metadata on who is allowed to access the data (i.e. access permissions)
- *User metadata*, metadata on the users of data, including names, phone numbers and addresses
- *User-defined metadata*, custom metadata attributes that have been defined by the users of metadata, extending the schema onto which the metadata is based on.
- *Annotation attributes*, metadata in the form of unstructured text written by the user to describe data or its attributes.
- *Creation and transformation history*, provenance metadata on how data was created, and what type of transformations were performed to the data.
- *External catalog metadata*, linking to metadata that is available from an external resources.

The concept of provenance has been investigated at multiple different levels of abstraction where each domain has its own challenges. Herschel, Diestelkämper, and Ben Lahmar (2017) classified provenance research to four different categories. The categories can be placed to a hierarchy according to granularity, how specific the domain is, and how high the level of instrumentation available for collecting provenance for that type is. We first start from the most fine-grained category, and move upward to less strict categories.

Data provenance The highest resolution of provenance tracking that allows tracking of individual data items and the operations that they go through. The level is typically viable only for structured data models and declarative query languages (e.g. SQL) as they allow highest possible degrees of instrumentation with clearly defined semantics and operators. Data provenance research is divided to two categories: Provenance of *existing results* that detail provenance of data that was given out as a result, and *missing results*, providing

provenance to data that was left out of the results, and more specifically, explaining why it was left out. Provenance of existing results seek to answer three main questions: *where* does the data originate from, *why* were they included in the result, and *how* was the data processed or modified in order to provide the given result.

Workflow provenance Instead of observing individual data, workflow provenance covers inputs, processes, and outputs of systems. Workflow provenance can be presented as a directed graph, with processes being nodes of the graph, and edges connecting them to form the workflow. The level of granularity of the provenance can vary between solutions. Some solutions might track provenance at the level of individual data items, while others are more coarse and are not concerned with individual changes to data. Fine-grained workflow provenance is close to data provenance, but does not necessarily fulfill the requirement of having operators with clearly defined semantics.

Information system provenance Information system provenance is metadata about processes that belong to an information system and have the role of storing, communicating or distributing information. Information system provenance generally considers all processing steps as "black boxes", but may use input, output and the parameters of the process to collect provenance. One example of information system provenance is provenance-aware storage systems (PASS) (Muniswamy-Reddy, Holland, Braun, and Seltzer 2006) that track provenance of files and documents through metadata. PASS is also a valid example of provenance tracking on level of the operating system (Carata, Akoush, Balakrishnan, Bytheway, Sohan, Seltzer, and Hopper 2014).

Provenance metadata The most general category of provenance that includes any arbitrary provenance related metadata. As a "catch-all" category, no assumptions or restrictions are made regarding how provenance is defined, collected, or modeled. The only requirement is that the metadata must be intended for purposes of provenance. Proprietary provenance solutions typically fall into this category as their internals are not publicly disclosed, and therefore their mode of operation can't be assessed.

Concepts of both data provenance and workflow provenance are both highly specific to the domain of databases and workflow system (Herschel et al. 2017), but the categories of information system provenance and provenance metadata are flexible enough to be used in multiple different scenarios. In the following section, we investigate an existing provenance metadata model.

4.1 Open Provenance Model and PROV

Open Provenance Model is a model collaboratively designed by groups of researchers from various disciplines with the shared goal of attempting to find a standard a model for provenance (Moreau et al. 2011). After the model reached sufficient maturity, it cornered enough support to form a base for a W3C (World Wide Web Consortium) specification. The model was implemented by W3C Provenance Group in the form of PROV, a concrete data model and technical

specification (Moreau, Groth, Cheney, Lebo, and Miles 2015) that can be used to build applications that use provenance metadata.

Open provenance model was designed to fulfill the following requirements:

1. The model must allow sharing provenance information between different systems with means of a shared provenance model
2. The model must allow for development and sharing of tools that operate on the model
3. Provenance in the model must be defined precisely, but without describing a specific technical implementation
4. The model must be able provide a digital representation of provenance regardless of the nature of the thing being observed
5. The model must allow for multiple levels of detail that can coexist
6. The model must define set of core rules that can be used to identify and infer relationships

To address these requirements, the group opted for a representation in the form of directed graph, and defined a set of semantics and rules for the nodes, edges, and their composition. There are three types of nodes. Firstly, **Artifacts** denote a state, representing a physical object, or a set of data. Artifacts are represented as circles. Second, **Processes** represent actions that are performed on artifacts (or due to them), resulting in new artifacts. Processes are denoted as squares. Third, **Agents** are entities in the process that facilitate, control or affect its execution. Agents are represented as octagons. Nodes are linked together with edges, represented with arrows, that communicate a relationship between each of the linked nodes. Some of the common relationships include *derived from*, when the artifact is derived from an earlier state, and *triggered by* when an agent or process has a causal effect to something. A basic example is provided in Figure 8, where a provenance graph of an incrementing operator is modeled at two levels of granularity. Thin and dotted arrows denote the direction of actions, while thick arrows represent derivation relationships. Although the graphs are different, they both observe the same events. In the context of OPM, these are considered a two related *accounts* observed from different perspectives. (Moreau et al. 2011)

The OPM specification provides a generic set of concepts and tools, but also provides freedom to extend the model for practical domain specific problems. OPM provides two mechanisms for this in the specification: *OPM annotation framework* and *OPM profiles*. (Moreau et al. 2011)

OPM annotation framework allows arbitrary information to be attached to relations, entities, agents, edges, and even other annotations. Annotations in OPM are simply a special class of OPM entities that can also have their own relationships and chain of provenance. An annotation entity must contain **a)** an identifier that allows linking it with the annotated object, **b)** A set of key-value pairs to contain the additional information provided by the annotation, and **c)** the subset of accounts from the annotated entity in which the annotation is relevant. (Moreau et al. 2011)

OPM profiles allows building extensions or specializations on top of OPM, while still staying compatible with the general principles and graph features. OPM profiles provide a tool for communities to create their own best practices and usage guidelines that best fit the specific domain to which the profile is applied on.

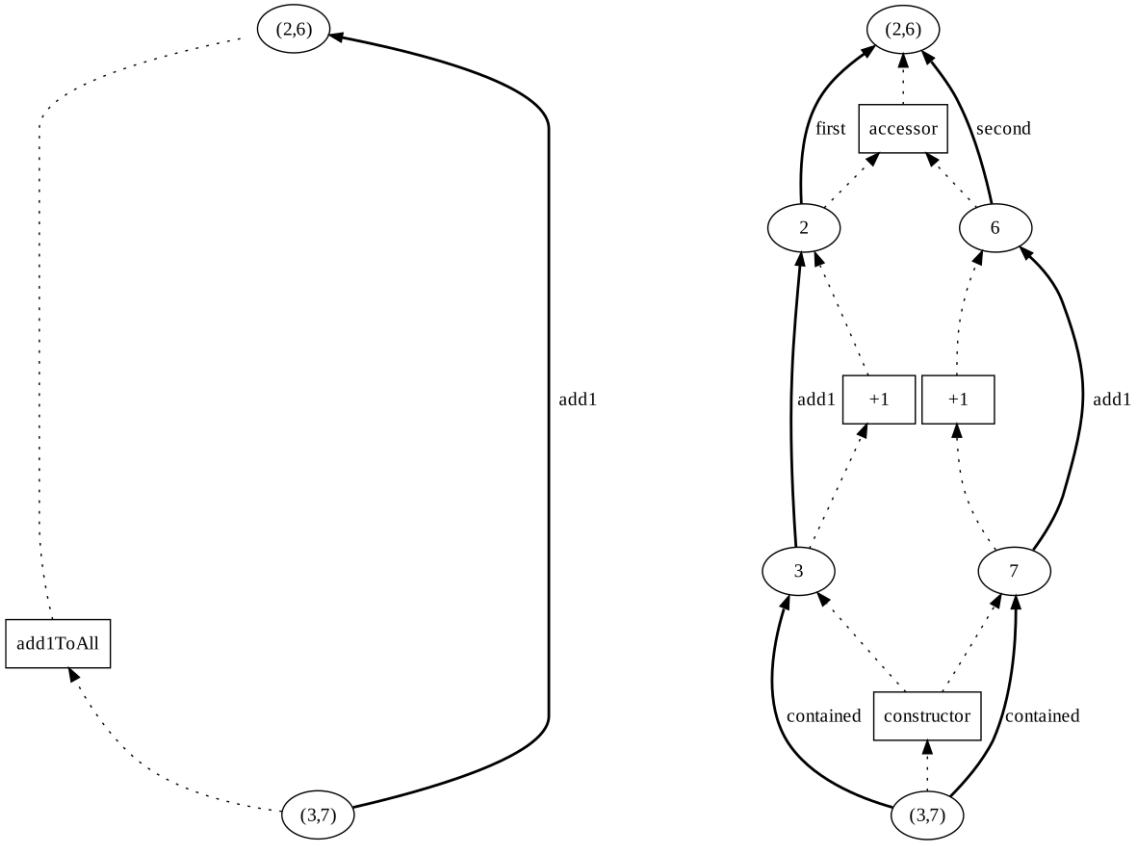


Figure 8: Examples of a provenance graph (Moreau, Clifford, Freire, Futrelle, Gil, Groth, Kwasnikowska, Miles, Missier, Myers, Plale, Simmhan, Stephan, and den Bussche 2011)

An OPM profile consist of a unique global identifier and contains one or multiple components. (Moreau et al. 2011)

1. *Controlled vocabulary for annotations:* Defining a set of key-value pairs, and the allowed types for values. Allowed values can be a predefined set (e.g. a key with the name "entrance" could only have values "open" or "closed"), or a general type (string, boolean, numeric etc.). The controlled vocabulary can be used to create custom types and add an arbitrary amount of domain-specific properties to nodes and edges.
2. *General guidance to expressing OPM graphs:* Guidance is used to provide instructions on the level of granularity, the level of expressiveness, and providing instructions for which types of nodes and edges should be present in the graph.
3. *Profile expansion rules:* Providing a set of rules that can be used to translate an OPM graph using the profile into a generic "profile expanded" OPM graph that requires no knowledge of the profile.
4. *Serialization specific syntax:* Instruction for serializing and deserializing the OPM graph. The component must explain how syntactic operations convert the graph into an arbitrary form of data, and how the data can be used to reconstruct the graph.

The core of Open Provenance Model created a foundation for W3C PROV, a W3C specification for a data model that enables provenance interchange on the Web (Gil, Miles, Belhajjame, Deus, Garijo, Missier, Soiland-Reyes, and Zednik 2013). Similar to Open Provenance Model, PROV expresses provenance in the form of entities and relationships between them, but without the same focus in graphical representation. Instead, PROV is mostly concerned with the concrete model of the data and how it can be used to accommodate provenance data through multiple different perspectives. Some of these perspectives include *agent-centered provenance*, where provenance is formed through interactions of people and organizations, *object-centered provenance* that follows the origin of documents or data, and *process-centered provenance* that focuses on actions and processes used to generate and handle data or objects. The choice of perspective depends on the view and the type of provenance information required by the use case. (Gil et al. 2013)

To enable enough flexibility for various use cases, the PROV data model (PROV-DM) features additional concepts that were not present in Open Provenance Model, but also simplifies the model of extending and customizing by allowing domain-specific information to be embedded more freely along with the rest of the data model. PROV-DM has six conceptual categories that can be used as building blocks for provenance metadata models, but in this thesis we will only focus on the most vital three concepts that form the core of PROV-DM that are also illustrated in Figure 9 (Gil et al. 2013):

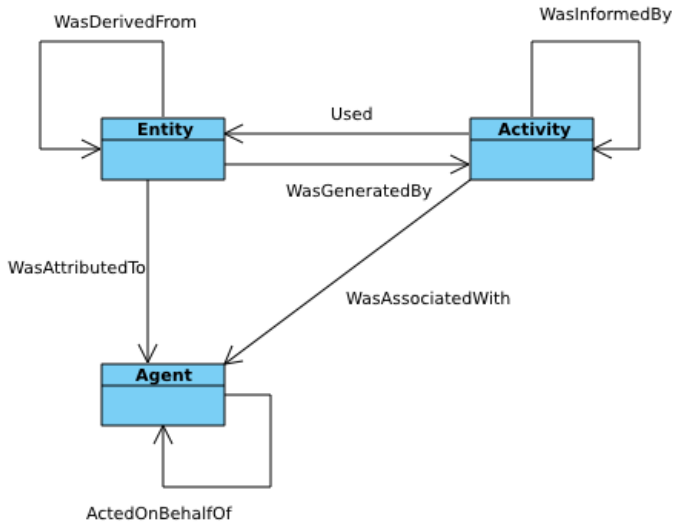


Figure 9: Essential PROV components and their relationships (Moreau, Missier, Belhajjame, B'Far, Cheney, Coppens, Cresswell, Gil, Groth, Klyne, McCusker, Miles, Myers, Sahoo, and Tilmes 2013)

Entities and activities Entities and activities are conceptually same as artifacts and processes of OPM. As described by Moreau, Missier, Belhajjame, B'Far, Cheney, Coppens, Cresswell, Gil, Groth, Klyne, McCusker, Miles, Myers, Sahoo, and Tilmes (2013), "an entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary".

Activities actions that can occur alongside with other activities, typically either *generating* or *using* entities, or *communicating* with other activities.

Derivation When an entity is altered or transformed into a new entity, it can be said that the new entity *was derived from* the former entity. Derivation is used to preserve the chain of transformations so that the origin of data can be inferred later. Some examples of derivation include creating a painting from a canvas, melting ice into water, and modifying data in a relational database to a new form. (Moreau et al. 2013)

Agents An agent is something that has influence on entities, activities, or other agents. An agent could, for instance, be a person, an organization or a software program. The concept of agents is used to signify the effect that different agents could have to the outcome. PROV recognizes that an entity can be *attributed* to an agent, agents can be *associated* with actions, and agents can *delegate* actions to other agents (Moreau et al. 2013). For instance, information that is attributed to a reputable news organization is more likely to be reliable than information that is attributed to a rumor circulating in social media.

These three concepts are codified to a syntax seen in the Table 4. The syntax can be used to generate a provenance record as part of a software program when the data is being processed. Although we will demonstrate the usage of PROV with pseudocode, PROV has bindings to multiple different programming languages, including Python, Java and Javascript (K. C. London 2018). In the following pseudocode example, we create a provenance record where an action `action1` took an entity `example1` yesterday and finished transforming it to `example2` today:

```
1  entity(example1, [])
2  activity(action1, now(), yesterday(), [])
3  used(action1, example1)
4  entity(example2, [])
5  wasDerivedFrom(example2, example1, action1)
```

Table 4: Core syntax of PROV and basic usage. Abbreviations are expanded in Table 5. **`Bold`** parameters indicate that they must be present, while those with `normal` weight are optional. (Moreau, Missier, Belhajjame, B'Far, Cheney, Coppens, Cresswell, Gil, Groth, Klyne, McCusker, Miles, Myers, Sahoo, and Tilmes 2013)

Category	**Name**	**Syntax**
Entities / Activities	Entity	`entity(`**`id, attrs`**`)`
	Activity	`activity(`**`id, st, et, attrs`**`)`
	Generation	`wasGeneratedBy(`**`e, a,`** `t, attrs)`
	Usage	`used(`**`a, e,`** `t, attrs)`
	Communication	`wasInformedBy(`**`a1, a2,`** `attrs)`
Derivations	Derivation	`wasDerivedFrom(`**`e2, e1,`** `a, g, u1, attrs)`
Agents, Responsibility, Influence	Agent	`agent(`**`id, attrs`**`)`
	Attribution	`wasAttributedTo(`**`e, ag,`** `attrs)`
	Association	`wasAssociatedWith(`**`a, ag,`** `pl, attrs)`
	Delegation	`actedOnBehalfOf(`**`ag2, ag1,`** `a, attrs)`

Table 5: Explanations for abbreviations used in Table 4.

`e`	Entity
`a`	Activity
`ag`	Agent
`pl`	Plan
`attrs`	Attributes (e.g. `[name="example"]`)
`id`	Unique identifier
`t`	Timestamp
`st`	Starting timestamp
`et`	Ending timestamp

4.2 Earlier PROV research and implementations

PROV has been used as a foundation for provenance metadata implementations at many different domains. In this subsection, we list some of them and discuss how they defined the requirements of the provenance model, and how the model was evaluated. This subsection provides background and additional reasoning for our custom implementation.

In the course of investigating existing solutions, it was found that provenance systems have been investigated from multiple perspectives, including usability, performance, and applicability. Both qualitative metrics have been used in the form of interviews (Ramchurn, Huynh, Wu, Ikuno, Flann, Moreau, Fischer, Jiang, Rodden, Simpson, Reece, Roberts, and Jennings 2016), and quantitative metrics in the form of benchmarks (Mohamed Jehad Baeth and Aktas 2017; Tas, Baeth, and Aktas 2016). The following list provides some examples of how PROV has been used in different domains:

- **A Disaster Response System based on Human-Agent Cooperation** Ramchurn et al. (2016) developed a provenance model for tracking information from disaster sites. Requirements of the system were gathered through interactions with emergency responders, and it was found that a provenance system should be able to **1)** tell the source of information **2)** explain how that information was transformed into decisions **3)** link complimetary sources of information together. The system was evaluated by describing the system to focus groups, demonstrating the functionalities in rescue scenarios, and by gathering feedback.
- **Recommending Energy Tariffs and Load Shifting Based on Smart Household Usage Profiling** Fischer, Ramchurn, Osborne, Parson, Huynh, Alam, Pantidi, Moran, Bachour, Reece, Costanza, Rodden, and Jennings (2013) presented a system for personalized energy-related recommendations. In order to increase confidence of recommendations, provenance was implemented with the requirements of being able to **1)** justify why data was required for privacy reasons, and **2)** how data how the presented information was computed. The system was evaluated with data gained from semi-structured interviews of the users.
- **Modeling Information Diffusion in Social Media as Provenance with W3C PROV** Taxidou, De Nies, Verborgh, Fischer, Mannens, and Van de Walle (2015) proposes a model known as PROV-SAID, an extended model of provenance based on PROV that is able to track the spread of information in social media (i.e. *Information diffusion*). The objective of PROV-SAID is to provide as high expressive capability for information diffusion as possible, but the model is not tested.
- **A Large Scale Synthetic Social Provenance Database** Mohamed Jehad Baeth et al. (2017) created a synthetic database for storing provenance of social interaction on social media in standard PROV notation. The requirements were derived from provenance database requirements and usability. Lastly, the database was evaluated through synthetic benchmarks.
- **An Approach to Standalone Provenance Systems for Big Social Provenance Data** Tas et al. (2016) contributes to provenance research by providing

test suite to evaluate provenance systems, and by proposing a software architecture for a decentralized and scalable provenance management system to manage big social provenance data. The requirements were defined according to the constraints of big data, requiring high scalability and performance. Although the proposed architecture was not evaluated, existing provenance systems were benchmarked using the test suite.

4.3 Summary

Provenance answers the questions of how, when, and by whom has the object of interest been handled (Lim et al. 2009). Provenance metadata encodes the provenance information as a record alongside the object of interest. In the context of data marketplaces, provenance metadata can contain information on when the data was created, how it has been transformed, and through which parties has the data passed through (Lim et al. 2009). Provenance information is vital for linking data back to its source and providing details regarding the quality and legality of data (Koutroumpis et al. 2017).

Metadata is data that is not part of the data of interest, but is able to support operations and processes related to it (Deelman et al. 2010). However, usage of metadata is not the only approach for tracking provenance. In addition to provenance metadata, provenance can be tracked through *data provenance*, *workflow provenance*, and *information system provenance* (Herschel et al. 2017).

There exists a standard model for expressing provenance – Open Provenance Model (OPM) provides a set of building blocks that can be used to model provenance through graphs. The provenance graph consists of three different types of nodes: *artifacts* denoting state, *processes* representing actions, and *agents* being entities that factilitate, control or affect execution of processes. The generic OPM concepts can be extended to domain specific problems using either *OPM annotation framework* or *OPM profiles*. (Moreau et al. 2011)

W3C PROV is an implementation of OPM, providing a framework and a syntax for creating provenance documents using the programming language of choice. PROV both extends OPM by providing additional building blocks for more complex use cases, and simplifies it by allowing embedding of domain specific information (Gil et al. 2013). W3C PROV has been used in multiple different domains in earlier research, and the resulting implementations have been evaluated either with qualitative and quantitative metrics.

5 A DATA MARKETPLACE METADATA MODEL

In the second chapter we looked into what data marketplaces are and what requirements they have. In the third chapter, we looked at DaaS and the general technical concepts behind them. Based on those insights, we form a metadata model that can be used to track provenance and other attributes required on data marketplaces. Finally, the metadata model is evaluated empirically by applying it to a prototype where data from multiple sources is used to form a hybrid data product. The model is evaluated based on objectives derived from literature and a set of benchmarks.

5.1 Requirements, Objectives and Evaluation

Following the principles of DSRM, we first *infer* objectives that can be used to measure the solution. These objectives can be either qualitative or quantitative. Qualitative objectives are similar to functional requirements found in engineering disciplines by describing how the solution would address the problem through its functionality. Quantitative objectives, on the other hand, reflect non-functional requirements that measure specific metrics, such as speed or performance, and can be directly benchmarked against other solutions. With the background theory described in earlier chapters, we can define the objectives of the solution based on existing literature and the reasoning behind each decision. The requirements are described according to RFC 2119 (Bradner 1997) that provides an industry standard manner for wording: Requirements that 'must' be fulfilled are vital components of the model, while requirements worded with 'should' are opinionated choices or complimentary features of the model.

5.1.1 Objectives

In the most general form, the two main requirements of the data marketplace provenance metadata model are:

1. **The metadata model must be able to represent the chain of provenance of hybrid data products** According to Koutroumpis et al. (2017), maintaining provenance of data is one of the key functionalities of a data marketplace. In this thesis, we constrain the definition of provenance to only include events that occur in the context of a data marketplace, and more specifically, in the creation of hybrid data products. The constraints are made in order to firstly to keep the scale of the thesis reasonable, and second, to restrict the resolution of the model, as provenance interactions at the level of server hardware and network transmission are uninteresting in the context of the thesis.
2. **The metadata model must be able to contain metadata relevant to data marketplaces** In addition to tracking provenance, data marketplaces must

track the metadata related to data products (Koutroumpis et al. 2017) in order to support institution of the marketplace discussed in section 2.1. In this thesis, we define "metadata relevant to data marketplace" to be equivalent with data contract terms provided by H.-L. Truong et al. (2012) (discussed in section 2.5). The data contract terms include *data rights*, *quality of data*, *regulatory compliance*, *pricing model*, and *control & relationship*.

3. **The performance cost of the metadata model must not hinder the the system by a significant degree** The performance drawback of the metadata model should be reasonable in real-world scenario. A metadata model that hinders the usage of a data marketplace is unlikely to be useful in practice. As creation of hybrid data products require accessing resources found from external resources, we consider significant latency to be in tens of milliseconds; latency similar to a hard drive disk seek or a network request.

In addition to the two base requirements, we introduce two additional requirements that further distinguish the thesis from earlier research:

- **The metadata model should apply to both stream and static data** There already exists models that can apply to static data sets (Vu et al. 2012; H.-L. Truong et al. 2012). To provide a new perspective, we apply the model in streaming context.
- **The metadata model should be JSON based** As an opinionated choice, JSON provides a less verbose format of data in comparison to XML (Kleppmann 2017). Earlier research has used XML (Vu et al. 2012) and Resource Description Framework (RDF) (H.-L. Truong et al. 2012) in their metadata representations. Using JSON as a base of the metadata model might provide unique insight as using different data formats may have different challenges and implementation details.

5.1.2 Evaluation and metrics

The evaluation approach and type of artifact was chosen based on the authors preferences and on the findings of Peffers et al. (2012) that charted the prevalence between various evaluation techniques and artifact types. The evaluation methods that have been used in other literature are listed in the Table 6. From these, the prototyping method was chosen to be the evaluation approach for the thesis. As the goal of the artifact is to demonstrate functionality of the metadata model through a novel use case, an example that is grounded in a potential real-world application would demonstrate the utility of the artifact effectively.

An expert review was also considered as a method for evaluating the resulting artifact in this thesis. Although the combination of instantiation and expert review in design science has not been commonly used in information system and computer science research (Peffers et al. 2012), PROV models have been evaluated with interviews and other qualitative methods (Ramchurn et al. 2016; Fischer et al. 2013). A subject-based experiment would also have been an option if the thesis would have investigated the desirability of the concept instead of focusing on the technical aspects. However, a qualitative approach was not viable due to the time constraints set to the thesis and a lack of contacts in the data marketplace industry.

Table 6: Evaluation methods used with instantiation (Peffers, Rothenberger, Tuunanen, and Vaezi 2012).

Evaluation Method	**Description**
Technical experiment	Evaluate performance of an implementation (optionally using synthetic or real-world data) to evaluate the technical performance of the system.
Prototype	Create an implementation to demonstrate the utility or sustainability of the artifact.
Subject-Based Experiment	Evaluate theory using test subjects
Illustrative scenario	Apply artifact to a synthetic or real-world situation to illustrate suitability or utility of the artifact.

In addition to subjectively evaluating whether the metadata model was sufficiently able to represent provenance of hybrid data products in the prototype data pipeline, the approach of using benchmarks was chosen to provide quantitative metrics on the impact of the metadata model. This approach is supported by earlier research where PROV artifacts have been evaluated by benchmarking the system impact (De Nies, Taxidou, Dimou, Verborgh, Fischer, Mannens, and Van de Walle 2015; Mohamed Jehad Baeth et al. 2017; M. J. Baeth and Aktas 2017; Tas et al. 2016). A test suite for measuring standalone provenance systems provided by Tas et al. (2016) was adapted to fit the purposes of this thesis. The original suite is specifically aimed to provenance databases, but it can be made suitable for evaluation streaming pipelines with slight modifications. Tas et al. (2016) introduces three suites: latency, simultaneous client connections, and message rate. In addition, we introduce a fourth metric, **event volume** to better measure the effect of the metadata model itself. Although insignificant at a smaller scale, the volume of the handled data affects the efficacy of systems at scale of big data. Larger data volumes lead to higher requirements for disk space and network throughput, making the metric relevant (Kleppmann 2017). Each of the tests measure performance from different perspectives:

Latency Measure the time it takes for system to produce an output from the input. Latency tests responsiveness of the system, with low latency being a general requirement of real-time systems (Kleppmann 2017). In the context of this thesis, latency is tested by comparing latencies when metadata model is enabled in comparison to when it's disabled.

Simultaneous pipelines Adapted from "Simultaneous client connections", we instead evaluate how many pending pipelines the system can handle with the given system resources.

Message rates Test how the rate of incoming messages affects the latency and stability of the system. In the thesis, we test this by sending artificial messages through the pipeline. To remove bottlenecks from querying external systems, we mock the behavior to test the raw processing capability with the given system resources.

Event volume Test how the metadata model affects the volume of data contained in a single event. This is measured by comparing data payload sizes with and without metadata.

5.2 Development of metadata model

During design and development phase of DSRM, the requirements and objectives created in the section 5.1 are used to develop an artifact. The reproducible steps to recreate it are documented in a manner that fits the type of the artifact. In this thesis, we form our own data model based on PROV by mapping concepts of data marketplaces to PROV concepts, and by applying the data contract metadata model of H.-L. Truong et al. (2012) in a new context.

In chapter 4, we investigated both the Open Provenance Model that provides a semantic base for building provenance models, and PROV, implementing Open Provenance Model with a concrete data modal and syntax. These solutions provide a base for building domain specific provenance models, which makes creation of an entirely new provenance metadata model redundant. We follow the same approach of mapping hybrid data product interactions to PROV as Taxidou et al. 2015 did in the domain of information diffusion.

We propose an extension to PROV in categories of *entities*, *actions*, *agents*, and *attributes* associated to them. Beginning with entities, or the objects of interest on data marketplace, we recognize the following taxonomy of data products:

- **`DataProduct`**: An abstract base class for data products that includes attributes common to all data products. *Extended* by `OriginDataProduct` and `HybridDataProduct`.
- **`OriginDataProduct`**: Data product from a specific origin that includes no earlier provenance information. Can be *derived* into hybrid data products when *used* by an action. `OriginDataProduct` is indicated by assigning entity's `prov:type` attribute to `datamarket:OriginDataProduct`.
- **`HybridDataProduct`**: A composite data product that is derived from multiple data products, such as `OriginDataProduct` or `HybridDataProduct`. A `HybridDataProduct` is *generated by* an action that *uses* data products. Similar to `OriginDataProduct`, a `HybridDataProduct` is defined by setting `prov:type` attribute to `datamarket:HybridDataProduct`.

Metadata relevant to data marketplaces can be embedded into these entities through attributes. As there already exists metadata models that have identified relevant attributes to data marketplaces (Vu et al. 2012; H.-L. Truong et al. 2012), one of these models can be applied to the provenance metadata model without conducting redundant research. In this thesis, a data contract metadata model by H.-L. Truong et al. (2012) was chosen to represent data marketplace metadata. The model by H.-L. Truong et al. (2012) was discussed more in detail in section 2.5. As data contract metadata terms were identified as simple key/value pairs, the terms could be directly mapped into the attributes of PROV. The resulting attribution mapping can be seen in Table 7.

Table 7: Metadata model namespaces and attributes including data contract terms mapped to PROV

Category	Attribute	Value(s)
Naming & Identification	dataproduct:<name>	N/A
	dataprovider:<name>	
PROV Types	datamarket:OriginDataProduct	N/A
	datamarket:HybridDataProduct	
	datamarket:DataProvider	
	datamarket:createHybridDataProduct	
Data rights	rights:derivation	{Undefined, Null, Allowed, Required, True, False}
	rights:collection	
	rights:reproduction	
	rights:attribution	
	rights:nonCommercialUse	
Data quality	dataquality:accuracy	Number $\in [0,1]$
	dataquality:completeness	
	dataquality:timeliness	
Regulatory Compliance	compliance:PrivacyCompliance	String
	compliance:<compliance>	
Pricing model	pricing:cost	Number $\in \mathbb{R}$
	pricing:currency	String
	pricing:model	String
	pricing:usageTime	Datetime
	pricing:maximumUse	Datetime
Control & relationship	control:Liability	String
	control:LawandJurisdiction	
	control:<control>	

As the focus of the thesis is in the formation of hybrid data products, we are only interested in the action of *creating a hybrid data product*. This is codified in a generic manner as `createHybridDataProduct`. In addition, the origin to which the data product is *attributed to* is recognized as an agent `DataProvider`. The more specific interactions between entities, agents and actions in the metadata model are documented in the Figure 10. Specific attributes are not defined for `DataProvider`, because discovering relevant metadata related to providers was deemed out of scope for this thesis.

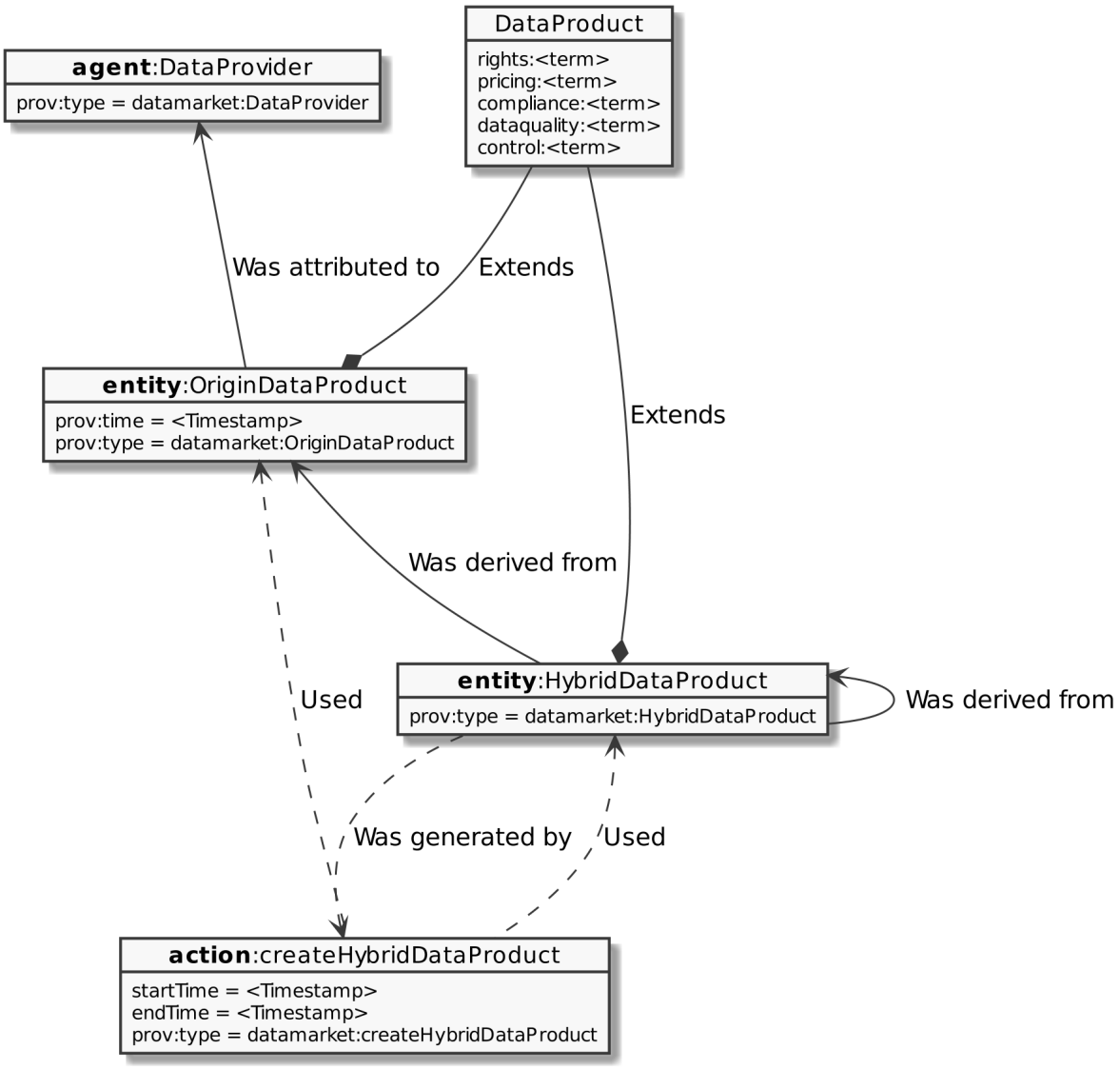


Figure 10: Extension of PROV for tracking provenance on data marketplace.

With all the elements of the metadata model described, we demonstrate how the PROV metadata model can be used in conjunction with PROV syntax in Table Figure 11. As a reference, the details of PROV syntax were earlier specified in Table 4. For brevity, the resulting metadata is included in Appendix section A.

```
agent(dataprovider:UniversityOfJyvaskyla,
      [prov:type='datamarket:DataProvider'])
entity(dataproduct:exampleData1,
       [prov:type='datamarket:OriginDataProduct',
        prov:time=now(),
        pricing:model='perUse',
        pricing:cost='0.001',
        pricing:currency='EUR',
        control:LawandJurisdiction='FI'])
entity(dataproduct:exampleData2,
       [prov:type='datamarket:OriginDataProduct',
        prov:time=now(),
        pricing:model='perUse',
        pricing:cost='0.001',
        pricing:currency='EUR',
        control:LawandJurisdiction='FI'])
wasAttributedTo(dataproduct:exampleData1,
                dataprovider:UniversityOfJyvaskyla)
wasAttributedTo(dataproduct:exampleData2,
                dataprovider:UniversityOfJyvaskyla)
activity(dataproduct:combineExampleData, start_time, end_time,
         [prov:type='createHybridDataProduct'])
used(dataproduct:combineExampleData,
     dataproduct:exampleData1, start_time)
used(dataproduct:combineExampleData,
     dataproduct:exampleData2, start_time)
entity(dataproduct:exampleHybridData,
       [pricing:model='perUse',
        pricing:cost='0.002',
        pricing:currency='EUR',
        prov:type='datamarket:HybridDataProduct']);
wasGeneratedBy(dataproduct:exampleHybridData,
               dataproduct:combineExampleData, now())
wasDerivedFrom(dataproduct:exampleHybridData,
               dataproduct:exampleData1)
wasDerivedFrom(dataproduct:exampleHybridData,
               dataproduct:exampleData2)
```

Figure 11: Example of PROV syntax that is used to define two data products (`exampleData1`,`exampleData2`) attributed to University Of Jyvä skylä that are combined to hybrid data product `exampleHybridData` through `combineExampleData` action.

5.3 Demonstration

In the demonstration step of DSRM, we apply the designed artifact to a prototype scenario in order to understand how it would work in practice and what is its capability for solving a problem. In this thesis, we apply the provenance metadata

model to a prototype data marketplace pipeline that models a specific scenario. The section is organized to three parts: **1.** illustrating a prototype data application that uses the metadata model, **2.** introducing available data, chosen data sets and the scenario, and **3.** implementing the scenario with the prototype application.

5.3.1 Prototype architecture

In order to test the metadata model, a prototype data pipeline was created to simulate transactions of a data marketplace supported by concepts of DaaS in chapter 3. Due to the scope of the thesis, the prototype was not designed for scale and applies only to four of the eight requirements set by Stonebraker et al. (2005) to stream processing systems:

Keep the Data Moving The prototype should be able to process events with minimal latency, and "on the fly" as the events arrive. The prototype should not have bottlenecks in the processing path that lead to notable delays.

Handle stream imperfections The prototype must be able to handle situations where the data might arrive late, out of order, or with some data missing entirely. This requirements also applies to intermediary steps of the processing pipeline, where data is enriched from external sources. In case that the data is notably delayed, each step of the processing pipeline should have a time out period to prevent standstills.

Integrate stored and streaming data The prototype must be capable of combining stored and stream data together. As multiple data sources are used in the prototype, it's expected that some of them are most likely stored at some external location. However, interfacing with external data sources can add latency to the pipeline, which is deemed acceptable in the context of this thesis.

Process and respond instantaneously The processing system must be able to process high volumes of data with very low latency, typically in the range of microseconds to milliseconds on commodity off-the-shelf hardware. This requires that all the system components are designed with performance in mind to prevent bottlenecks.

In the course of writing this thesis, multiple different approaches to implementing the prototype were considered. In the beginning, industrial stream processing frameworks, such as Apache Kafka and Apache Flink were considered as implementation technologies. However, these technologies are not very suitable for prototying purposes as creating implementations with them require time and expertise. The second approach considered during the writing was Apache Nifi as it provides a graphical user interface for composing stream processing pipelines in a manner that enables fast prototyping. The critical drawback of Apache Nifi was that it lacks documentation and instructions for more complicated use cases. Although creating simple prototypes with Nifi is easy, the system becomes exceedingly complicated if the user wishes to use more complex features. Lastly, the option of programming a custom solution was chosen as the most viable alternative.

The custom solution is created using NodeJS. NodeJS is a programming language that has a relatively simple, but powerful core abstractions, making it

a viable choice for fast prototyping. The prototype application is built around a pipeline that is formed from an asynchronous chain of operations that can act parallel to other running data pipeline instances. When data is requested from an external source, a data provider for that type of data is randomly chosen from a list of available providers, and queried for the information requested by the pipeline. The fetched data is associated with metadata of the data provider. The data pipeline architecture is similar to the general approach with stream processing systems, where a set of tasks is continuously executed on each incoming event (Vijayakumar and Plale 2006).

The architecture of the prototype is kept simple by using functional programming principles. Instead of creating and composing objects according to principles of object oriented programming, we merely focus on composing functions in the data pipeline. The general architectural of the data pipeline can be seen in Figure 12.

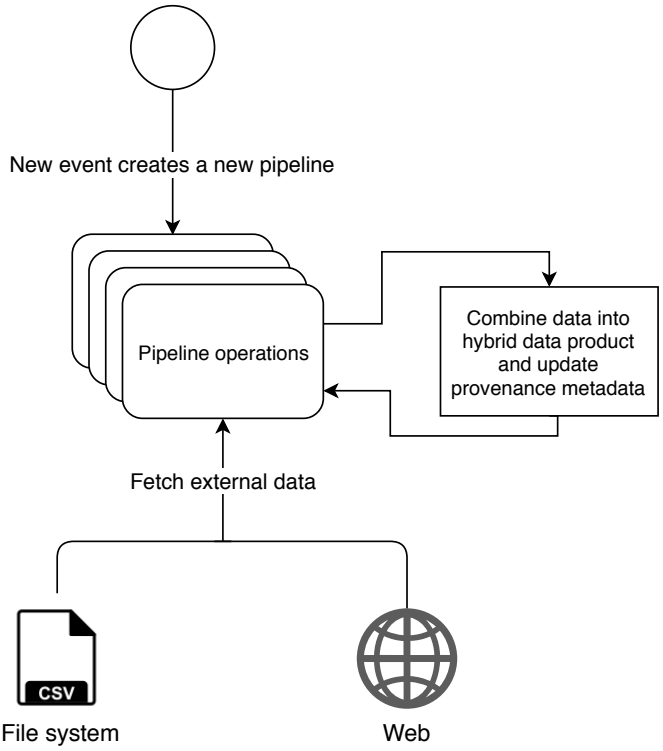


Figure 12: Architecture of the prototype data pipeline

5.3.2 Data sources and the scenario

In the course of the thesis, two major sources of data were considered as the base of the prototype: Transport for London and Transport for Luxembourg. The reason behind this decision was that **a)** both data providers offer open APIs that can freely be used by developers and researchers alike, and **b)** both London and Luxembourg have a comprehensive open data program that allows combining data sets in order to fulfill the requirement of composing a hybrid data product.

The APIs offered by Transport for Luxembourg (Duton and Degeling 2018) and Transport for London (T. f. London 2017) are compared in the Table 8.

Table 8: Comparison of open APIs between transport authorities of London and Luxembourg (Duton and Degeling 2018; T. f. London 2017)

Data type	**Description**	**London**	**Luxembourg**
Air quality	Level of pollutants and weather data	✓	✓
Bike points	Data from rental bike docks	✓	✓
Car parks	Data and occupancy of parking slots	✓	✓
Bus data	Data on bus departures and lines	✓	✓
Airport data	Departing and arriving flights	✗	✓
Highway data	Data on roads and congestion	✓	✓
Place data	Data on arbitary locations within city	✓	✗
Taxi data	Data on available taxis within area	✓	✗

London offers a repository of open data at `https://data.london.gov.uk/` and Luxembourg provides their repository at `https://data.public.lu/en/datasets/`. Both of the data repositories have highly diverse data sets in varying formats that range from typical spreadsheet files to geospatial data sets that associate data with geographical locations. Although both of the data portals provide a comparable amount of data sets, with London portal featuring 740 datasets and Luxembourg having 669 data sets, the accessibility of Luxembourg data is constrained due to the website and data being provided in an arbitary mix of English, French and German.

In the end, a decision to use Luxembourg data was made on the basis that Transport for Luxembourg provides open and freely accessible stream data APIs. Although Transport for London also features a stream data API, it's restricted to paying customers and specific use cases only. Choosing to use data available from London would have required simulating a stream data environment.

Transport for Luxembourg provides five stream data APIs that are listed in the Table 9 (Duton et al. 2018). However, only the bus departure API seems to be truly real time while other endpoints merely provide hourly updates. Due to this, the departure API was considered the main candidate for the illustrative scenario.

Table 9: Stream data APIs provided by Transport for Luxembourg

Data type	**Description**	**Update frequency**
Weather	Level of pollutants and weather data	Hourly
Bike points	Data from rental bike docks	Hourly
Car parks	Data and occupancy of parking slots	Hourly
Departures	Data on bus and train departures	Real-time
Highway data	Data on roads and congestion	Hourly

A data payload provided by departure API can be seen in Figure 13. The structure is based on the REST specification documented by Duton et al. (2018), where the `"type"` key indicates whether new data was created, old data was updated, or if some data was deleted from the resource. In the experiment, we only use rows with the `"new"` type that indicate new departures. The payload under `"data"` key is categorized according to stations. In the example from Figure 13, one new departure was recorded from station with the ID of `220902004`.

```
{
  "type": "new",
  "data": {
    "220902004": [
      {
        "id": "1|1533|8|82|16032017",
        "type": "bus",
        "trainId": null,
        "lineId": "4:TIC---:13",
        "line": "13",
        "number": 180,
        "departure": 1489658400,
        "delay": 0,
        "live": true,
        "departureISO": "2017-03-16T11:00:00+01:00",
        "destination": "Esch/Alzette, Gare",
        "destinationId": 220402034
      },
      ...
      ]
  }
}
```

Figure 13: JSON data payload from a Transport for Luxembourg departure event

As the data is somewhat incomplete, referring to entities by their ID, but not by their content, we can start building an imaginary data marketplace scenario that is used to provide a more complete picture of each departure to the consumers of the data.

To demonstrate a scenario built onto this data, let us first assume that the following organizations would exist as seperate entities and would interact with each other on a data marketplace:

- **Autonomous transport provider** Autonomous transport provider manages a network of vehicles that have no pre-scheduled lines or timetables. Instead, users of the vehicles can request for a pick-up, and a vehicle is routed to the destination autonomously, forming an 'ad hoc' transportation line where other passengers heading to the same location can also be picked on the way. In order to have up-to-date data on traffic conditions every time a vehicle departs, the autonomous transport provider joins a data marketplace to create a data pipeline with data sources provided by station authority

and distance matrix providers. This entity is simulated by the the stream departure API provided by Transport for Luxembourg (Duton et al. 2018).

- **Station authority** A station authority manages stations and stops that are used by the autonomous transport provider. Station authority provides a REST API that can be used to query details regarding each station. Most importantly for autonomous vehicles, the API provides exact coordinates of the stop. If station authority is not reachable, a local backup is used. Similar to transport authority, the REST API of Transport for Luxembourg is used to simulate this actor. (Duton et al. 2018)
- **Distance Matrix provider(s)** Distance matrix calculation is used for optimizing logistics and finding optimal routes between two or more locations. Due to the scope of the thesis, we limit this feature only to *distance* and *duration* properties between two locations that can be used to provide an estimated time of arrival to the users of autonomous transport provider. At the time of writing this thesis, free or easily accessible trials of Distance Matrix APIs were offered by Google (2018), HERE (2018), OpenRouteService (2017), and MapQuest (2018).

In the scenario, the autonomous transport provider creates a data pipeline that forms hybrid data products from the station authority. The steps of the pipeline can be seen in Figure 14. To simulate randomness, the use of different sources is randomized to varying degrees. Retrieving data from station authority will fail and fall over to a local backup 15% of the time, but the distance matrix provider is chosen randomly without favoring any specific provider.

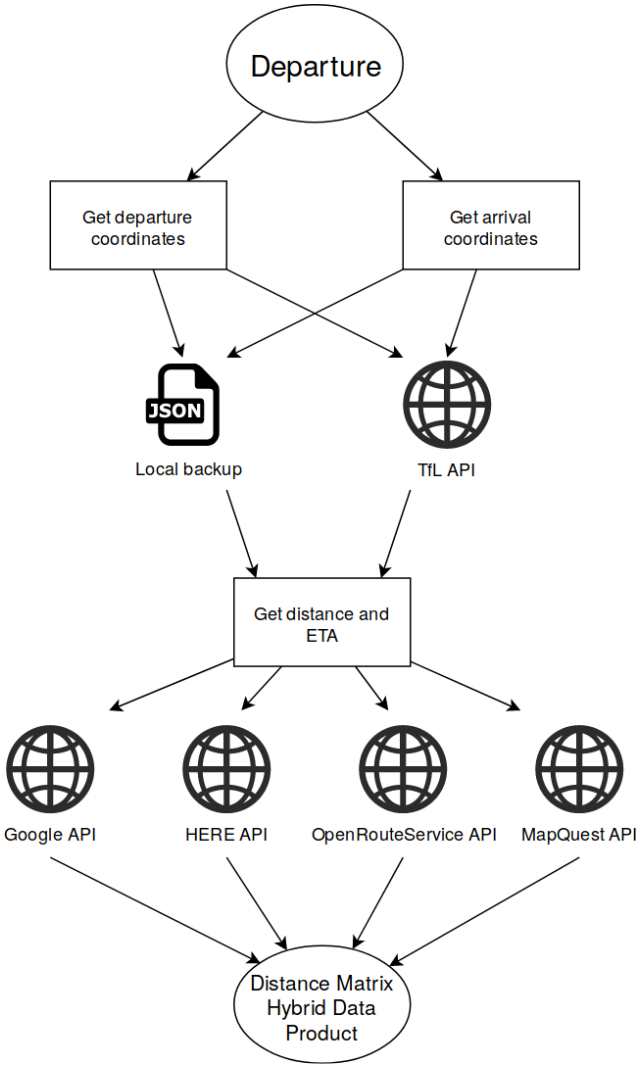


Figure 14: Steps of enrichment in the prototype

The resulting hybrid data product and it's record of provenance is valuable to the autonomous transport authority. If the estimated time of arrival or the station coordinates are incorrect, the reason why can be deducted later from the provenance record. It might be that one of the distance matrix provider is less accurate than the others. Alternatively, the autonomous vehicle might've gotten lost, which could be explained if the departure used a stale local copy of the station data in a situation where the station had been temporarily closed due to maintenance.

5.3.3 Executing the prototype

The prototype was programmed according to the architecture described in subsection 5.3.1 and the scenario found from subsection 5.3.2. The prototype data pipeline was designed to run from command line and features no user inteface.

Every time a departure event happens, a new pipeline is spawned that will retrieve data from data providers, combine it to a hybrid data product, and update

the metadata on the fly. The resulting data product is written to filesystem for inspection. A screenshot of the program can be seen in Figure 15.

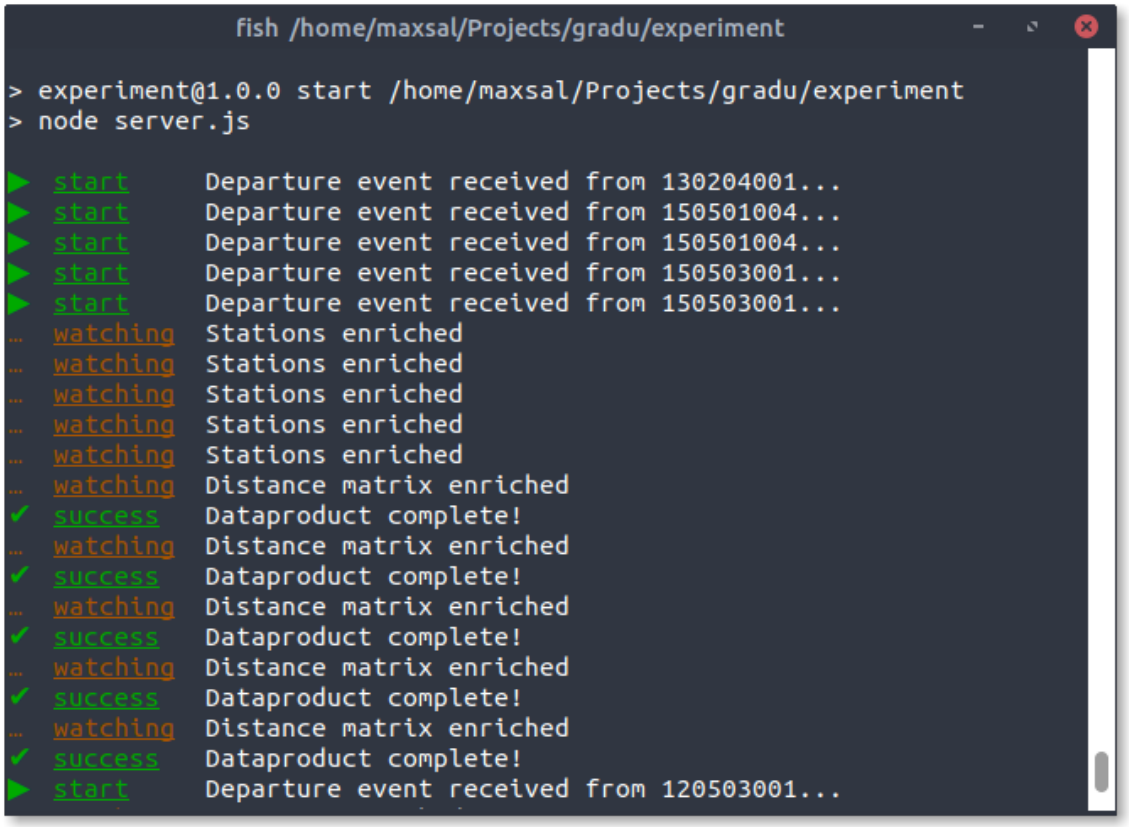


Figure 15: Steps of enrichment in the prototype

In order to properly evaluate the performance impact of the metadata model, some features of the prototype had to be toggleable in order to control for external factors:

- **Metadata model:** To compare performance of the prototype with and without metadata model, the metadata model had to be toggleable.
- **Network requests:** Network requests are costly and introduce randomness to the prototype. In order to control for the effect of network request latency, a toggle was created that disables network requests and provides mock data instead.

5.4 Evaluation

In evaluation phase of DSRM, the performance and applicability of the artifact is evaluated using metrics, analysis, and knowledge gained from the demonstration (section 5.3). In this thesis, we use quantitative metrics gained from the prototype to evaluate how the metadata model affected performance in the prototype scenario.

5.4.1 Evaluation environment

The benchmarks were conducted on a quad-core X86 CPU (AMD 2200g) with 16GB of DDR4 memory. The system was running Ubuntu 18.04 bionic on Linux kernel 4.15.0-34 with the NodeJS environment at version 8.10.0. The testing library used during evaluation was Benchmark.js (Tan, Dalton, Cambridge, and Bynens 2018) that can be used to gain statistically significant results.

To provide more complete results, metrics were gathered both with and without network calls enabled. When network calls were enabled, the real-world scenario based on the prototype application described in Figure 14 was fully emulated. When network calls were disabled, the same steps were followed as in the scenario, but all network calls provided mock data. Network calls were controlled in order to gain accurate data on the performance impact of the metadata model itself. Enabling network calls, on the other hand, provided perspective on whether the impact of metadata model was meaningful in practice.

For tests where network requests were disabled, Benchmark.js was configured to a minimum sample size of 1000. In order to prevent throttling problems in the real-world tests, the minimum sample size in those tests was restricted to 100.

5.4.2 Measurement: Output data volume

Output data volume was measured by forming a JSON record and investigating its total file size. As seen from Figure 16, provenance metadata increased the total volume roughly $\times 5.4$ in the prototype scenario. The total volume will, however, depend both on the complexity of provenance record and the size of actual data.

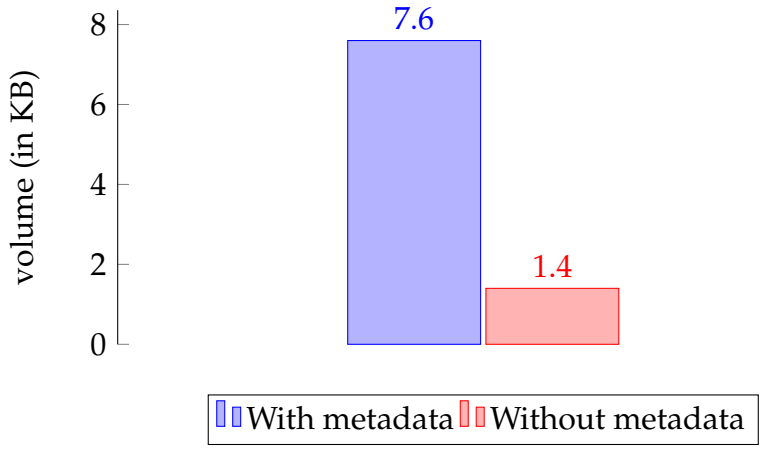


Figure 16: Volume of the data payload, including metadata and actual data

5.4.3 Measurement: Latency

Latency measure the time it takes for system to produce an output from the input. The metric is used to measure the impact of the metadata model on the "length" of the pipeline in terms of time.

The test results seen in Figure 17 indicate that including provenance metadata into the pipeline relatively increases the latency by a significant degree. The median latency of 125μs with metadata is roughly $\times$7.8 (109 μs) larger than 16μs it takes without metadata.

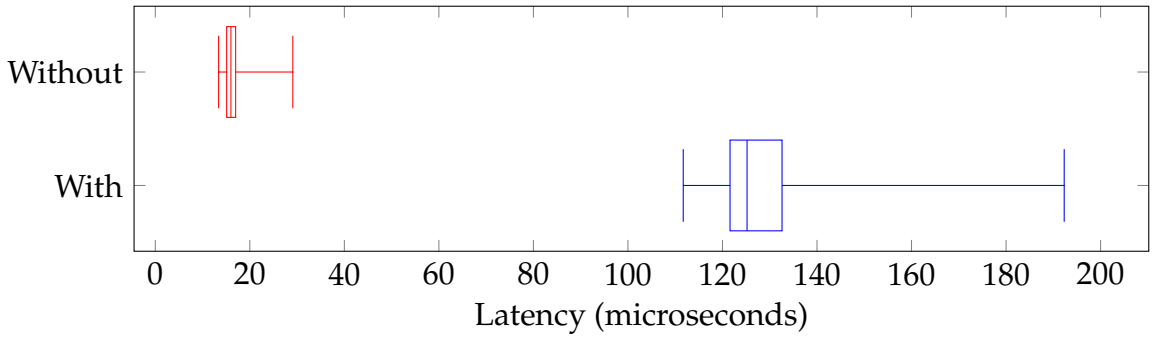


Figure 17: Latency with network requests disabled

To investigate how the increase in latency affects the scenario in practice, we ran the same benchmark again with network requests enabled. At first, a measure at minimum sample size of 1000 was attempted. However, this caused Transport for Luxembourg API to refuse service due to the prototype exceeding a request quota. This lead to the decision to restrict minimum sample size to 100 for real-world tests.

As seen from Figure 18, the latency increase of roughly 109μs witnessed in Figure 17 has little impact on the latency of the pipeline with network requests enabled: The total latency of the pipeline is multiple orders of magnitude higher than the impact caused by the metadata model. The difference between the two bars in Figure 18 are explained by the significant element of randomness introduced by the network calls.

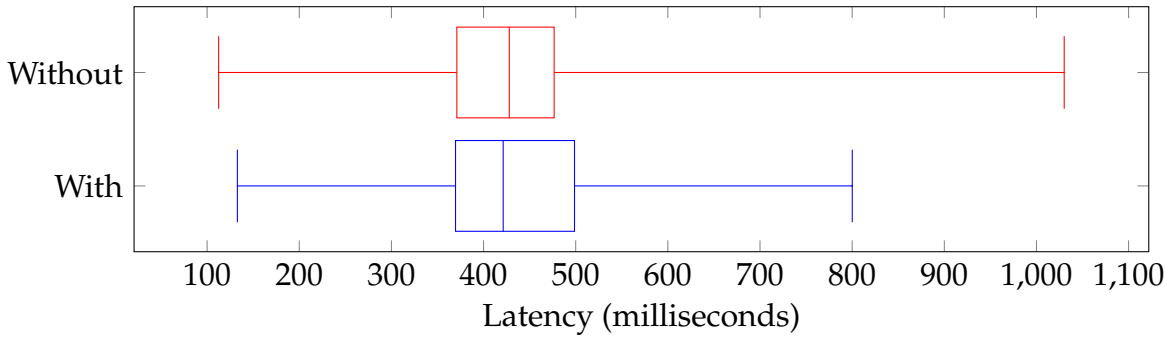


Figure 18: Latency with network requests enabled

5.4.4 Measurement: Simultaneous pipelines

To test the limit of simultaneous parallel pipelines, a benchmark was created where network requests were disabled, and an infinitely blocking operation was added at the end of the pipeline. After this, artificial departure events were spawned in a loop with an incrementing counter to quickly pile up simultaneous pipelines.

It was discovered that NodeJS virtual machine halts due to a `heap out of memory` error once the memory usage of the virtual machine reaches 1.76GB. This is a design choice made in the V8 javascript virtual machine used by NodeJS in order to prevent memory leaks and garbage collection slowdowns (Degenbaev, Payer, Lippautz, and Kozyatinskiy 2017). Although it might be possible to bypass the limit, we will comply with the recommended standard heap limit.

Figure 19 displays the resulting limits to simultaneous pipelines using the standard V8 memory limits. When metadata was enabled, the benchmark was able to handle roughly half the number of simultaneous pipelines. It should be noted, however, that the limit is correlated to memory usage of the metadata and actual data combined, and will most likely vary between different scenarios. The difference of results between Figure 16 and Figure 19 is explained with different internal and external representations of the PROV metadata: When processed inside the application, the data structure of metadata is kept in a format that requires less space.

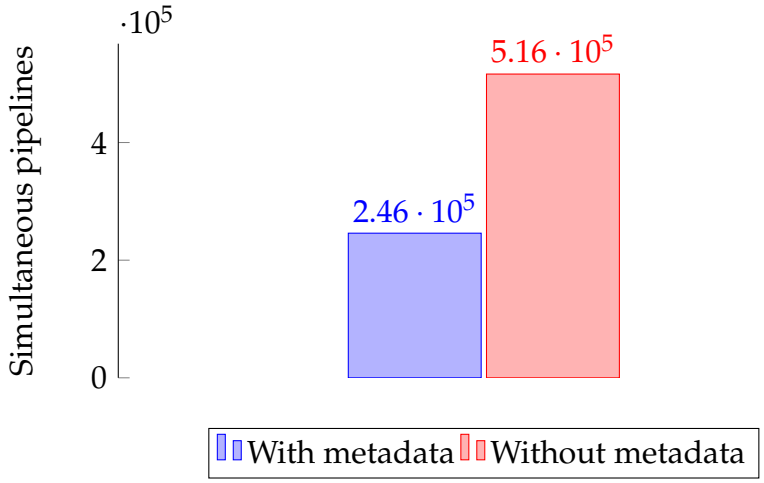


Figure 19: Limit of simultaneous pipelines in the benchmark

5.4.5 Measurement: Message rates

Message rates, or throughput of the system, was measured by investigating the amount of rows that the prototype is able to process in a second. Similar to earlier metrics, the throughput is first tested with network requests disabled before testing the throughput in the real-world environment by enabling network requests.

As the application is not multithreaded, the results in Figure 20 are in line with results from the latency test and Figure 17. When network requests were disabled, the metadata model reduces the message rate of prototype application by a significant degree with an approximate message rate of $\times 0.16$ in comparison to when metadata model is disabled.

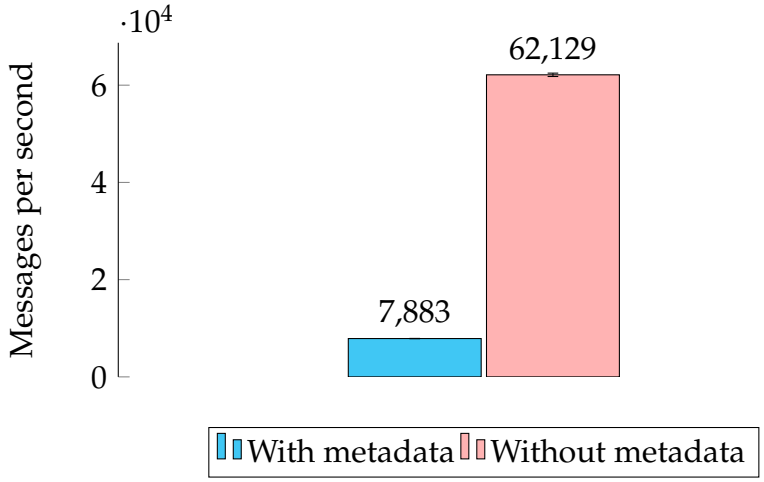


Figure 20: Message rate with network requests disabled. Relative margin of error (With: ±0.41%) (Without: ±0.56%)

To prevent exceeding a usage threshold with external service providers similar to the problems encountered in latency tests, parallellism of pipelines had to be disabled when running the benchmark with network requests enabled. Although this caused the throughput to be highly constrained in comparison to the theoretical maximum of the prototype, the resulting data can still be used as a representative metric of throughput.

As with the latency test and Figure 18 the results of the benchmark with network requests enabled at Figure 21 show little to no difference whether metadata model is enabled or disabled. The variation between the throughput when metadata model was enabled and when it was disabled can be explained with the high margin of error.

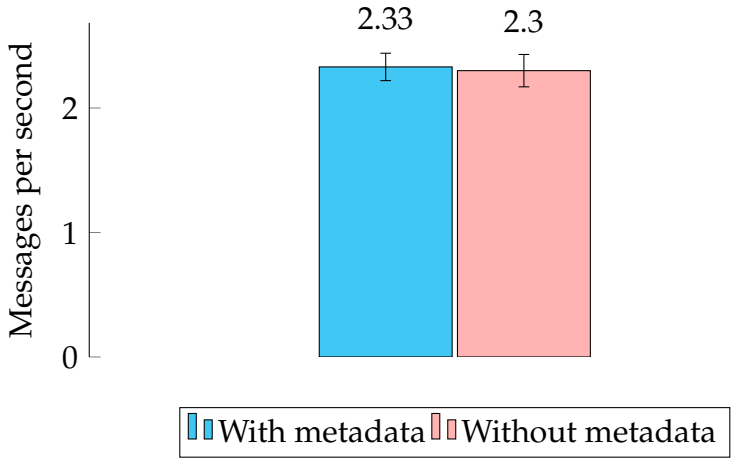


Figure 21: Message rate with network requests enabled, Relative margin of error (With: ±4.8%), (Without: ±5.71%)

5.5 Conclusion

The metadata model was able to fulfill the functional requirements set to it: It's both able to represent the chain of provenance of hybrid data products with the syntactical framework provided by PROV, and able to contain data relevant to marketplaces in the form of data contract terms formulated by H.-L. Truong et al. (2012), fulfilling requirements **1.** and **2.** described in section 5.1. Provenance events that occured in the formation of hybrid data products are encoded to metadata alongside with marketplace metadata, as seen in extracts at section B.

The metadata model was applied to a non-trivial prototype where a hybrid data product was created from multiple different data sources, demonstrating its applicability in a practical scenario that involves both static and stream data. In addition to providing JSON output, PROV also enables the metadata model to be presented in XML format. Benchmarks with network requests disabled in subsection 5.4.3, and subsection 5.4.5 display that handling and updating metadata model in the prototype has a performance impact. However, the latency of $109\mu s$ is well below the significant latency threshold of 10ms set in requirement **3.** of section 5.1. To support the chosen latency threshold, the benchmarks where network requests were enabled demonstrate that the performance cost caused by the metadata model is negligible in practice. The performance impact measured in hunders of microseconds is unlikely to affect performance of typical stream processing systems where latencies are measured in milliseconds (Kleppmann 2017; Kleppmann and Kreps 2015; Wang 2016).

6 Discussion

In this thesis, a metadata model for data marketplaces was proposed, enabling tracking of provenance and metadata of hybrid data products. The metadata model was implemented in a functional prototype to investigate its viability and to evaluate the performance impact of the metadata model through a set of benchmarks. The research was performed according to the steps of design science research method (DSRM) that also provided the general structure of the thesis. From the perspective of the author, DSRM was a fitting choice as it helped narrowing the scope of the thesis to a problem area that was still feasible to evaluate. The resulting metadata model was successfully able to fulfill its requirements without causing a considerable performance impact.

At the time of writing this thesis, research on data marketplaces was still scarce, but sufficient to provide background. Academic sources were prioritized in the writing of the thesis, but for topics that discussed technical and implementation details, non-academic sources were more relevant and available. Even when academic sources were not used, we attempted to use literature sources that were highly regarded by the software developer community.

The scope of the metadata model was restricted to only cover hybrid data products and the data sources they originate from. Additionally, the definition of data marketplace metadata was restricted to data contract terms of H.-L. Truong et al. (2012). It is likely that the model can be both extended with new concepts in the future, and made more complete by adding additional metadata attributes. Doing so, however, would require additional insight to data marketplaces and their needs. This thesis was constrained by the lack of literature, but future research might enable formulation of more comprehensive metadata models.

The PROV model provided a suitable base for building the metadata model, and it seems to also have been used in non-academic contexts (K. C. London 2018). In the development of the prototype, however, it felt that some implementations of PROV have received more attention than others. Although the JavaScript implementation of PROV, ProvJS, was advertised in (K. C. London 2018), it's not mentioned in W3C specification (Gil et al. 2013). Additionally, ProvJS did not have documentation on its usage and only provides example code snippets in its repository. Based on this, it might be preferable to develop future PROV solutions using implementations made for other languages.

To measure the performance of metadata model, a benchmark for provenance systems by (Tas et al. 2016) was adapted for the purposes of this thesis; a benchmark that has also been used for other provenance systems. In addition to the three metrics provided the benchmark, including latency, message rate and simultaneous connections (pipelines), a fourth metric of filesize was included to represent the volume of metadata model. The benchmarks were conducted using consumer hardware, on a single machine, and without multi-threading. Although the benchmarking environment was not similar to cloud computation environment that favors multi-threading and distributed computation (Koutroumpis

et al. 2017), we believe the metrics to be sufficiently valid in the context of this thesis. The relative performance impact of the metadata model should should not vary significantly between environments because the PROV-JS library used in the prototype is not parallellized and gains no performance benefits in server environments.

The thesis and the prototype source code is hosted on gitlab at `https://gitlab.com/duics/gradu` where the results of the benchmarks can be independently verified for as long the APIs that the prototype relies on are still available. However, unless the benchmark environment is identical, it's likely that the results are nonidentical. Regardless, relative performance differences between when metadata is enabled and when metadata is disabled should still remain approximately same.

The performance of the metadata model relied highly on the NodeJS runtime and ProvJS library provided by K. C. London (2018). Although the tests where network requests were enabled in section 5.4 indicated that the performance impact of the metadata model was negligible in that specific scenario, there might be use cases where sub-millisecond latencies are required. it might be possible that PROV libraries designed for other programming languages are more performant. Additionally, it might be possible to improve performance of the metadata model by using binary encoded formats, such as Protocol Buffers or Apache Trift (Kleppmann 2017). Both of these encodings trade flexibility for higher performance, leading to additional design constraints in the metadata model. In addition to potentially allowing the metadata model to be used in scenarios where sub-millisecond latencies are a requirement, such approaches might also be useful in systems that encounter very high traffic where even a minor performance increases can lead to notable savings on processor time.

At an earlier time at the writing of the thesis, an alternative perspective of investigating architecture of a data marketplace was also considered. Currently there exists little research on how one might build a data marketplace or a Data as a Service (DaaS) platform while taking into notion the special requirements set by those specific services. Stream processing and event-based architectures have recently become more commonplace in the industry due to their capability of handling real-time data with simple abstractions (Kleppmann 2017), leading to an abundance of technologies that can be used to facilitate the technical architecture of a stream data marketplace. Although the challenges of scalability and adaptability of data marketplace architecture are interesting, investigating such a broad topic would not have been viable in the limited context of a master's thesis. The idea could, however, be applicable in other contexts.

The topic of data marketplaces in general has many potential branching directions for future research. As the concept of a metadata is ubiquitous on data marketplaces, there is potential in developing the metadata model alongside the new research topics. Koutroumpis et al. (2017) suggests two major future research topics for data marketplaces: Creation of *data contract management* systems for managing and enforcing contractual terms and conditions for data transactions and *distributed data marketplaces* that require no central authority.

Koutroumpis et al. (2017) suggests that a data contract management system would require a *data contract clearance service* to generate contractual terms to

hybrid data products. The contractual terms of the hybrid data product would be generated from the derived sources based on the contractual logic of each data source. The idea was experimented with in the prototype created during the empiric part of the thesis: A hybrid data product derived from multiple sources was 'priced' according to the cost of each source used in the generation of hybrid data product. However, applying similar logic to other data contract terms would require investigating and applying contractual law: A topic that was out of scope in this thesis. Future research could include investigating how contractual terms could be inherited by hybrid data products.

Research on distributed data marketplaces is closely related to blockchains and distributed ledgers (Koutroumpis et al. 2017). Distributed marketplaces might require adjustments to the metadata model described in this thesis, or possibly an entirely different approach to metadata. Depending on how a distributed marketplace is built, provenance information could even be deferred from blockchain transactions themselves. As distributed marketplaces have no central authority that manages the data marketplace, transactions and metadata related to them has to be shared with all participants of the data marketplace. This would allow each participant to verify trustworthiness of other actors. However, the transparency of blockchain also has privacy and security concerns as none of the transactions on the blockchain are anonymous. A decentralized marketplace would, in essence, just provide the communication structures that facilitate operations of a decentralized market (Koutroumpis et al. 2017). Another possible alternative to distributed data marketplaces could be found in secure Multi-party Computation (MPC) architecture. In a secure MPC architecture, the trusted middleman is "emulated" through cryptographic interactions (Goldreich 1998), but there exists no earlier research on how the concept could be applied to data marketplaces.

In this thesis data marketplaces were discussed as a platform to facilitate interactions between organizations that seek to trade information with each other. However, Kortuem and Kawsar (2010) suggest that, with the growth of Internet of Things, even consumers might be incentivized to take part in the data markets. There exists some research that points towards the willingness of consumers to share data from IoT devices. Research by Foster (2009) and Foster, Blythe, Cairns, and Lawson (2010) indicates that consumers are willing to share data on their electricity usage. Additionally, the findings of Grossklags, Hall, and Acquisti (2007) point towards consumers having a high preference of giving up private information for monetary gain. In future, this could enable data marketplaces where consumers could trade their private data as a commodity. Companies could then use that data to provide services to the consumers, or use that data in exchange for monetary compensation to the user. To maintain metadata and provenance of data products in such data marketplaces, the metadata model would have to be extended with agents that represent the consumers while still remaining compliant with privacy regulations.

Common to all future research directions, it might be preferrable to develop a metadata model in cooperation with data marketplaces themselves. This would likely lead to an interesting metadata model, as it would be closely aligned with requirements and practices of the industry.

REFERENCES

Akidau, Tyler, Slava Chernyak, and Reuven Lax. 2018. *Streaming Systems.* ISBN: 978-1-4919-8387-4, visited on August 9, 2018. http://shop.oreilly.com/product/0636920073994.do.

Allen, Sean T., Matthew Jankowski, and Peter Pathirana. 2015. *Storm Applied: Strategies for Real-Time Event Processing*. OCLC: ocn907676623. Shelter Island, NY: Manning Publications Co. ISBN: 978-1-61729-189-0.

Arafati, M., G. G. Dagher, B. C. M. Fung, and P. C. K. Hung. 2014. "D-Mash: A Framework for Privacy-Preserving Data-as-a-Service Mashups". In *2014 IEEE 7th International Conference on Cloud Computing*, 498–505. Visited on February 20, 2018. doi:10.1109/CLOUD.2014.73. http://ieeexplore.ieee.org/document/6973779/.

Assunção, Marcos D., Rodrigo N. Calheiros, Silvia Bianchi, Marco A. S. Netto, and Rajkumar Buyya. 2015. "Big Data Computing and Clouds: Trends and Future Directions". *Journal of Parallel and Distributed Computing*, Special Issue on Scalable Systems for Big Data Management and Analytics, 79-80 (): 3–15. ISSN: 0743-7315, visited on February 20, 2018. doi:10.1016/j.jpdc.2014.08.003. http://www.sciencedirect.com/science/article/pii/S0743731514001452.

Baeth, M. J., and M. S. Aktas. 2017. "Detecting Misinformation in Social Networks Using Provenance Data". In *2017 13th International Conference on Semantics, Knowledge and Grids (SKG)*, 85–89. doi:10.1109/SKG.2017.00022.

Baeth, Mohamed Jehad, and Mehmet S Aktas. 2017. "A Large Scale Synthetic Social Provenance Database". In *Proceedings of the Ninth International Conference on Advances in Databases, Knowledge, and Data Applications*, 16–22.

Bradner, Scott. 1997. "Key Words for Use in RFCs to Indicate Requirement Levels". Visited on September 15, 2018. https://tools.ietf.org/html/rfc2119.

Carata, Lucian, Sherif Akoush, Nikilesh Balakrishnan, Thomas Bytheway, Ripduman Sohan, Margo Seltzer, and Andy Hopper. 2014. "A Primer on Provenance". *Queue* 12, number 3 (): 10:10–10:23. ISSN: 1542-7730, visited on September 8, 2018. doi:10.1145/2602649.2602651. http://doi.acm.org/10.1145/2602649.2602651.

Chen, Y., J. Kreulen, M. Campbell, and C. Abrams. 2011. "Analytics Ecosystem Transformation: A Force for Business Model Innovation". In *2011 Annual SRII Global Conference*, 11–20. doi:10.1109/SRII.2011.12.

De Nies, Tom, Io Taxidou, Anastasia Dimou, Ruben Verborgh, Peter M. Fischer, Erik Mannens, and Rik Van de Walle. 2015. "Towards Multi-Level Provenance Reconstruction of Information Diffusion on Social Media". In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 1823–1826. CIKM '15. New York, NY, USA: ACM. ISBN: 978-1-4503-3794-6, visited on September 11, 2018. doi:10.1145/2806416.2806642. http://doi.acm.org/10.1145/2806416.2806642.

Deelman, Ewa, Bruce Berriman, Ann Chervenak, Oscar Corcho, Paul Groth, and Luc Moreau. 2010. "Metadata and Provenance Management" (). Visited on September 8, 2018. arXiv: 1005.2643 [astro-ph]. http://arxiv.org/abs/1005.2643.

Degenbaev, Ulan, Hannes Payer, Michael Lippautz, and Alexey Kozyatinskiy. 2017. "One Small Step for Chrome, One Giant Heap for V8 · V8". Visited on September 25, 2018. https://v8.dev/blog/heap-size-limit.

Deutch, Daniel, Nave Frost, and Amir Gilad. 2018. "Provenance for Non-Experts". *IEEE Data Eng. Bull.* 41:3–14.

Dikaiakos, M. D., D. Katsaros, P. Mehra, G. Pallis, and A. Vakali. 2009. "Cloud Computing: Distributed Internet Computing for IT and Scientific Research". *IEEE Internet Computing* 13, number 5 (): 10–13. ISSN: 1089-7801, visited on February 20, 2018. doi:10.1109/MIC.2009.103. http://ieeexplore.ieee.org/document/5233607/.

Du, B., R. Huang, X. Chen, Z. Xie, Y. Liang, W. Lv, and J. Ma. 2016. "Active CTDaaS: A Data Service Framework Based on Transparent IoD in City Traffic". *IEEE Transactions on Computers* 65, number 12 (): 3524–3536. ISSN: 0018-9340, visited on February 20, 2018. doi:10.1109/TC.2016.2529623. http://ieeexplore.ieee.org/document/7406757/.

Duton, Daniel, and Thierry Degeling. 2018. "Streaming APIs · TfL". Visited on September 11, 2018. https://docs.api.tfl.lu/v1/en/Streaming_APIs.html.

Fischer, Joel E., Sarvapali D. Ramchurn, Michael A. Osborne, Oliver Parson, Trung Dong Huynh, Muddasser Alam, Nadia Pantidi, et al. 2013. "Recommending Energy Tariffs and Load Shifting Based on Smart Household Usage Profiling", 383–394. ISBN: 978-1-4503-1965-2, visited on September 11, 2018. https://eprints.soton.ac.uk/346991/.

Flannagan, Mike. 2016. "Maximize the Value of Your Perishable Data". Visited on October 5, 2017. http://data-informed.com/maximize-the-value-of-your-perishable-data/.

Foster, Derek. 2009. "Social Networking Sites as Platforms to Persuade Behaviour Change in Domestic Energy Consumption" (September): 116–116.

Foster, Derek, Mark Blythe, Paul Cairns, and Shaun Lawson. 2010. "Competitive Carbon Counting". *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems - CHI EA '10:* 4039–4039. ISSN: 9781605589305. doi:10.1145/1753846.1754099. http://www.scopus.com/inward/record.url?eid=2-s2.0-77953096794&partnerID=tZOtx3y1.

Friedman, Ellen, and Kostas Tzoumas. 2016. *Introduction to Apache Flink: Stream Processing for Real Time and Beyond.* 1 edition. Sebastopol, CA: O'Reilly Media. ISBN: 978-1-4919-7658-6.

Gandomi, Amir, and Murtaza Haider. 2015. "Beyond the Hype: Big Data Concepts, Methods, and Analytics". *International Journal of Information Management* 35, number 2 (): 137–144. ISSN: 0268-4012, visited on March 26, 2018. doi:10.1016/j.ijinfomgt.2014.10.007. http://www.sciencedirect.com/science/article/pii/S0268401214001066.

Gil, Yolanda, Simon Miles, Khalid Belhajjame, Helena Deus, Daniel Garijo, Paolo Missier, Stian Soiland-Reyes, and Stephan Zednik. 2013. "PROV Model Primer". Visited on September 8, 2018. https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/.

Goldreich, Oded. 1998. "Secure Multi-Party Computation" (): 109. Visited on August 9, 2018. https://www.researchgate.net/profile/Oded_Goldreich/publication/2934115_Secure_Multi-Party_Computation/links/00b7d52bb04f7027d4000000.pdf.

Google. 2018. "Get Started | Distance Matrix API". Visited on September 11, 2018. https://developers.google.com/maps/documentation/distance-matrix/start.

Grossklags, Jens, South Hall, and Alessandro Acquisti. 2007. "When 25 Cents Is Too Much : An Experiment on Willingness-To-Sell and Willingness-To-Protect Personal Information". *Information Security:* 7–8. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.137.696&rep=rep1&type=pdf.

HERE. 2018. "Calculate Matrix - Routing API". Visited on September 11, 2018. https://developer.here.com/documentation/routing/topics/resource-calculate-matrix.html.

Herschel, Melanie, Ralf Diestelkämper, and Houssem Ben Lahmar. 2017. "A Survey on Provenance: What for? What Form? What From?" *The VLDB Journal* 26, number 6 (): 881–906. ISSN: 0949-877X, visited on September 8, 2018. doi:10.1007/s00778-017-0486-1. https://doi.org/10.1007/s00778-017-0486-1.

Kleppmann, Martin. 2017. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems.* 1 edition. Cambridge: O'Reilly Media. ISBN: 978-1-4493-7332-0.

Kleppmann, Martin, and Jay Kreps. 2015. "Kafka, Samza and the Unix Philosophy of Distributed Data." *IEEE Data Eng. Bull.* 38 (4): 4–14. Visited on January 4, 2018. https://martin.kleppmann.com/papers/kafka-debull15.pdf.

Kortuem, G., and F. Kawsar. 2010. "Market-Based User Innovation in the Internet of Things". In *2010 Internet of Things (IOT)*, 1–8. doi:10.1109/IOT.2010.5678434.

Koutris, Paraschos, Prasang Upadhyaya, Magdalena Balazinska, Bill Howe, and Dan Suciu. 2015. "Query-Based Data Pricing". *Journal of the ACM* 62, number 5 (): 1–44. doi:10.1145/2770870. http://dl.acm.org/citation.cfm?doid=2841330.2770870.

Koutroumpis, Pantelis, Aija Leiponen, and Llewellyn D. W. Thomas. 2017. *The (Unfulfilled) Potential of Data Marketplaces,* ETLA Working Papers 53. The Research Institute of the Finnish Economy. Visited on June 19, 2018. https://ideas.repec.org/p/rif/wpaper/53.html.

Li, Guoliang, Beng Chin Ooi, Jianhua Feng, Jianyong Wang, and Lizhu Zhou. 2008. "EASE: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data". In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 903–914. SIGMOD '08. New York, NY, USA: ACM. ISBN: 978-1-60558-102-6, visited on March 23, 2018. doi:10.1145/1376616.1376706. http://doi.acm.org/10.1145/1376616.1376706.

Lim, Hyo-Sang, Yang-Sae Moon, and Elisa Bertino. 2009. "Research Issues in Data Provenance for Streaming Environments". In *Proceedings of the 2Nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, 58–62. SPRINGL '09. New York, NY, USA: ACM. ISBN: 978-1-60558-853-7, visited on August 18, 2018. doi:10.1145/1667502.1667516. http://doi.acm.org/10.1145/1667502.1667516.

London, King's College. 2018. "Provenance Web Services". Visited on September 24, 2018. https://openprovenance.org/.

London, Transport for. 2017. "Unified API". Visited on October 5, 2017. https://www.tfl.gov.uk/info-for/open-data-users/unified-api.

MapQuest. 2018. "Route Matrix API". Visited on September 11, 2018. https://developer.mapquest.com/documentation/directions-api/route-matrix/post/.

Moreau, Luc, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, et al. 2011. "The Open Provenance Model Core Specification (v1.1)". *Future Generation Computer Systems* 27, number 6 (): 743–756. ISSN: 0167-739X, visited on September 8, 2018. doi:10.1016/j.future.2010.07.005. http://www.sciencedirect.com/science/article/pii/S0167739X10001275.

Moreau, Luc, Juliana Freire, Joe Futrelle, Robert E. McGrath, Jim Myers, and Patrick Paulson. 2008. "The Open Provenance Model: An Overview". In *Provenance and Annotation of Data and Processes*, 5272:323–326. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-89964-8 978-3-540-89965-5, visited on September 8, 2018. doi:10.1007/978-3-540-89965-5_31. http://link.springer.com/10.1007/978-3-540-89965-5_31.

Moreau, Luc, Paul Groth, James Cheney, Timothy Lebo, and Simon Miles. 2015. "The Rationale of PROV". *Web Semantics: Science, Services and Agents on the World Wide Web* 35 (): 235–257. ISSN: 1570-8268, visited on September 8, 2018. doi:10.1016/j.websem.2015.04.001. http://www.sciencedirect.com/science/article/pii/S1570826815000177.

Moreau, Luc, Paolo Missier, Khalid Belhajjame, Reza B'Far, James Cheney, Sam Coppens, Stephan Cresswell, et al. 2013. "PROV-DM: The PROV Data Model". Visited on September 10, 2018. https://www.w3.org/TR/2013/REC-prov-dm-20130430/.

Muniswamy-Reddy, Kiran-Kumar, David A. Holland, Uri Braun, and Margo Seltzer. 2006. "Provenance-Aware Storage Systems". In *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, 4–4. ATEC '06. Boston, MA: USENIX Association. http://dl.acm.org.ezproxy.jyu.fi/citation.cfm?id=1267359.1267363.

Muschalle, Alexander, Florian Stahl, Alexander Löser, and Gottfried Vossen. 2012. "Pricing Approaches for Data Markets". In *Enabling Real-Time Business Intelligence*, 129–144. Lecture Notes in Business Information Processing. Springer, Berlin, Heidelberg. ISBN: 978-3-642-39871-1 978-3-642-39872-8, visited on March 29, 2018. doi:10.1007/978-3-642-39872-8_10. https://link-springer-com.ezproxy.jyu.fi/chapter/10.1007/978-3-642-39872-8_10.

Narkhede, Neha, Gwen Shapira, and Todd Palino. 2017. *Kafka - The Definitive Guide.* O'Reilly. ISBN: 978-1-4919-3616-0.

OpenRouteService. 2017. "OpenRouteService API Documentation (Isochrones, Directions, Matrix, Geocode)". Visited on September 11, 2018. https://openrouteservice.org/documentation/.

Opresnik, David, and Marco Taisch. 2015. "The Value of Big Data in Servitization". *International Journal of Production Economics* 165 (): 174–184. ISSN: 0925-5273, visited on March 29, 2018. doi:10.1016/j.ijpe.2014.12.036. http://www.sciencedirect.com/science/article/pii/S0925527314004307.

Pathirage, M., J. Hyde, Y. Pan, and B. Plale. 2016. "SamzaSQL: Scalable Fast Data Management with Streaming SQL". In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 1627–1636. doi:10.1109/IPDPSW.2016.141.

Peffers, Ken, Marcus Rothenberger, Tuure Tuunanen, and Reza Vaezi. 2012. "Design Science Research Evaluation". *Design Science Research in Information Systems. Advances in Theory and Practice:* 398–410. ISSN: 978-3-642-29863-9. doi:10.1007/978-3-642-29863-9_29. http://link.springer.com/10.1007/978-3-642-29863-9_29.

Peffers, Ken, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. 2007. "A Design Science Research Methodology for Information Systems Research". *J. Manage. Inf. Syst.* 24, number 3 (): 45–77. ISSN: 0742-1222, visited on March 15, 2018. doi:10.2753/MIS0742-1222240302. http://dx.doi.org/10.2753/MIS0742-1222240302.

Ramchurn, Sarvapali D., Trung Dong Huynh, Feng Wu, Yukki Ikuno, Jack Flann, Luc Moreau, Joel E. Fischer, et al. 2016. "A Disaster Response System Based on Human-Agent Collectives". *Journal of Artificial Intelligence Research* 57 (): 661–708. ISSN: 1076-9757, visited on September 8, 2018. doi:10.1613/jair.5098. https://jair.org/index.php/jair/article/view/11037.

Richter, Katja, and Holger Nohr. 2002. *Elektronische Marktplätze: Potenziale, Funktionen Und Auswahlstrategien.* Aachen: Shaker. ISBN: 978-3-8265-9890-6. http://www.shaker.de/de/content/catalogue/index.asp?lang=de&ID=8&ISBN=978-3-8265-9890-6.

Schmid, Beat F. B.F., and M.a. Lindemann. 1998. "Elements of a Reference Model for Electronic Markets". In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, 4:193–201. IEEE Comput. Soc. ISBN: 0-8186-8255-8. doi:10.1109/HICSS.1998.655275. http://ieeexplore.ieee.org/document/655275/.

Schomm, Fabian, Florian Stahl, and Gottfried Vossen. 2013. "Marketplaces for Data". *ACM SIGMOD Record* 42 (1): 15–15. doi:10.1145/2481528.2481532. http://dl.acm.org/citation.cfm?doid=2481528.2481532.

Stahl, Florian, Fabian Schomm, Lara Vomfell, and Gottfried Vossen. 2015. "Marketplaces for Digital Data : Quo Vadis ?" *Working Papers, ERCIS-European Research Center for Information Systems.* https://www.ercis.org/sites/ercis/files/structure/network/research/ercis-working-papers/ercis_wp_24.pdf.

Stahl, Florian, Fabian Schomm, and Gottfried Vossen. 2014. "Data Marketplaces: An Emerging Species". *Frontiers in Artificial Intelligence and Applications* Databases (August 2013): 145–158. ISSN: 9781614994572. doi:10.3233/978-1-61499-458-9-145. http://ebooks.iospress.nl/publication/38196.

Stahl, Florian, Fabian Schomm, Gottfried Vossen, and Lara Vomfell. 2016. "A Classification Framework for Data Marketplaces". *Vietnam Journal of Computer Science* 3, number 3 (): 137–143. ISSN: 2196-8888, 2196-8896, visited on March 29, 2018. doi:10.1007/s40595-016-0064-2. https://link-springer-com.ezproxy.jyu.fi/article/10.1007/s40595-016-0064-2.

Stonebraker, Michael, Uğur Çetintemel, and Stan Zdonik. 2005. "The 8 Requirements of Real-Time Stream Processing". *SIGMOD Rec.* 34, number 4 (): 42–47. ISSN: 0163-5808, visited on December 30, 2017. doi:10.1145/1107499.1107504. http://doi.acm.org/10.1145/1107499.1107504.

Tan, Benjamin, John-David Dalton, Kit Cambridge, and Mathias Bynens. 2018. "Benchmark.Js". Visited on September 25, 2018. https://benchmarkjs.com/.

Tang, Ruiming, Antoine Amarilli, Pierre Senellart, and Stéphane Bressan. 2014. "Get a Sample for a Discount", 20–34. Springer, Cham. doi:10.1007/978-3-319-10073-9_3. http://link.springer.com/10.1007/978-3-319-10073-9_3.

Tang, Ruiming, Huayu Wu, Zhifeng Bao, Stéphane Bressan, and Patrick Valduriez. 2013. "The Price Is Right". In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8056 LNCS, 380–394. Springer, Berlin, Heidelberg. ISBN: 978-3-642-40172-5. doi:10.1007/978-3-642-40173-2_31. http://link.springer.com/10.1007/978-3-642-40173-2_31.

Tas, Y., M. J. Baeth, and M. S. Aktas. 2016. "An Approach to Standalone Provenance Systems for Big Social Provenance Data". In *2016 12th International Conference on Semantics, Knowledge and Grids (SKG)*, 9–16. doi:10.1109/SKG.2016.010.

Taxidou, Io, Tom De Nies, Ruben Verborgh, Peter M. Fischer, Erik Mannens, and Rik Van de Walle. 2015. "Modeling Information Diffusion in Social Media As Provenance with W3C PROV". In *Proceedings of the 24th International Conference on World Wide Web*, 819–824. WWW '15 Companion. New York, NY, USA: ACM. ISBN: 978-1-4503-3473-0, visited on September 11, 2018. doi:10.1145/2740908.2742475. http://doi.acm.org/10.1145/2740908.2742475.

Truong, H. L., and S. Dustdar. 2009. "On Analyzing and Specifying Concerns for Data as a Service". In *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*, 87–94. Visited on February 20, 2018. doi:10.1109/APSCC.2009.5394136. http://ieeexplore.ieee.org/document/5394136/.

Truong, H. L., S. Dustdar, J. Gotze, T. Fleuren, P. Muller, S. E. Tbahriti, M. Mrissa, and C. Ghedira. 2011. "Exchanging Data Agreements in the DaaS Model". In *2011 IEEE Asia-Pacific Services Computing Conference*, 153–160. doi:10.1109/APSCC.2011.59.

Truong, Hong-Linh, Marco Comerio, Flavio De Paoli, G. R. Gangadharan, and Schahram Dustdar. 2012. "Data Contracts for Cloud-Based Data Marketplaces". *Int. J. Comput. Sci. Eng.* 7, number 4 (): 280–295. ISSN: 1742-7185, visited on August 19, 2018. doi:10.1504/IJCSE.2012.049749. http://dx.doi.org/10.1504/IJCSE.2012.049749.

Vijayakumar, Nithya N., and Beth Plale. 2006. "Towards Low Overhead Provenance Tracking in Near Real-Time Stream Filtering". In *Provenance and Annotation of Data*, 4145:46–54. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN: 978-3-540-46302-3 978-3-540-46303-0, visited on August 18, 2018. doi:10.1007/11890850_6. http://link.springer.com/10.1007/11890850_6.

Vu, Q. H., T. V. Pham, H. L. Truong, S. Dustdar, and R. Asal. 2012. "DEMODS: A Description Model for Data-as-a-Service". In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, 605–612. Visited on February 20, 2018. doi:10.1109/AINA.2012.91. http://ieeexplore.ieee.org/document/6184925/.

Wang, Yangjun. 2016. *Stream Processing Systems Benchmark: StreamBench.* Visited on December 30, 2017. https://aaltodoc.aalto.fi:443/handle/123456789/20991.

Yin, S., and O. Kaynak. 2015. "Big Data for Modern Industry: Challenges and Trends [Point of View]". *Proceedings of the IEEE* 103, number 2 (): 143–146. ISSN: 0018-9219, visited on March 23, 2018. doi:10.1109/JPROC.2015.2388958. http://ieeexplore.ieee.org/document/7067026/.

Zaslavsky, Arkady, Charith Perera, and Dimitrios Georgakopoulos. 2013. "Sensing as a Service and Big Data" (). Visited on January 24, 2018. arXiv: 1301.0159 [cs]. http://arxiv.org/abs/1301.0159.

Zhu, Hongwei, and Stuart E Madnick. 2009. "Finding New Uses For Information". *MIT Sloan Management Review* 50 (4): 18–21.

Appendices

A Metadata model example extract

Output data of the example in Figure 11.


```
{
  "prefix":{
    "prov":"http://www.w3.org/ns/prov#",
    "xsd":"http://www.w3.org/2001/XMLSchema",
    "datamarket":"N/A",
    "rights":"N/A",
    "pricing":"N/A",
    "dataquality":"N/A",
    "dataproduct":"N/A",
    "dataprovider":"N/A",
    "compliance":"N/A",
    "control":"N/A"
  },
  "agent":{
    "dataprovider:UniversityOfJyvaskyla":{
      "prov:type":"datamarket:DataProvider"
    }
  },
  "entity":{
    "dataproduct:exampleData1":{
      "prov:type":"datamarket:OriginDataProduct",
      "prov:time":"2018-09-24T12:07:34.891Z",
      "pricing:cost":{
        "$":0.001,
        "type":"xsd:float"
      },
      "pricing:currency":"EUR",
      "pricing:model":"perUse",
      "control:LawandJurisdiction":"FI"
    },
    "dataproduct:exampleData2":{
      "prov:type":"datamarket:OriginDataProduct",
      "prov:time":"2018-09-24T12:07:34.891Z",
      "pricing:cost":{
        "$":0.001,
        "type":"xsd:float"
      },
      "pricing:currency":"EUR",
      "pricing:model":"perUse",
      "control:LawandJurisdiction":"FI"
    },
    "dataproduct:exampleHybridData":{
      "pricing:cost":{
        "$":0.002,
        "type":"xsd:float"
      },
      "pricing:currency":"EUR",
```

```
      "pricing:model":"perUse",
      "prov:type":"datamarket:HybridDataProduct"
    }
  },
  "wasAttributedTo":{
    "_:id1":{
      "prov:entity":"dataproduct:exampleData1",
      "prov:agent":"dataprovider:UniversityOfJyvaskyla"
    },
    "_:id2":{
      "prov:entity":"dataproduct:exampleData2",
      "prov:agent":"dataprovider:UniversityOfJyvaskyla"
    }
  },
  "activity":{
    "dataproduct:combineExampleData":{
      "prov:startTime":"2018-09-24T12:07:34.891Z",
      "prov:endTime":"2018-09-24T12:07:34.891Z",
      "prov:type":"datamarket:createHybridDataProduct"
    }
  },
  "used":{
    "_:id3":{
      "prov:activity":"dataproduct:combineExampleData",
      "prov:entity":"dataproduct:exampleData1",
      "prov:time":"2018-09-24T12:07:34.891Z"
    },
    "_:id4":{
      "prov:activity":"dataproduct:combineExampleData",
      "prov:entity":"dataproduct:exampleData2",
      "prov:time":"2018-09-24T12:07:34.891Z"
    }
  },
  "wasGeneratedBy":{
    "_:id5":{
      "prov:entity":"dataproduct:exampleHybridData",
      "prov:activity":"dataproduct:combineExampleData",
      "prov:time":"2018-09-24T12:07:34.893Z"
    }
  },
  "wasDerivedFrom":{
    "_:id6":{
      "prov:generatedEntity":"dataproduct:exampleHybridData",
      "prov:usedEntity":"dataproduct:exampleData1",
      "prov:type":"prov:PrimarySource"
    },
    "_:id7":{
      "prov:generatedEntity":"dataproduct:exampleHybridData",
      "prov:usedEntity":"dataproduct:exampleData2"
    }
```

```
  }
}
```

B Data extracts from prototype

This section contains raw data snippets extracted from the prototype data pipeline that is used in the demonstration phase in section 5.3.

B.1 Data

Example of JSON hybrid data product generated by the prototype data pipeline:


```
{
  "id":"1|2674|0|82|23092018",
  "type":"bus",
  "trainId":null,
  "lineId":"3:RGTR--:555",
  "line":"555",
  "number":3413,
  "departure":1537693260,
  "delay":60,
  "live":true,
  "departureISO":"2018-09-23T11:01:00+02:00",
  "destination":"Huldange, Schmitt",
  "destinationId":110605006,
  "departureId":"110307001",
  "departureStation":{
    "type":"Feature",
    "geometry":{
      "type":"Point",
      "coordinates":[
        6.082928,
        50.118251
      ]
    },
    "properties":{
      "id":110307001,
      "name":"Lausdorn"
    }
  },
  "destinationStation":{
    "type":"Feature",
    "geometry":{
      "type":"Point",
      "coordinates":[
        6.024526,
        50.180537
```

```
        ]
      },
      "properties":{
        "id":110605006,
        "name":"Huldange, Schmitt"
      }
    },
    "distanceMatrix":{
      "distance":9515.68,
      "duration":550.82
    }
}
```

B.2 Metadata

Example of JSON W3C PROV metadata generated by the prototype data pipeline:

```
{
  "prefix":{
    "prov":"http://www.w3.org/ns/prov#",
    "xsd":"http://www.w3.org/2001/XMLSchema",
    "datamarket":"N/A",
    "rights":"N/A",
    "dataquality":"N/A",
    "dataproduct":"N/A",
    "datacompliance":"N/A",
    "datacontrol":"N/A"
  },
  "entity":{
    "dataproduct:departure":{
      "datamarket:source":"https://api.tfl.lu/v1/stream/
         StopPoint/Departures",
      "rights:derivationRights":true,
      "rights:commercialUse":false,
      "dataquality:accuracy":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:completeness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:uptodateness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "datacontrol:LawandJurisdiction":"LUX",
      "prov:type":"datamarket:OriginDataProduct",
```

```
      "prov:time":"2018-09-23T08:00:00.951Z"
    },
    "dataproduct:station-departureStation":{
      "datamarket:cost":"0.001",
      "datamarket:source":"https://api.tfl.lu/v1/StopPoint/110307001",
      "rights:derivationRights":true,
      "rights:commercialUse":true,
      "dataquality:accuracy":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:completeness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:uptodateness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "control:LawandJurisdiction":"LUX",
      "prov:type":"datamarket:OriginDataProduct",
      "prov:time":"2018-09-23T08:00:01.197Z"
    },
    "dataproduct:station-destinationStation":{
      "datamarket:cost":"0.001",
      "datamarket:source":"https://api.tfl.lu/v1/StopPoint/110605006",
      "rights:derivationRights":true,
      "rights:commercialUse":true,
      "dataquality:accuracy":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:completeness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:uptodateness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "control:LawandJurisdiction":"LUX",
      "prov:type":"datamarket:OriginDataProduct",
      "prov:time":"2018-09-23T08:00:01.356Z"
    },
    "dataproduct:departureWithCoords":{
      "datamarket:cost":"0.002",
      "prov:type":"datamarket:HybridDataProduct"
    },
```

```
    "dataproduct:distanceMatrix":{
      "datamarket:cost":"0.001",
      "datamarket:source":"https://api.openrouteservice.org/
         matrix",
      "rights:derivationRights":true,
      "rights:commercialUse":true,
      "dataquality:accuracy":{
        "$":0.9,
        "type":"xsd:float"
      },
      "dataquality:completeness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "dataquality:uptodateness":{
        "$":1.0,
        "type":"xsd:float"
      },
      "control:LawandJurisdiction":"EU",
      "prov:type":"datamarket:OriginDataProduct",
      "prov:time":"2018-09-23T08:00:01.659Z"
    },
    "dataproduct:departureWithDistanceMatrix":{
      "datamarket:cost":"0.003",
      "prov:type":"datamarket:HybridDataProduct"
    }
  },
  "activity":{
    "dataproduct:enrichCoordinates":{
      "prov:startTime":"2018-09-23T08:00:00.958Z",
      "prov:endTime":"2018-09-23T08:00:01.356Z",
      "prov:type":"datamarket:createHybridDataProduct"
    },
    "dataproduct:enrichDistanceMatrix":{
      "prov:startTime":"2018-09-23T08:00:01.359Z",
      "prov:endTime":"2018-09-23T08:00:01.659Z",
      "prov:type":"datamarket:createHybridDataProduct"
    }
  },
  "wasDerivedFrom":{
    "_:id1":{
      "prov:generatedEntity":"dataproduct:departureWithCoords",
      "prov:usedEntity":"dataproduct:departure",
      "prov:type":"prov:PrimarySource"
    },
    "_:id2":{
      "prov:generatedEntity":"dataproduct:departureWithCoords",
      "prov:usedEntity":"dataproduct:station-departureStation"
    },
    "_:id3":{
```

```
      "prov:generatedEntity":"dataproduct:departureWithCoords",
      "prov:usedEntity":"dataproduct:station-destinationStation"
    },
    "_:id8":{
      "prov:generatedEntity":"dataproduct:
         departureWithDistanceMatrix",
      "prov:usedEntity":"dataproduct:departureWithCoords",
      "prov:type":"prov:PrimarySource"
    }
  },
  "used":{
    "_:id4":{
      "prov:activity":"dataproduct:enrichCoordinates",
      "prov:entity":"dataproduct:departureWithCoords",
      "prov:time":"2018-09-23T08:00:00.958Z"
    },
    "_:id5":{
      "prov:activity":"dataproduct:enrichCoordinates",
      "prov:entity":"dataproduct:dataset1",
      "prov:time":"2018-09-23T08:00:01.356Z"
    },
    "_:id6":{
      "prov:activity":"dataproduct:enrichCoordinates",
      "prov:entity":"dataproduct:dataset2",
      "prov:time":"2018-09-23T08:00:01.356Z"
    },
    "_:id9":{
      "prov:activity":"dataproduct:enrichDistanceMatrix",
      "prov:entity":"dataproduct:departureWithCoords",
      "prov:time":"2018-09-23T08:00:01.359Z"
    }
  },
  "wasGeneratedBy":{
    "_:id7":{
      "prov:entity":"dataproduct:departureWithCoords",
      "prov:activity":"dataproduct:enrichCoordinates",
      "prov:time":"2018-09-23T08:00:01.356Z"
    },
    "_:id10":{
      "prov:entity":"dataproduct:departureWithDistanceMatrix",
      "prov:activity":"dataproduct:enrichDistanceMatrix",
      "prov:time":"2018-09-23T08:00:01.659Z"
    }
  }
}
```