



**Hochschule
Bonn-Rhein-Sieg**
University of Applied Sciences

MASTER'S OF COMPUTER SCIENCE

Developing an abstract Quantified Self Provenance-Model

Author:

Bojan JANISCH

Supervisors:

Andreas SCHREIBER
Prof. Dr. Ralf THIELE

*A project report submitted in fulfillment of the requirements
for the degree Master's of Computer Science*

in collaboration with

[The German Aerospace Center](#)



**Deutsches Zentrum
für Luft- und Raumfahrt**

Simulations- und Softwaretechnik

August 2015

Declaration of Authorship

I, Bojan JANISCH, declare that this document titled, 'Developing an abstract Quantified Self Provenance-Model' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the report is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Contents

Declaration of Authorship	i
Contents	ii
Abbreviations	iii
1 Introduce the WWW - The What, Where and Why	1
1.1 The Working Environment - Always worth an introduction	1
1.2 The Motivation - Why doing all this?	2
1.3 The project's frame work	3
1.4 The report's structure	3
2 Basics - What you should know to read further	4
2.1 Quantified Self - More than a social movement	4
2.1.1 What is Quantified Self?	4
2.1.2 A brief history of Quantified Self	4
2.2 Pedigree - More than dog food	5
2.2.1 W3C's definition of provenance	6
2.2.2 PROV DM - A generic provenance data model	6
2.3 Graph Databases - A glimpse at it	8
2.3.1 Property Graphs - Why are they mentioned?	9
2.3.2 A sample Graph Database - Neo4j	9
2.4 ProvStore - A free provenance repository	10
3 Preparations - Things to handle before we start	11
3.1 Quantified Self Workflows - How do People quantify themselves?	11
3.1.1 The Quantified Self Scope - User, Devices, Applications and Services . . .	11
3.1.2 Quantified Self - Let's get practical	13
3.1.3 A Sample QS Workflow	14
3.2 Provenance Questions - Reason and Samples	15
3.3 Quontology - A Quantified Self Ontology	17
3.4 The PROV-N Mapper	19
4 The Quantified Self Provenance Model - An approach	20
4.1 Commonalities in Quantified Self Workflows	20
4.2 The Provenance Model	21
5 A Quantified Self Provenance System - Some thoughts	25
5.1 Disclosed Provenance Systems - Requirements, Effort and Quality	25
5.2 Observed Provenance Systems - An Introduction of multiple approaches	26
6 Approached Dead Ends	28
6.1 Data Modeling: Bottom-Up	28
6.2 Representation of Quantified Self Workflows	28
7 Conclusions, Experience and Final Thoughts	29
7.1 Quantified Self and Provenance - A field that needs to be researched	29
7.2 My work at the DLR	30
Primary Literature	iv
Secondary Literature	iv

Abbreviations


DBMS	Database Management System
DLR	The German Aerospace Center
NoSQL	Not only SQL
QS	Quantified Self
SC	Simulation and Software Technology Facility
SRV	Software for Space Systems and Interactive Visualization
VSS	Department Distributed System and Component Software
W3C	World Wide Web Consortium

1 Introduce the WWW - The What, Where and Why

This report discusses a project submitted in fulfillment of the requirements for the degree Master's of Computer Science at Bonn-Rhein-Sieg University of Applied Sciences in collaboration with the German Aerospace Center. This chapter introduces the master's project regarding the working environment in section 1.1. The motivation behind this work is elaborated in section 1.2, followed by a description of the project's frame work in section 1.3. Additionally a brief structural overview of this document will be given in section 1.4.

1.1 The Working Environment - Always worth an introduction

This master's project was processed at and in collaboration with the German Aerospace Center (DLR). The DLR is Germany's national research center for aeronautics and space. Its focus areas in research and development work are aeronautics, space, energy, transport, defence and security. The DLR is also responsible for planning and implementing Germany's space program. It employs approximately 8000 people at 32 institutes and facilities at 16 locations in Germany. With Cologne as headquarter the DLR is also present in Augsburg, Berlin, Bonn, Braunschweig, Bremen, Göttingen, Hamburg, Jülich, Lampoldshausen, Neustrelitz, Oberpfaffenhofen, Stade, Stuttgart, Trauen and Weilheim. The DLR also has offices in Brussels, Paris, Tokyo and Washington DC. [Deu14]

The Simulation and Software Technology Facility (SC) researches and develops software engineering technologies from which many are incorporated into DLR software projects. SC is divided into the departments "Distributed Systems and Component Software" (VSS) and  software for Space Systems and Interactive Visualization" (SRV).

VSS currently manages three working groups focused on distributed software systems for aerospace, suitability diagnostics and telemedicine, high performance computing in simulation of flow processes or interactive visualization, and software engineering in topics like provenance, process-visualization and knowledge management.

In SRV four working groups are currently researching in the field of simulation and modeling: "Model-based Systems Engineering" (MBSE), in the domain of embedded systems: new concepts for onboard software in satellites and orbiters including topics like onboard navigation and "Command & Data Handling" systems, in scientific visualization: the development of solutions for various projects like "classical" flow processes, but also energy flows of rail vehicles or an urban disaster management system and lastly 3D interaction of foreign planet surfaces. [Hem15]

1.2 The Motivation - Why doing all this?

In almost any medical visit nowadays it is common to collect additional data of a patient in form of a medical history or raw data like weight, height and blood pressure. The data gives the doctor additional knowledge about your health status, possible treatments or indication of diseases.

Quantified Self (QS) is a recently upcoming movement that describes a community of people (users) with a similar aim to that of the doctor. The user collects various types of data, related to him, to get a better understanding of himself. The main differences between the data collected in a medical visit and recorded by the user are, that the data recorded by the user can cover a much wider area of interest, e.g. calories consumed or miles walked and additionally it is - in general - recorded in a much shorter period, enabling precisely analytics, that result in more reliable diagnostics than in any medical history collected by the doctor.

The goals of the community are supported by software developers and manufacturers that are providing applications and gadgets for the user that are easy to use and mostly inter-operable with each other, so that the user can create a flexible chain of different gadgets and programs that are analyzing his data and showing him easy understandable results.

Since the QS movement came up just a few years ago [Kel07] it is still a young field of research with many open questions from users, like “Can I trust the developer with my data?”, “How dangerous can it be quantifying and optimizing myself without proper medical support, e.g. through a doctor or personal coach?” or “What are the benefits and risks on long term?”, which will need to be answered by science in the future.

One step to answering a bunch of questions and to get an overview of this complex process-chains would be to understand how the data of the user is created, manipulated and processed, which is described by its provenance. Provenance has many synonyms like lineage, genealogy or pedigree that results from different domains in which they were elaborated separately until a few years ago.

By capturing and storing provenance of the behavior of users practicing QS - in short capturing QS workflows - the user could answer questions regarding the creation process of his data, security aspects and even determining the trustfulness of his diagnostic results. Like any database management system (DBMS), a provenance system needs an underlying data model, a provenance model, that defines the data that will be captured in QS workflows.

Developing such a provenance model, usable for any QS workflow and thus starting the exploration of provenance in QS, is the main target and motivation of this master’s project.

1.3 The project's frame work

As aforementioned, this project aims to define a provenance model for data in QS workflows, based on W3C's generic provenance model: PROV-DM. For this, specialization of the data model is necessary, additionally to the realization of a capture system for provenance data in QS workflows. The implementation of a provenance storage is out of scope for this project, hence an available storage like ProvStore shall be used, with an optionality to store the provenance in a local database using PyProv. Since the developed provenance data model will be published as open source, the aftercare of it shall be given to the community. The project will be processed mainly at SC's headquarter in Cologne.

1.4 The report's structure

The report is structured into seven sections. After this introduction, section 2 discusses some basic knowledge, that is required to understand further decisions taken in the project. Section 3 explains some requirements for the developed provenance data model, which is elaborated in section 4. Although not implemented, section 5 discusses some problems and approaches in recording provenance data suited for the data model. Since good and bad decisions were made during this project, section 6 elaborates some huge mistakes. Finally section 7 concludes this report with some final thoughts and gained experience while working intensively at a young field of research and at the DLR.

2 Basics - What you should know to read further

To understand more of the relevant provenance data itself, the domain at which it will be collected, needs to be understood and thus QS will be introduced in subsection 2.1. After that provenance in general will be elaborated in subsection 2.2, its meaning and definition. Also a generic provenance data model will be explained on which the final provenance model will be based. Subsequent to provenance, subsection 2.3 thematizes graph databases to get an understanding of how provenance can be stored. Finally this section concludes in subsection 2.4 with ProvStore, an online repository for W3C's provenance documents.

2.1 Quantified Self - More than a social movement

Several years ago, a movement came up through the social environment of people. Suddenly more and more people quantified different aspects of their life through modern technology (mostly wearable) over a longer time period. This report does not aim to explain the reasons of the movement or its future behavior. Instead this subsection explains what QS represents, how far does it reach and how it was founded.

2.1.1 What is Quantified Self?

Up till today there is no clear definition of QS, because the term refers to two dimensions of this movement. The first includes the idea of getting knowledge about oneself through quantifying and analyzing self-related data as it is described in the communities slogan: "Self knowledge through numbers". The second refers to the community itself, which members are tracking aspects of their lives and sharing it for competitions or interest with sympathizers. They're connected through a central website [Lab15b], social networks (e.g. Facebook groups) or real-life Meetup groups [Mee15]. Quantified Self is constantly evolving new use cases and needs through the community, but also from developer and manufacturer which constantly develop new approaches of human quantization.

2.1.2 A brief history of Quantified Self

Although the idea of quantimetric self tracking began in 1970s [Rip+13], the movement called "Quantified Self" is a protected term that came up in 2007 by Gary Wolf and Kevin Kelly. In June 2010, Wolf held a talk at TED where he introduced QS [Wol15]. Almost one year later, in May 2011, the first annual international QS conference was held in Mountain View, California. Since 2013 there exist also an annual QS conference in Amsterdam, called QS Europe Conference [Lab15a].

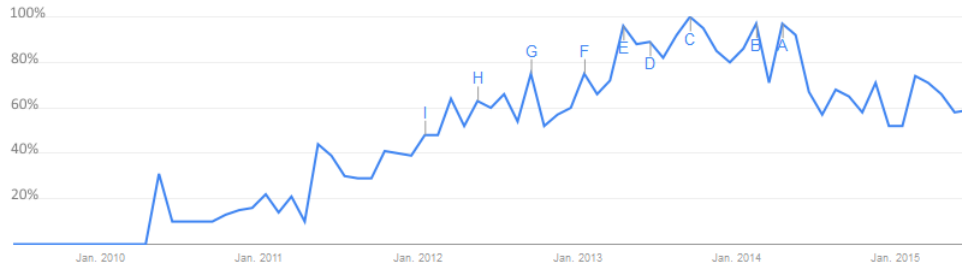


FIGURE 1: Google trends for “Quantified Self” in percent from June 2009 to June 2015. [NN15a]

Figure 1 shows the Google Trend [NN15a] of the term “Quantified Self” from June 2009 to June 2015 in percentage. Even though it seems that the interest is dropping from 2014, it still shows an overall progression of interest in QS.

Today the global community includes over two hundred groups in 38 countries and consists of more than 50.000 members. The largest group meets in San Fransisco and counts currently over 4000 members.

2.2 Pedigree - More than dog food

Pedigree, lineage, parentage or genealogy, all these words arose from different domains but refer to the same thing: provenance. Provenance is defined by The Oxford Dictionary [NN15b] as “The place of origin or earliest known history of something.”

The term is well known through the domain of art - where it refers to the history of an art work - or in geology, where the history of sediments movements is reconstructed. Ensuing to this, provenance of data describes the origination process of a piece of data in such a way, that if all these processes and related data that were created or used by them, are interacting in the same way again, would result in the exact same data.

In computational environment provenance is well elaborated in the meteorologic domain, where the data needs to be trusted to create a dependable the weather forecast. Other areas, like QS are still lacking of any provenance elaboration. This development of separate elaboration of provenance as a field of research was also realized by “The EU Provenance Project”, a partnership of six well-known companies and universities, which developed a standardized open provenance architecture in 2008 [Mor+08]. It was the first project of elaborating a standardized provenance system, which was also used as basis to define a generic provenance model.

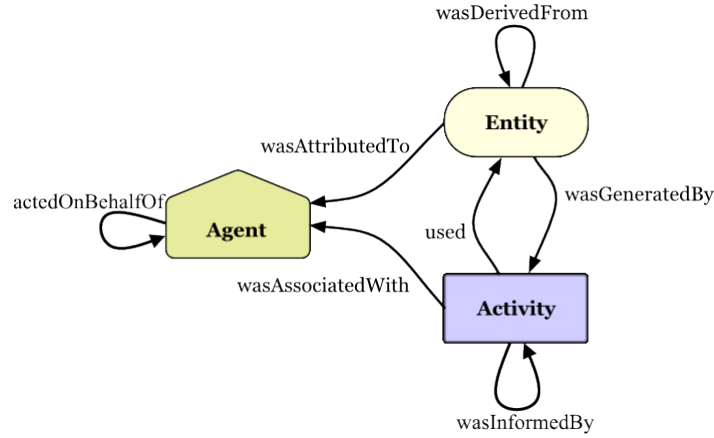


FIGURE 2: PROV DM core structure. [Gro15a]

2.2.1 W3C's definition of provenance

To define a solid baseline for further work, a clear definition of provenance is needed. Simmhan et al. [Sim+05] surveyed five different provenance systems for different domains, that differ in data capture and storage approaches. To avoid a mix of definitions from different sources, the definition of a prestigious council, the World Wide Web Consortium (W3C), shall be accepted as definition for provenance. The W3C defines provenance as following:

“Provenance of a resource is a record that describes entities and processes involved in producing and delivering or otherwise influencing that resource. Provenance provides a critical foundation for assessing authenticity, enabling trust, and allowing reproducibility. Provenance assertions are a form of contextual meta data and can themselves become important records with their own provenance.” [Gro15b]

Applying this definition to the provenance of QS data, it describes a documentation-like record of every process - manual, semi-automatic and automatic - in the QS workflow, taking into account the generated and used objects, participated people and various environmental parameter like devices, location, activity and many more.

2.2.2 PROV DM - A generic provenance data model

With the previous definition, W3C started in 2011 and finalized in 2013 a generic provenance model, inspired by various different approaches, that is - in general - adaptable to any domain.

The general provenance model, illustrated in figure 2, can be seen as a property graph with - at its core - three different types of nodes: **Entities**, **Activities** and **Agents**.

An entity represents a physical (e.g. a scale), digital (e.g. data), conceptual (e.g. a plan) or any other kinds of object. An activity is a process that uses or generates entities and that can be

TABLE 1: Mapping of PROV core concepts to types and relations. [Gro15a]

PROV Concepts	PROV-DM types and relations	Name
Entity	PROV-DM Types	Entity
Activity		Activity
Agent		Agent
Generation	PROV-DM Relations	WasGeneratedBy
Usage		Used
Communication		WasInformedBy
Derivation		WasDerivedFrom
Attribution		WasAttributedTo
Association		WasAssociatedWith
Delegation		ActedOnBehalfOf

associated with an agent, meaning that the agent is responsible for the activity. In QS context the activity could be weighing, which generates data of the user's weight by using the data given from the scale's sensor. The agent, usually a person, organization or even a piece of software, is in some way responsible for an activity, through this connected to an entity and infers trust onto these. Using the previous example, weighing can be associated to the user because he triggered the process and is responsible for executing it correctly. Also the scale, produced by a trustful manufacturer, can be attributed to him and thus inferring trust onto the entity.

The core structure also consists of seven different relations, represented as directed edges of the graph, that describe the actual provenance. Table 1 shows a mapping of concepts with their respective type and name.

Generation and usage are relations that describe when an activity generates or uses a specific entity, like in the previous example. A communication describes a relation between two activities when one triggers the other. For example using the scale again, the process of showing the weight to the user would be informed by the activity weighing, which is a communication concept.

Derivation describes an assumption between two entities where one is generated through modification or process of the other and thus can be derived from it. Even though in simple cases this may be inferable, it should not be taken as granted. The n-m problem [Car+14] states that, if an activity uses three entities and generates two new, there is no assurance which generated entity was derived from a used.

There might be cases in which the generation of an entity is unknown or irrelevant. To ensure trust to that entity, the attribution relation allows agents to take responsibility for an entity. To ensure trust in activities, a similar relation - association - was defined for agents and activities.

Finally an agent can act on behalf of others or delegate its responsibility to other agents. For this case delegation was defined which is a relation between two agents.

Additionally to table 1 there exist a special entity type, called Bundle, which serves the purpose of representing the provenance of provenance. To structure the granularity of provenance, bundles can represent fine captured provenance of complex system, into a single entity, that refers to it and thus simplifying provenance graphs.

As stated before the provenance model resembles a property graph, meaning that nodes and edges contain specific properties. A property of a node usually describes a time irrelevant fact like the name of an user (agent) or the value of an data (entity), while a property of an edge usually stores additional event-based and as such, time-based information. Since provenance of something is a chronological course of events, time stamps are crucial properties and defined in relations between activity and entity. Also activities were given start and end time by the W3C. More properties of PROV DM's core structure are described in [Gro15a].

To integrate machine processing and communication of provenance through humans, W3C defined for the representation of a provenance graph an own notation, called PROV-N. The syntax follows similar principles to XML and JSON, thus can be converted to each of them, called PROV-XML and PROV-JSON. A document containing provenance data is called provenance document [Gro15c].

2.3 Graph Databases - A glimpse at it

This subsection is concerned with graph databases, the difference to relational databases, their usefulness for provenance querying and Neo4j (with Cypher) as sample database.

Unlike relational databases that store information in tables, graph databases store information in a graph structure. Due to this graph databases are part of the NoSQL concept databases, like key-value, column-family and document stores, that use techniques that differ from the relational approach to store and manage data [Van14].

Graph databases are suited well for highly connected data, meaning that there are many relations between objects. Since most NoSQL DBMS do not require a pre-defined scheme [Kar15], it is possible to change the data model during runtime, which increases the complexity and changeability of the data.

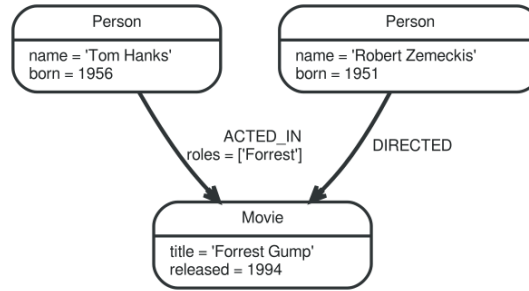


FIGURE 3: Sample property graph from Neo4j docs [Inc15c].

2.3.1 Property Graphs - Why are they mentioned?

Property graph models are - in some way - specialized graph models, because they're optimized for: directed graphs (meaning that relationships can differ outgoing from incoming nodes), multi-relational graphs (meaning that the graph contains multiple types of relationships) and storing key-value pairs as properties of nodes and relationships. Figure 3 visualizes all specializations in a compact sample graph.

Provenance data already resembles a property graph and so graph databases with a property graph are suited as storage of provenance data. Since there exist multiple approaches to realize property graphs in databases, like the resource description framework (RDF) data model that uses rows of triples - subject, predicate and object - to formalize statements and building a directed graph, we decided to use Neo4j as provenance data storage, because it already implements natively a property graph.

Due to this provenance data can be directly imported and exported from the database, making it suitable as provenance storage. The differences between RDF and Neo4j were already revealed in [Abr+13] and shall not be elaborated further.

2.3.2 A Graph Database - Neo4j

Neo4j [Inc15b] is one of the most popular open source (restricted to community edition) graph databases written mainly in Java and was initially released in 2007 by Neo Technology. With nodes, relationships and properties Neo4js DBMS implements a property graph, like the aforementioned provenance model resembles, making it easily importable into the database. Neo4j supports materialization of relationships at creation time, constant time traversals for relationships in the graph, compact storage and memory caching. Due to these features, Neo4j is able to handle up to 34 billion nodes and relationships with an overall maximum of up to 274 billions properties and 32.000 different relation types [Inc15a].

With version 2.0, Neo4j introduced labels as optional characteristics beside properties, which allows pre-clustering of nodes and relationships, to increase the performance of querying big data. Querying can be done by via **Cypher**, Neo4js own querying language. Cypher is inspired by SQL, resulting in similar keywords and syntax. Querying a graph means traversing from a start node, through a specified algorithm to the searched end node(s). Unlike Gremlin, which builds its complexity iteratively through concatenating multiple filters and creating huge unreadable query statements, Cypher simplifies queries syntactically and semantically through a SQL similar way of outlining the wanted node(s). For clarification listing 1 and 2 show the same query written in Cypher and Gremlin. Holzschuher and Peinl compared graph querying languages more detailed in [HP13].

```
1 MATCH    person-[:KNOWS]->friend-[:KNOWS]->friend_of_friend
2 WHERE    not (friend_of_friend<-[:KNOWS]-person)
3 RETURN    friend_of_friend, COUNT(*)
```

LISTING 1: Cypher query from [HP13]

```
1 t = new Table();
2 x = [];"
3 g.idx('persons')[[id:id_param]].out('FRIEND_OF').fill(x);"
4 g.idx('persons')[[id:id_param]].out('FRIEND_OF').out('FRIEND_OF').dedup().except(x).id.as('ID').
   ➔ back(1).displayName.as('name').table(t,['ID','name']){it}{it}.iterate();
```

LISTING 2: Gremlin query from [HP13]

2.4 ProvStore - A free provenance repository

ProvStore [HM14] is the first online public repository supporting W3C's provenance standard. Developed in 2014 at the University of Southampton it allows users to store, publish, validate and visualize provenance documents. A provenance document can be uploaded manually via browser or through a REST API as a private or public document.

ProvStore provides multiple views on the provenance document including: Data (focused on information flow and entity transformation), Process (concerned with the processes that participated) and Responsibility (highlights responsibility of agents). Additionally ProvStore can “flat” a provenance document through extracting a bundle's content into the same hierarchy, which allows relating specific objects of the bundle with the rest of the provenance document.

ProvStore is based on “prov” [Huy15], a python library for W3C's provenance standard. The library is licensed under “MIT License” and thus open source.

3 Preparations - Things to handle before we start

This section is concerned with preparations for developing a universal provenance data model for QS workflows. Hence, first QS workflows are elaborated in subsection 3.1. After that, subsection 3.2 thematizes provenance questions, which are needed to specifying the scope of the provenance data model. Since the based provenance data model, elaborated in section 2.2.2, is generic and not suited for semantically interpretation of data yet, subsection 3.3 elaborates an own developed ontology, that can be used on top of W3C's PROV-DM. Lastly since this projects relies on ProvStore as provenance storage, subsection 3.4 introduces the PROV-N Mapper, a program with the aim to store provenance graphs in local graph databases, currently supporting only Neo4j.

Aside from the provenance data model or storage approach, a capturing approach should also be introduced in this section. But, since project time was too short to realize a complete provenance system, this part has been moved from preparations to some subsequent thoughts in section 5.

3.1 Quantified Self Workflows - How do People quantify themselves?

To develop a provenance model, abstract enough to be used in every QS workflow, these workflows needs to be elaborated first. After a theoretical elaboration of QS given in section 2.1, this section focuses on workflows in QS, their definition, scope and problems.

A QS workflow is defined as the data flow of a complete process of activities, triggered by the user, going from the actual record of user's data to its visualization. Hence a QS workflow is always viewed from the position of the user, which activities he's done and which devices, application and services he's used.

In the following the scope and complexity of QS workflows will be described in section 3.1.1. After that section 3.1.2 will elaborate the possibilities and current general usage of these workflows from a user's perspective. To clarify the previous theoretical elaboration, section 3.1.3 will describe a real life sample QS workflow.

3.1.1 The Quantified Self Scope - User, Devices, Applications and Services

As before mentioned, a QS workflow starts and ends with the user. Usually the user starts a workflow with an activity, that is recorded or functions as trigger for other activities. After that the user interacts in many different ways with various activities of the workflow until he see the results of his tracking. All activities of the workflow can be structured into four categories: user, device, application and service.

Each category represents a “technological zone”, mostly defined by Barooah et al. in [Bar+14]. All activities in each zone share a similar technological environment and thus interaction space with the user.

The most basic way of QS consists of the **user** alone. The user can track himself through common tools, like a scale, a watch, a map or any other utility that helps him to quantify himself. Hence the first zone does not need to have any technology in it. This zone is mentioned, because the user has the possibility to record himself manually, completely without technology. Although technological gadgets create a simpler way to capture provenance, there will always be a part in a QS workflow that is only executable and recordable by the user and hence cannot be ignored.

A **device** is an application-specific computer, e.g. an activity tracker. Even though differences in usage, appearance and even scope exist, devices share the common target to collect automatically or semi-automatically data from the user, via peripheral sensors. These include wearables like bands, straps, sport watches or any other wearable gadget with the purpose of quantifying the user, but also non-wearable devices like a digital online scale. Since every device is a computer that has software running on it, only these computers are devices that use - from a user's perspective - transparent software, that is “baked on” the hardware, to collect the data.

An **application** is software running on a computer. The appearance of the computer has no restrictions (e.g. smart phone / watch, PC, tablet or other), but it needs to have a communication channel to the user. The difference to the software on a device is that the user is aware of the application, is interacting with and exploring it on a more complex way than choosing its functionalities. Also applications are installable and removable from a computer by the user.

A **service** is also software running on a computer. In this case the software is not running locally within the user's environment, but on a server that is accessible through websites. Although not very different to the user, from a software engineering perspective it is completely distinct, because the administration and capturing of internal processes are completely determined by the owner of the service. Additionally services are not installable and removable by the user.

A QS workflow is created through the user's interaction with applications in these zones, that are usually communicating through various opened APIs with each other. This communication creates a data flow between the zones, that can vary its direction multiple times. Also in most cases there's no predefined order of interaction with the user. The user alone decides which activities he uses, what order they have and how complex his QS workflow grows by actually practicing QS. This means the scope of a QS workflow is not pre-definable and differs from user to user and use to use, because he does not need to interact always with the same activities.

TABLE 2: Quantified Self - Practical Overview [TS15]

Area	Data	Tools
Endurance Sports	Time, Heart Rate, Speed, Frequency	Heart Rate Monitor, Smart Phone Application
Athletic Sports	Weights, Repetitions, Pause Duration	Pen & Paper, Stopwatch, Smart Phone Applications
Nutrition	Macronutrient Composition (Fat, Protein, Carbs), Calories, Weight	Kitchen Scales, Nutrient Tables
Productivity	ToDo's, Written Mails, Time Scope, Used Applications	Desktop Software, Smart Phone Applications
Finances	Expenses and Earnings of different categories	Excel, Desktop Software, Smart Phone Applications
Sleep	Duration, Quality, Volume, Light Intensity, Sleep Time	Sleep Tracker, Smart Phone Applications
Blood Values	Miscellaneous Hormones, Inflammation Marker	(Send In) Labo-Kits
Visited Places	GPS Data	GPS Logger, Smart Phone Applications

3.1.2 Quantified Self - Let's get practical

Due to the in section 2.1.1 mentioned slogan, a user should ask himself first “What do I want to know about myself, why and how do I track it?”. Since there are limitless approaches in quantifying his data, these questions will help him to specify which data he needs to record and what approaches are suited for him. They can reach from manual records with pen and paper to automatic recording through high tech wearables and everything in between.

Rather than presenting multiple devices or applications, it will be more useful to define the areas that are currently usually practiced. Table 2 presents some examples of recordable areas, data which is useful and easy to be quantified in this area and possible tools to record it. Although not complete, the table can give some ideas of things to track.

If decided what to track, the next step is to look for methods how to record the correct data. There's a broad market for wearable QS gadgets and applications for smart phones. Famous manufacturer for QS devices are Fitbit, Jawbone and Withings, supporting mostly activity tracking like steps taken, calories burned or miles passed.

Tracking one sort of data may be sufficient for answering simple questions like “How long did I sleep yesterday?”, but it cannot answer more complex questions like “Why did I sleep x hours yesterday?” or “How can I improve my sleep?”. To get a more complex knowledge about oneself the data needs to get more complex. Usually homogeneous is are very limited in information gain while heterogeneous data can be interpreted in many different ways. In QS this usually

means, the more heterogeneous the data about oneself is, the more knowledge about oneself can be interpreted.

Since the community - including the developer and manufacturer - is well known among themselves, many application and devices are compatible with each other, which allows the creation of an arbitrarily complex profile of the user and thus creating an arbitrarily complex QS workflow. This is especially supported by web services like Fluxstream [LK15] or Zenobase [LLC15], which are called aggregation portals and serves only the purposes of aggregating and visualizing data from various different input sources.

3.1.3 A Sample QS Workflow

To get a better understanding of the complex composition, this section elaborates a relative compact sample workflow in which a user is only tracking one data type. Figure 4 visualizes the workflow of a user that tracks his weight. Drawing was chosen as representation of workflows because the user could intuitively reflect his actions through a data flow diagram. Due to missing standards to describe QS workflows, the user had to draw his workflow using the following guidelines:

1. Each workflow starts and ends with the user.
2. For every component it must be clear, where the data is coming from and going to, denoted through a directed edge.
3. If it's known what data is transmitted between components, it has to be described beside the edge.

A component in a workflow is a process - independent of granularity - that is triggered by something and executed. The components were clustered by the aforementioned zones, excluding the user zone, because the user has an affinity to technology.

The workflow start with the user, who steps on Withings' online scale to measure his weight. After measurement, the scale connects through WLAN to Withings' cloud storage to update the weight data set. Withings' notification server is executed after the update to notify the subscriber about the data change. Withings' Health Mate application updates the data of the user at start and shows him the progress of his weight. The user now shares his new weight over Twitter through a button in the application, which will be displayed to the user's subscriber on Twitter. At the same time the user can read his own post in his Twitter application. Since the Weight Companion application is also a subscriber of Withings' notification server, the data in the application will be updated at start too. It also shows him his progress in weight loss, but

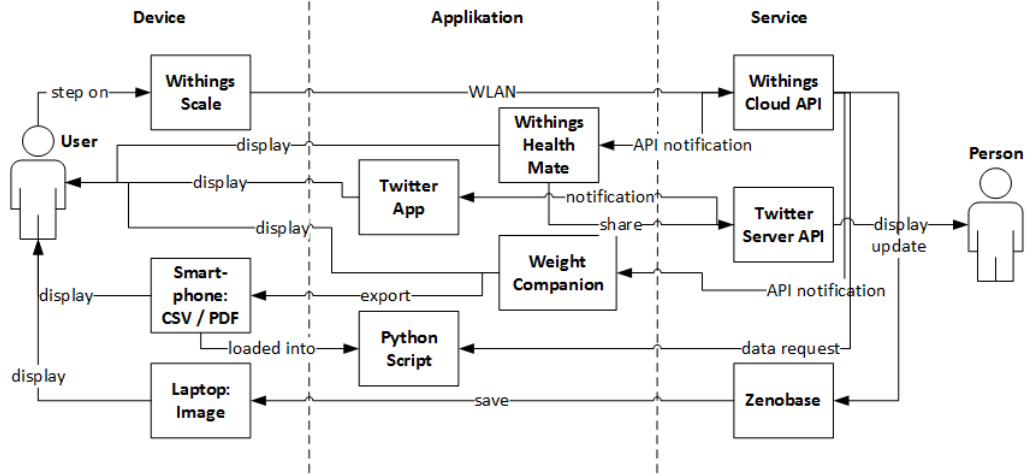


FIGURE 4: A sample workflow of a user tracking his weight

additionally the user is exporting his data as CSV to his smart phone's storage. Through the smart phone he can see his exported data and load it from a self written python script at which it will be further processed. Additionally the script can update his data set by a request call at Withings' cloud API. Lastly the user connects the data of Withings' cloud through the API with Zenobase, an aggregation portal accessible through the web, at which it will be aggregated with other data and saved as an image at the user's laptop.

This example should clarify the complexity in capturing provenance of QS workflows. Although if the data transmitted between those components is known, it is not possible to predict the order and thus the provenance of how data was derived, e.g. the origin of python script's data. Additionally there exist more than one QS workflow from a user, even if it's just one data type he's tracking. Every path from user to user in figure 4 represents a QS workflow. And finally the abstraction level of components in QS workflow can differ, e.g. between a scale which has no other functionalities or a smart phone which has thousands of it.

3.2 Provenance Questions - Reason and Samples

One of the main tasks of provenance is to answer questions regarding the process that led to a specific piece of data, its corresponding data and agents that were involved in the progress. Since provenance data could be anything related to the origination process, some sort of filter is needed, so that a provenance system can focus its resources into recording specific data, rather than simple audit everything possible.

In data modeling such a focus is usually described with use cases. Since one of the aforementioned task of provenance is answering questions, it is obvious that for this case, to describe the focus through questions which are of interest to be answered. These questions can be modified or

extended during the modeling process, because the interest on which they are based could change during a project's lifetime.

In this subsection, ten such questions will be defined and explained regarding the importance and reason for them. Since provenance can be separated into three views (entity, activity and agent), the following questions will be clustered by these.

Note that these questions are just an excerpt of the many possible provenance questions in this domain, but enough to give an overview about its feasibility. By extending the question list, the resulting provenance model may need to be extended too.

The following questions were formalized from the position and interest of a regular user. To keep the first generalized QS provenance model simple and reduce the complexity of the provenance system due to the short project time frame, mostly very general questions were considered.

Entity focused questions

1. **What data about the user were created during the activity X?** Sometimes the user would want to know how much data of him was collected during an activity. This question aims to reveal all data generated during an specific activity.
2. **What data about the user were automatically generated?** This question is a specialized form of the previous one. Since all data during an activity can be classified into automatically and manually generated data, this question sets a focus on the first one.
3. **What data about the user were derived from manual input?** Although the user might know which data he entered, this question aims to reveal the data, that was derived from his manual input. Sometimes it's not clear to the user, what can be gained by the data he's giving to others.

Activity focused questions

4. **Which activities support visualization of the users data?** In some complex cases, there can exist possibilities to use software or hardware in ways that the user did not know. This question aims to reveal all possibilities on visualizing the user's data.
5. **In which activities can the user input data?** Like the previous question, this question aims on possible activities on hard- or software that the user is not aware of. In this case he could want to know which devices, applications and web services support manual data input.
6. **What processes are communicating data?** This general question would - in best case - reveal the complete QS workflow to the user. Depending on the effort a user takes in

quantifying himself, he might lose the overview of his QS workflow. By answering this question, the user could easily keep the overview of it.

7. **Which activity generated the origin of data X?** The final activity focused question is of a more complex nature. Some users might want to know how a specific entity came into existence, especially where the origin data came from.

Agent focused questions

8. **What parties were involved in generating data X?** This question, due to agent focused view, aims like many questions of this view, a security or trustiness aspect of the QS workflow. Often the user shares data through applications and web services and view the visualized results without really knowing who processed his data. This question shall reveal all parties that take responsibility in generating a specific piece of data.
9. **What parties got access on data X?** Although the user may know who processed his data, it's a different story to have a particular access on a specific data, reusing it several times. This question targets to reveal all security issues which the user might be not aware of.
10. **Can other parties see user's data X?** This question has a slightly different focus from the previous one. In some applications and web services, the user can grant access of his data to other people, e.g. for competitive reasons. To get an overview, whom users he had given access to a specific piece of data, is this question's target.

3.3 Quontology - A Quantified Self Ontology

Although, capturing provenance with a generic data model could be implemented, the missing semantics would constraint the information gain of such a provenance system. Due to this Quontology, a “**Quantified Self ontology**” was developed in this project. Based on Noy's and McGuinness' ontology creation guide [NM01] it allows inferring semantics onto PROV-DM types and relations.

Figure 5 visualizes the current ontology at a first version. The relation between PROV-DM and Quontology can be seen through the similar hierarchy and label they share. Since it infers semantics on a already defined data structure, Quontology is structured into three domains: **Agents**, **Activities** and **UserData**. Each domain extends in some way W3C's PROV-DM: Agents → Agents, Activities → Activities and Entities → UserData. In the following only the extensions are explained.

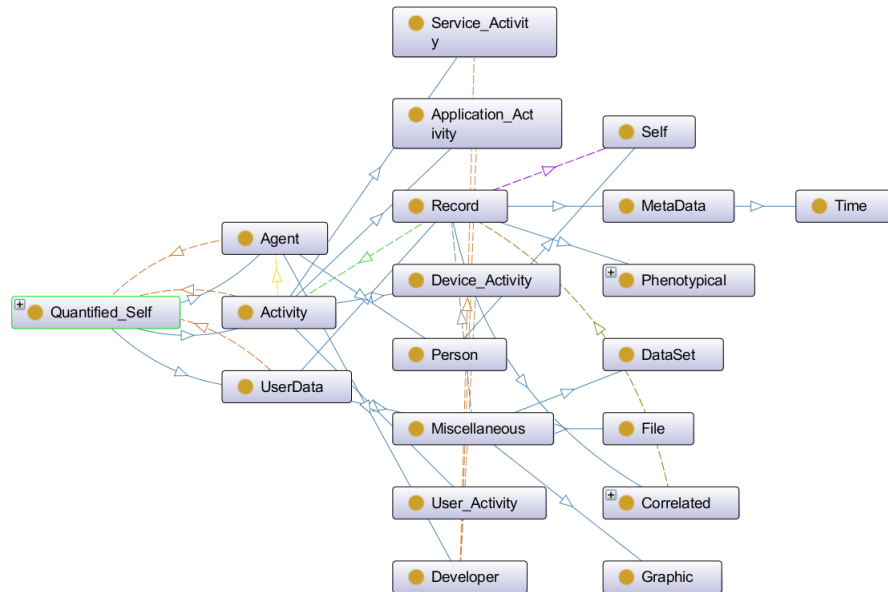


FIGURE 5: Quontology’s first class diagram

The extensions in PROV-DM's Agent class were the fewest and shall be elaborated first. As subclass of an Agent the **Developer** was created. The Developer differs from a Person or Organization in such a way, that he can be both. In the QS domain the Developer produces software or hardware for users and the difference if he's a person or organization is irrelevant. Additionally in QS the user signifies - in a semantic way - more than any normal person and thus the class **Self** - representing the user - was added as subclass of PROV-DM's Person.

A few more subclasses were added to Activity. The in subsection 3.1.1 mentioned zones led to four different subclasses of Activity: **User-**, **Device-**, **Application-** and **Service-Activity**. Although these activities could be structured into many fine granulated subclasses, it was decided to keep the ontology on a mostly abstract level at first and extend it when needed.

Lastly the Entity class had the most new semantics added. UserData is a subclass of Entity, that represents all data generated or related to the user. Since in QS exist various forms of data, two subclasses were defined: **Miscellaneous** - that represents all forms of data that are no original or real recorded data, e.g. **Graphics**, **DataSets** or abstract **Files** - and **Record** that represents real value data in form of Integer, Float, Double or String. Since there exists almost endless data that can be quantified by a user, thoughts were made on what aspects they share. Three subclasses were defined, that cover all areas in table 2 : **Phenotypical**, **Correlated** and **MetaData**.

Although quantification of a user’s genes may not be covered by this ontology yet, everything else observable from the user, can be categorized as **Phenotypical**, e.g. his weight, height or even psychological characteristics are phenotypical data. Even though many types of user data

could fit into this class, there exist real value data, which gives direct information measured by a sensor, and there exist fictive value data which mostly is correlated from multiple real value data, that gives information on a more abstract level, like the Body-Mass-Index, Waist-to-Hip-Ratio or Waist-to-Height-Ratio. All values, which are not generated through measurement, but a correlation of these, are classified as **Correlated**, meaning that every phenotypical value can be used to built correlated values but not vice versa. Lastly there are values that are not phenotypical, but also needed for correlated values or miscellaneous data, e.g. time, distance or duration. These sort of data does not fit neither in phenotypical nor in correlated data and thus are defined as MetaData.

This is just a first approach of defining some semantics on the generic PROV-DM. It should not be taken as finalized and complete.

3.4 The PROV-N Mapper

Although ProvStore has some useful functionalities and is suited for storing provenance documents, it needs online connectivity for use and stores the data in a non administrative environment, dedicate the data management, security and safety to others.

In order to store provenance documents locally, a suitable database is needed that can handle provenance documents and is executable on a local machine. As before mentioned, graph databases are suited well for provenance data and thus Neo4j shall be used as local provenance storage. To transform provenance documents into Neo4j's graph database, a mapping class in Python was developed in this project.

The class uses ProvStore's Python library [Huy15] to read provenance documents and build a provenance data structure out of it. Additionally it uses Python's py2neo library, to connect to a local Neo4j database.

PROV-N Mapper transforms each node and relation including their properties of the provenance data structure into Cypher queries, that create nodes and relations with their specific properties. Each query will be then added to an array. After the complete document is processed, a connection to the database is established and all queries are fired at once onto the database. Finally the connection will be closed. Now the provenance graph can be queried using Cypher, without the need of online connectivity and under complete control from the owner of the provenance document.

4 The Quantified Self Provenance Model - An approach

Due to the aforementioned arbitrarily complexity in QS workflows, which are not pre-definable, creating an abstract provenance model that could include every possible QS workflow was the main challenge in this project.

4.1 Commonalities in Quantified Self Workflows

To generate an abstract provenance model usable for any QS workflow, commonalities in as many workflows as possible were needed to be identified. Commonalities in QS workflows exist on three different layers: entities, activities and agents. Since the provenance of data flow needs to be captured, the most logical choice would be to identify similar activities in QS workflows.

But, since there exist endless activities as devices, software or user activities, abstract concepts of these QS workflow activities, their core functionalities, needed to be identified instead. In this project provenance of five different abstract functionalities were developed that exist in almost every QS workflow.

Input specifies the functionality of using raw data from the user - through interaction - or device - like a sensor - and generating structured data from it. Usually every QS workflow needs such a functionality to record data from the user.

Export specifies the functionality of exporting data from the current system into a format, readable for other computer systems or humans. Since a user usually uses more than one device, this functionality is processed multiple times, till the user see the visualized results.

Request is a functionality often seen in QS workflows, where the user stores his data on web servers. Often clients of these servers requests data regularly from other web services or applications to update their data set. This functionality can also be triggered by the user, if the system does not update the data automatically but through a button.

Another identified functionality is **aggregation**. A user usually records his data over time to interpret knowledge from it. This data needs to be aggregated, which is done by every software capable of storing and updating data sets. The aforementioned aggregation portals' purposes are aggregating heterogeneous data to create a deep knowledge about the user and help him understand his data.

Finally the functionality **visualization** is also part in most QS workflows. For many users who track themselves over a longer period, the visualization of their data over time in various graphics is a common approach to reveal knowledge of his data to him.

Even though there are many more common functionalities in QS workflows, the project's aim was not to create a complete universal provenance data model. As new use cases evolve, new functionalities are developed and extend the current provenance data model.

4.2 The Provenance Model

In this section the provenance model will be presented. Since the aforementioned impossible pre-definability of QS workflows, the functionalities will be elaborated separately and then explained why a complete model cannot be realized. Due to missing tools to visualize the provenance model in W3C's PROV notation, this section starts by introducing new graphics as replacement.



FIGURE 6: Components of the provenance data model

Figure 6 gives an overview of all components used in the provenance model. Each graphic is labeled as his replacement. The properties are added through a separate node to entities, activities, agents and relations like it is used to be in W3C's notation. Except these graphics, the provenance model does not differ from W3C's notation.

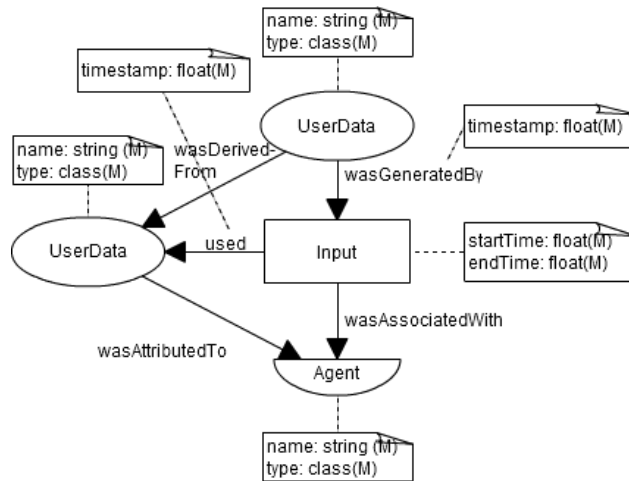


FIGURE 7: Provenance data model of the input functionality

Figure 7 shows the provenance data model of the input-functionality. Since the generated and used data depends on the context the user tracks something, both entities were modeled as UserData for which the names and types that describe their meaning needs to be recorded. Additionally a derivation relation will be recorded between the generated and used data. To determine the origin of the data (entered by user or measured by sensor) the input data shall be attributed to the responsible agent including its name and type. Also a record of the agent associated to the functionality needs to be created as relation between them. Timestamps will

be created and recorded each time when the data is used or generated. Lastly the start and end time of the functionality is recorded as part of the activity, like defined by W3C's Activity type.

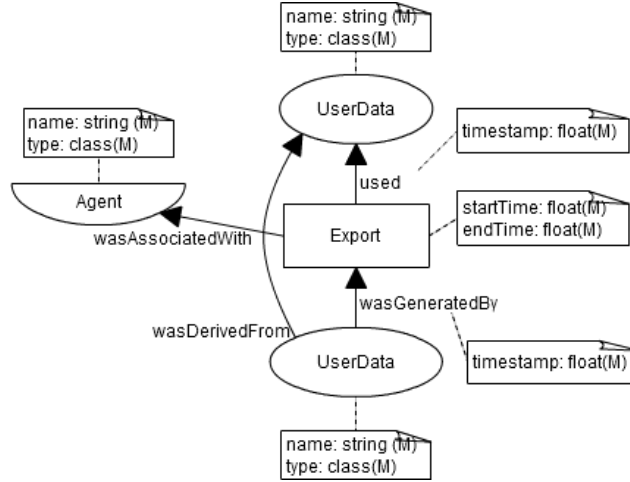


FIGURE 8: Provenance data model of the export functionality

The provenance model of the export-functionality is shown in figure 8. Since the used data has to be in the system, no attribution to an agent needs to be recorded. Again timestamps of used and generated data events are recorded together with the data as UserData. Also start and end time of the functionality and derivation of the data shall be recorded. Since export can be triggered by user or program, an agent will be associated to the activity and a relation will be recorded.

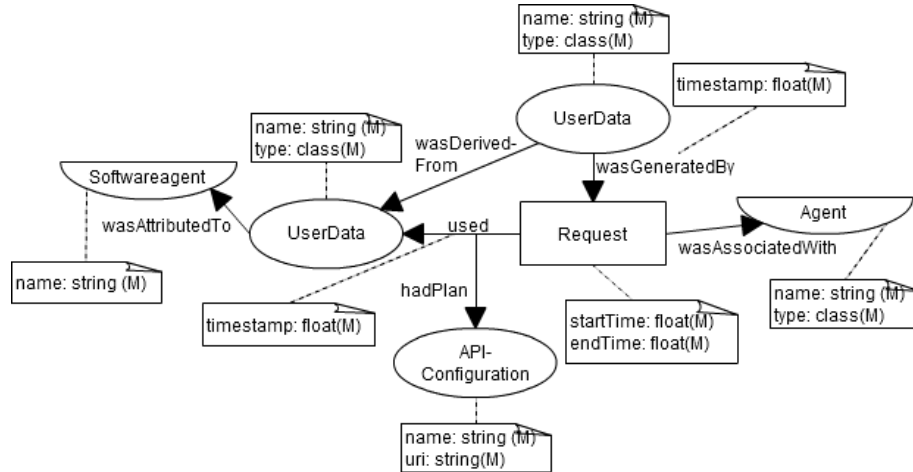


FIGURE 9: Provenance data model of the request functionality

Similar to the previous functionalities, the provenance model of the request-functionality, shown in figure 9, records the used and generated data, timestamps when these events happened, the associated agent who's responsible for this request and all aforementioned properties of these nodes. Additionally in this case, the originator of the used data will be recorded as Softwareagent including his name. And since the data that is used is stored on another system, the API

configuration is stored into the recorded used-relation, including the name of the server and its uniform resource identifier.

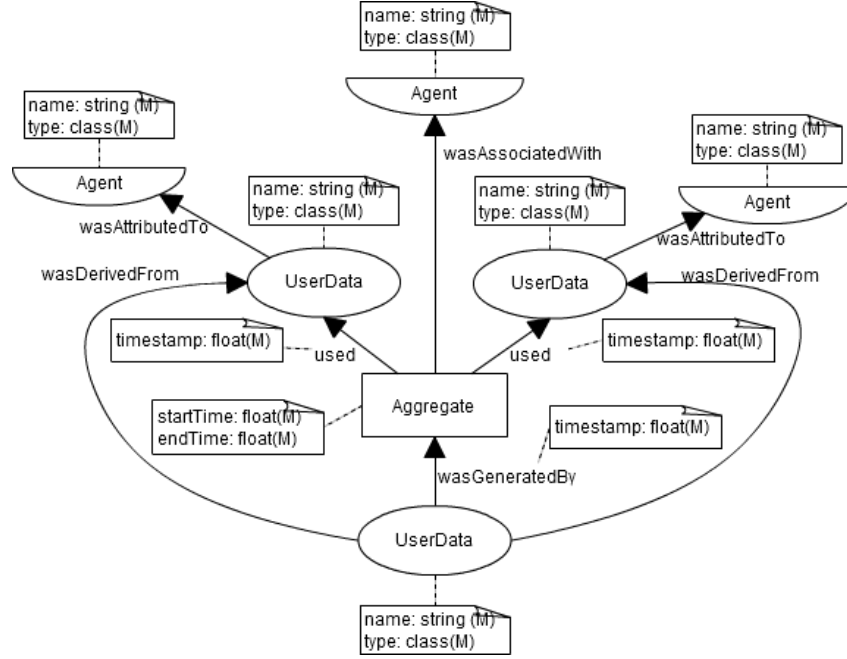


FIGURE 10: Provenance data model of the aggregate functionality

The provenance model of the aggregate-functionality, shown in figure 10, was one of the more complex models to define, because these aggregation portals allowing users to aggregate data from various different sources to create homogeneous and heterogeneous data sets. Hence the used and generated data needs to be captured as *UserData* entities. Since for aggregation are always at least two piece of data needed, two entities are used by the activity. In cases where aggregation of more than two entities occur, the model can be used recursively by using the generated data as input for the next aggregation process. Aggregation can be triggered by software that updates his data set periodically or by a user's interaction (e.g. pressing a button).

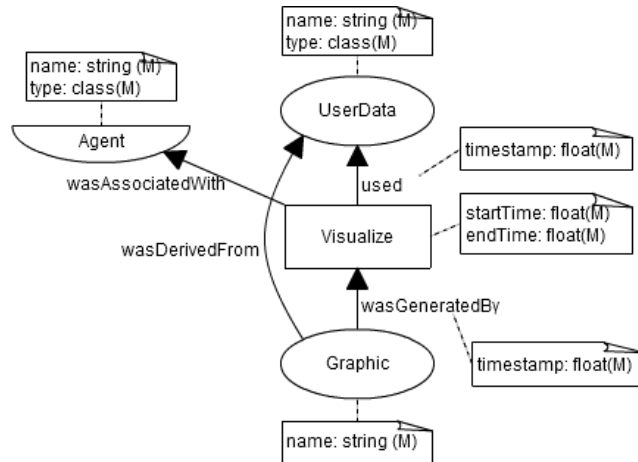


FIGURE 11: Provenance data model of the visualize functionality

Lastly the provenance model of the visualization-functionality is shown in figure 11. This functionality differs from the previous in this way, that the provenance of the generated entity, does not need to be recorded as *UserData*, but as a *Graphic* entity, including its name or caption. There could be many data additionally recorded for digital graphics, like size or resolution, but since a user could draw a graphic of his data on a paper, these properties could not be included into an universal provenance data model, usable for any QS workflow. Again all previous defined properties, in relations and entities shown in figure 11, are also captured.

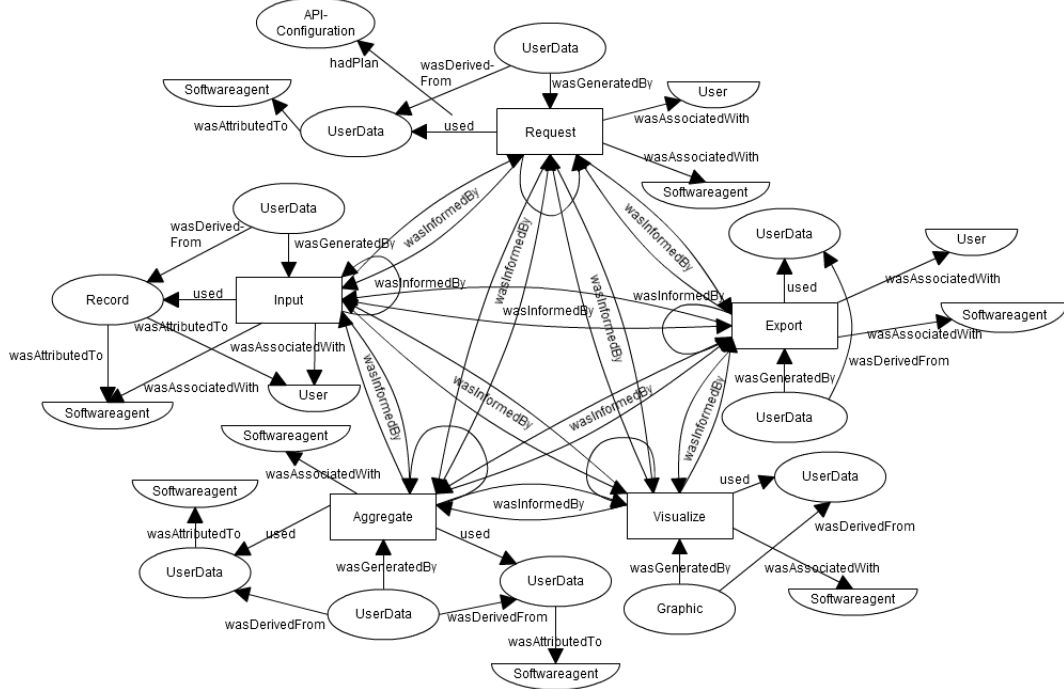


FIGURE 12: Complete “provenance data model”, without properties

Since the order of activities in a QS workflow is not pre-definable, no final provenance model could be realized. Figure 12 visualizes the problem, through a complete graph. In provenance graphs, multiple paths do not exist, because a piece of data has only a specific way it was created.

This contradiction between a complete universal provenance data model (that development was the target of this project), which describes the creation process and the unpredictability of QS workflows, results in a multi-relational “provenance model”, where every possible concatenation of functionalities is minded, which results in something else than a provenance graph.

Rather than developing a whole universal unpredictable provenance data model, it seems more useful to develop provenance data models of abstract functionalities and concatenate them during runtime and thus create a provenance graph on runtime.

5 A Quantified Self Provenance System - Some thoughts

Even though unable to finalize the project with a provenance system, this section is concerned with theoretical aspects of such a system, especially the approaches of capturing provenance data in QS. A provenance system can be described by four questions:

1. What do I want to capture? This question is answered through the provenance model, which defines the entities, activities, agents and relations to capture.
2. How can I capture the data? This questions is answered through a record system, which defines the architecture to capture provenance data.
3. Where do I store the data? This question is answered by the chosen provenance storage, previously mentioned in section 2.4 and 3.4.
4. How can I access the data? This question can be answered by a combination of the chosen storage system, its possibility to access the data and an optional representation format.

Since three of four questions were already answered through the previous sections, this section is focused on approaches of capturing provenance data in QS workflows. The manual approach, to capture all provenance data by the user, is not feasible enough and thus will be ignored as possible solution. Hence, there generally there exist two solutions to realize a provenance system in QS, either from the applications itself (disclosed) or by monitoring the communication between them (observed). Subsection 5.1 elaborates the disclosed approach, its pros and cons, the effort of realization and the quality of the results. In subsection 5.2 multiple observed approaches will be elaborated with the same issues as disclosed approaches.

5.1 Disclosed Provenance Systems - Requirements, Effort and Quality

A disclosed provenance system consists of provenance aware components. A provenance aware component can generate provenance data of his activities, entities and agents that are involved at runtime of the component, like a documentation about the execution, generated at runtime. In QS these components are devices, applications and services. But also the user needs to have the possibility of adding provenance into the system during or after the execution of his activities.

After execution, the components usually send their own provenance data onto a provenance storage, which will then be merged with provenance data of other components, according to the QS workflow. To realize such a provenance system, support of every participant in QS (user, developer and manufacturer) is needed, because there currently exist neither provenance aware components in QS nor a standardized provenance storage that can merge coherent provenance

data into a whole graph. To merge multiple provenance data from different sources, an identifier is needed, that can differentiate multiple user among the community and workflows from the same user. The merging itself can be realized through chronically relating the provenance subgraphs.

The most effort in disclosed provenance systems, at least in QS, would be convincing as many participants as possible to use a standardized provenance model to capture provenance data and sending it through an API, which is accessible through the web. Naturally such an API and its corresponding provenance storage and query mechanism needs to be developed first. Additionally the elaborated provenance model from the previous section needs to be extended for this purpose. To capture provenance from the user's activities, an application or web services could be offered that allows him to outline his activities and entities.

But the effort is worth it. Since disclosed provenance systems have access to their own code, provenance can be captured and a high level of granularity and semantic interpretation. This allows the generation of very accurate and trustful provenance graphs.

5.2 Observed Provenance Systems - An Introduction of multiple approaches

Since knowing from the previous subsection, that the current QS environment is missing provenance aware applications, observed provenance systems are the most promising approach to directly capture provenance data of the whole QS environment. One requirement of such a provenance system is the capability of observing every communication in the workflow, sending the inferred provenance data to a standardized API and merging them again into a whole provenance graph. Additionally the provenance system needs to observe and capture manual activities done by the user. Allen et al. wrote in [All+10] a practical guide on how to collect provenance data. They described five approaches:

1. Application reporting
2. Manual capture supported by task-specific editors
3. Operating system observation
4. Log file parsing
5. Observation at multi-system coordination points.

The first two approaches are already declined due to previous explanation. Operating system observation is also not suited, since devices and user activities do not have a modifiable operating system. Log file parsing would be sufficient, if every application and the user are using log files

and if there's access to it. Also it is difficult to infer valuable provenance data from log files only, since semantics are partially or completely missing. The last approach is observation at multi-system coordination points. Since the QS environment contains multiple systems and uses specific standard communication channels, e.g. Bluetooth, Wifi or USB, observing the coordination points could be a promising approach.

Such an observation could be realized through **Enterprise Services Buses** (ESB). Such an ESB is an abstraction on top of a messaging architecture, which allows multiple applications to be coordinated by routing and sequencing messages between those applications. ESB are often confused with service oriented architectures (SOA). While in SOA the services are decoupled and interact with each other, the ESB implements the message bus, so that these services can interact with each other. Since every message flows through the ESB, one could easily "fingerprint" every message's content to capture multiple use of the same data. This allows partial semantic interpretation.

The key advantage of this approach is, that they provide relatively fast and inexpensive, an integration of multiple independent systems into a single coherent whole. The main disadvantage of ESBs is that they require a messaging model, which wraps and handles the content that is passed through the bus and that is abstract enough to be used by every participating system, regardless of the implementation details.

Another approach to realize this observation lies in monitoring the communication points, capturing the communication protocol and inferring provenance data from it. This approach differs to ESB in such a way, that it even lesser intrude the QS environment. On the other hand it is not possible to "fingerprint" the messages so that no partial semantic interpretation of provenance data is possible. Naturally both approaches needs to support manual capture of the user's provenance data, which can be realized again through an application or web service.

Although these approaches are sufficient for capturing almost the complete QS environment, it will be difficult to get valuable provenance data through watching only at communication because the processes are still "black boxes". This means, that as long as the interface of a specific device, application or service is not changed, the provenance system cannot recognize if the process implementation has changed, resulting in completely lack of knowledge of the concrete data process and thus lesser accuracy and trust.

6 Approached Dead Ends

Although describing the way that led to the final provenance data model is important, several approaches in different stages that resulted in a dead end may be useful information for future work. In this section those dead ends will be elaborated and reasoned why they did not success.

6.1 Data Modeling: Bottom-Up

To generate a data model, a common approach - called “boottom-up” - to start with, is to collect information about single entities, cluster them together and repeat this process with the cluster iteratively, until an abstraction criterion is met. Since provenance of QS workflows is needed for this approach, this information could only be provided through the developer and manufacturer who have developed the devices, applications and services that a user is using. Hence, various developer were contacted and directly asked for help in this project by providing information about their data processing workflows. Unfortunately only a single developer was cooperative, through providing information about the collection, process and generation of data his application does. Since much more heterogeneous data is needed to generate an universal provenance data model, this approach failed due to uncooperativeness of the developer and manufacturer.

6.2 Representation of Quantified Self Workflows

To generate a provenance data model for QS workflows, these workflows needs to be defined first. Since a QS workflow is described through the user’s utilization of devices, applications and services, a representation format was needed, that allows the user to describe his workflow for other people. Before the visual representation was defined, multiple other approaches were tested and will be elaborated in this subsection. The first approach of describing a QS workflow, was through text. A user should simply write his QS workflow as text in an online-survey. This approach failed since every user could wrote the same QS workflow differently and thus make them incomparable.

Another approach was to structure the workflow through a schema, and thus make them comparable. The schema was implemented as an excel sheet that a user could fill out, structuring the workflows (by name, input and output), entities, activities, agents and their relations. Although comparable if an ontology is used, this approach was not user friendly, because it relied on keywords that the user needed to learn and kept much redundant information and thus was discarded.

7 Conclusions, Experience and Final Thoughts

This last section reflects the experiences, gained by working at the DLR and this project, the conclusions about provenance in QS and some personal thoughts about the future of this domain.

7.1 Quantified Self and Provenance - A field that needs to be re-searched

In this project multiple universal provenance data models of QS workflow functionalities were developed. Although new as a field of research, QS already compounds - from a software engineering perspective - high flexibility in data management and various possibilities of user interaction on multiple layers. This result was partly driven by the community, which is QS's backbone, that does not really care how something is implemented, as long as it works. The other part came from the various developer and manufacturer, who constantly develop new soft- and hardware and try to dominate the economy market in their area. This situation reminds one of a bazaar where many merchants are competing for the community's attention. This competitive situation exacerbates the establishment of well defined standards for QS data, resulting in a huge amount of different APIs and more work for developer.

Provenance on the other hand, even though existing already for several years as a field of research, is still elaborating standardized methods for creation of provenance data models and systems, like the well elaborated techniques used in context with relational databases. It may be advanced elaborated in specific domains, but not well explored in most new or little areas. Even so, the importance of provenance is increasing through all fields of research and due to this, future systems need to start using standardized provenance solutions for their specific domains and current systems need to be modified. Additionally, the advantages of standardized provenance approaches, needs to be realized in the future, e.g. by combining multiple provenance system from different domains.

Provenance in QS is something that should have been elaborated already years ago, since it processes highly personal and thus recognizable data of people. But this domain was left out mainly due to two reasons: competition and missing standards. Provenance reveals, through derivation of specific data, internal data management and thus information on how the company makes money. This was realized through communication with developer that did not want to share provenance information. Additionally, due to missing standards for provenance in QS, no complete capture of QS workflow's provenance data could be realized.

To introduce provenance in QS, a universal provenance management system - which uses universal provenance data models like the ones developed in this project - needs to be realized first, that

can capture provenance data from various different sources or formats and merging them into a whole provenance graph. Since the developer do not want to share information voluntarily, the community needs to be convinced to push developer and manufacturer in using the developed provenance management system in their devices, applications and services.

7.2 My work at the DLR

Well the heading says more than it actually was, because the DLR is such a huge enterprise, that I couldn't even see every building at headquarters' location. Even so the positive impressions I gained from SC were a very friendly and productive working atmosphere emitted by every employee, a high fluctuation on student internships with many different and interesting projects and the popularity of a kicker table in office jobs. On the contrary is just one negative impression I will remember, which is the missing payment of student projects or final theses. In my opinion a student should be paid for these works, because he's investing time and knowledge for the company's goal. But this arrangement differs between the Institutes and Facilities of the DLR and so this negative impression should not be reflected at the DLR.

Primary Literature

- [All+10] M. David Allen et al. “Provenance Capture and Use: A Practical Guide”, 2010.
- [Bar+14] Robin Barooah et al. “Quantified Self APIs and Data Flows”. In *Quantified Self Public Health Symposium*, April 2014, pages 41–44.
- [Hem15] Rolf Hempel. “Simulations- und Softwaretechnik: Statusbericht 2011–2015”. 2015.
- [Mor+08] Luc Moreau et al. “The Provenance of Electronic Data”. *Commun. ACM*, 51(4):52–58, April 2008. ISSN: 0001-0782. DOI: [10.1145/1330311.1330323](https://doi.org/10.1145/1330311.1330323). URL: <http://doi.acm.org/10.1145/1330311.1330323>.
- [NM01] Natalya F. Noy and Deborah L. McGuinness. “Ontology Development 101: A Guide to Creating Your First Ontology”. Online. 2001. URL: <http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html>.
- [Van14] Rik Van Bruggen. “*Learning Neo4j*”. Packt, Birmingham, 2014. ISBN: 978-1-84951-716-4.

Secondary Literature

- [Abr+13] Domingo De Abreu et al. “Choosing Between Graph Databases and RDF Engines for Consuming and Mining Linked Data”. In *Cold*. Volume 1034. In CEUR Workshop Proceedings. CEUR-WS.org, 2013. URL: <http://dblp.uni-trier.de/db/conf/semweb/cold2013.html#AbreuFPPPQSV13>.
- [Car+14] Lucian Carata et al. “A Primer on Provenance”. *Queue*, 12(3):10:10–10:23, March 2014. ISSN: 1542-7730. DOI: [10.1145/2602649.2602651](https://doi.org/10.1145/2602649.2602651). URL: <http://doi.acm.org/10.1145/2602649.2602651>.
- [Deu14] German Aerospace Center Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR). “The German Aerospace Center”. August 2014. URL: http://www.dlr.de/dlr/en/Portaldata/1/Resources/documents/2012_1/The_DLR_GB.pdf.
- [Gro15a] W3C Prov Working Group. “PROV Model Primer”. 2015. URL: <http://www.w3.org/TR/prov-primer/> (visited on 07/06/2015).
- [Gro15b] W3C Prov Working Group. “Provenance WG Wiki”. 2015. URL: http://www.w3.org/2011/prov/wiki/Main_Page (visited on 07/03/2015).
- [Gro15c] W3C Prov Working Group. “PROV-N: The Provenance Notation”. 2015. URL: <http://www.w3.org/TR/2013/REC-prov-n-20130430/> (visited on 07/07/2015).

- [HM14] Trung Dong Huynh and Luc Moreau. “ProvStore - a free provenance repository”. In *5th international provenance and annotation workshop*. In IPAW ’14. Cologne, Germany, June 2014. URL: <https://provenance.ecs.soton.ac.uk/store/> (visited on 07/07/2015).
- [HP13] Florian Holzschuher and René Peinl. “Performance of Graph Query Languages: Comparison of Cypher, Gremlin and Native Access in Neo4J”. In *Proceedings of the joint edbt/icdt 2013 workshops*. In EDBT ’13. ACM, Genoa, Italy, 2013, pages 195–204. ISBN: 978-1-4503-1599-9. DOI: [10.1145/2457317.2457351](https://doi.org/10.1145/2457317.2457351). URL: <http://doi.acm.org/10.1145/2457317.2457351>.
- [Huy15] Trung Dong Huynh. “prov 1.3.2 : Python Package Index”. 2015. URL: <https://pypi.python.org/pypi/prov> (visited on 07/07/2015).
- [Inc15a] Neo Technology Inc. “Capacity - The Neo4j Manual v2.2.3”. 2015. URL: <http://neo4j.com/docs/stable/capabilities-capacity.html> (visited on 07/07/2015).
- [Inc15b] Neo Technology Inc. “Neo4j, the World’s Leading Graph Database”. 2015. URL: <http://neo4j.com/> (visited on 07/07/2015).
- [Inc15c] Neo Technology Inc. “The Neo4j Graph Database - The Neo4j Manual v2.2.3”. 2015. URL: <http://neo4j.com/docs/stable/graphdb-neo4j.html> (visited on 07/07/2015).
- [Kar15] Vivek Karnataki. “NoSQL Databases – A Primer | Netwoven Blogs”. 2015. URL: <http://netwovenblogs.com/2013/09/08/nosql-databases-a-primer/> (visited on 07/07/2015).
- [Kel07] Kevin Kelly. “What is the Quantified Self?” October 2007. URL: <http://quantifiedself.com/2007/page/3/>.
- [Lab15a] Quantified Self Labs. “Quantified Self Conference Europe 2013”. 2015. URL: <http://quantifiedself.com/conference/Amsterdam-2013/> (visited on 07/08/2015).
- [Lab15b] Quantified Self Labs. “Quantified Self | Self Knowledge Through Numbers”. 2015. URL: <http://quantifiedself.com/> (visited on 07/08/2015).
- [LK15] CMU CREATE Lab and Candide Kemmler. “Fluxtream”. 2015. URL: <https://fluxtream.org/> (visited on 07/15/2015).
- [LLC15] Zenobase LLC. “Zenobase”. 2015. URL: <https://zenobase.com/#/> (visited on 07/15/2015).
- [Mee15] Meetup. “Find your people - Meetup”. 2015. URL: <http://www.meetup.com/> (visited on 07/08/2015).
- [NN15a] N.N. “Google Trends”. 2015. URL: <https://www.google.de/trends/explore?q=quantified%20self> (visited on 07/08/2015).
- [NN15b] N.N. “provenance - definition of provenance in Oxford dictionary (American English)”. 2015. URL: <http://www.oxforddictionaries.com/de/definition/englisch/provenance> (visited on 07/22/2015).

- [Rip+13] Margreet Riphagen et al. “LEARNING TOMORROW: VISUALISING STUDENT AND STAFF’S DAILY ACTIVITIES AND REFLECT ON IT”, November 2013.
- [Sim+05] Yogesh L. Simmhan et al. “A Survey of Data Provenance in e-Science”. *Sigmod rec.*, 34(3):31–36, September 2005. ISSN: 0163-5808. DOI: [10.1145/1084805.1084812](https://doi.org/10.1145/1084805.1084812). URL: <http://doi.acm.org/10.1145/1084805.1084812>.
- [TS15] Arne Tensfeldt and Dennis Singh. “Quantified Self: Selbstvermessung, Daten & Optimierung”. 2015. URL: <http://was-ist-quantified-self.de/#warum> (visited on 07/08/2015).
- [Wol15] Gary Wolf. “The quantified self | TED Talk”. 2015. URL: http://www.ted.com/talks/gary_wolf_the_quantified_self (visited on 07/08/2015).