Deutsches Zentrum für Luft- und Raumfahrt
Simulations- und Softwaretechnik

Hochschule Düsseldorf
University of Applied Sciences

Fachbereich Medien
Faculty of Media

VISUALIZATION OF THE PROVENANCE OF QUANTIFIED SELF DATA

Regina Struminski

592672

Master thesis for the course

MSc. Media Informatics

January 2017

Supervisors:

Prof. Dr. Markus Dahm

Prof. Dr. Holger Schmidt

Statutory Declaration

I hereby declare to have written this Master thesis on my own,
having used only the listed resources and tools.

____

Date, Signature

Contact details

Regina Struminski
Hammerweg 60
51399 Burscheid
regina.struminski@study.hs-duesseldorf.de

Abstract

The German Aerospace Center (*Deutsches Zentrum für Luft- und Raumfahrt, DLR*) is currently doing research in the field of data provenance. One specific subject of interest is the visualization of the provenance of Quantified-Self data in an easy-to-understand, self-explaining way, with the goal to communicate it to end users of Quantified-Self applications.

A review of existing works on provenance visualization showed that currently available visualizations are more suitable for expert audiences than for non-expert end users. However, the review also brought forth the idea to depict provenance in a comic-style manner: PROV COMICS. A large part of this work is devoted to the development of a visualization concept that determines a mapping of provenance data onto graphics, thus providing general guidelines for the comics. A prototypical implementation was built as proof of concept, demonstrating the automatic generation of comics from provenance documents stored at the PROVSTORE website.

Results from a small qualitative study suggest that a non-expert audience is generally able to understand the provenance of Quantified-Self data through PROV COMICS without any prior instruction or training.

Contents

1 Introduction

The German Aerospace Center (*Deutsches Zentrum für Luft- und Raumfahrt, DLR*) is currently doing research in the field of data provenance and has implemented the collection of provenance data into several applications, such as Quantified-Self apps. However, no custom visualization of that provenance data has been developed yet; existing visualization tools are used to view the collected data.

Available provenance visualizations all seem to be aimed at an expert audience, such as data analysts and provenance researchers. Most visualization tools display data provenance in the form of abstract graphs, sometimes very large in size and complex in structure. To an average end user of Quantified-Self apps, these graphs will often seem unintelligible and incomprehensible. However, being able to view the provenance of their own data – collected, processed, and stored by the app – would benefit the user.

Therefore a simple, readily understandable visualization of data provenance is needed. This task was addressed at the DLR's *Simulation and Software Technology* department and is described in the present work.

1.1 Background information

1.1.1 Data provenance

Data provenance (sometimes also referred to as *data lineage* or *data pedigree*) describes the life cycle of data, i.e. where and how it originated, by whom it has since been accessed, and whether or not it has been manipulated over time. It consists of "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness." (Moreau and Missier, 2013) In other words, provenance allows tracking the place and time something happens to the data as it undergoes different processes.

Looking at the available literature about data provenance, one can see that it is a relatively young field of research: Hardly any work on the subject is older than fifteen years. However, interest in this topic is growing as the amounts of data generated in scientific research, but also in commercial applications, increase. (Simmhan et al., 2005) The possibility to backtrace and verify data can be useful in different scenarios. For example, it may aid in locating errors in complex, possibly distributed applications, which are otherwise difficult to discover. It can also help to better assess the authenticity and trustworthiness of data.

Recording, modeling, and storing of data provenance have been increasingly studied in the past decade, which has led to the development of standardized notations like the Open Provenance Model (OPM) in 2007 and the W3C PROV Recommendations in 2010. (Moreau et al., 2008) (Gil et al., 2010)

1.1.2 Quantified Self

The Quantified Self movement has become very popular in recent years. Members of the community use special hard- and software to record, analyze, and evaluate personal data from their daily lives in order to gain insights and better self-knowledge.

The data may, for example, be related to health (e.g. blood pressure, insulin) or sports (e.g. step counter, GPS tracking while running), but also mental states (e.g. mood, arousal). The hardware often consists of so-called *wearables*, i.e. small sensors that are worn close to the body. These are usually accompanied by a computer or smartphone application which is able to receive, process, and store the collected data from the sensor.

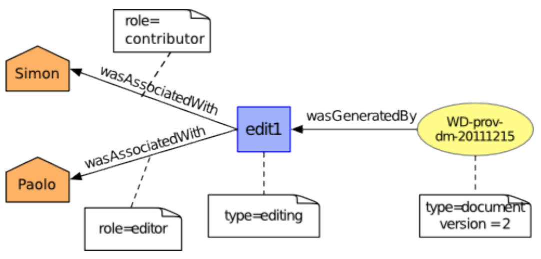
1.1.3 Provenance and Quantified Self

Typically, different devices, applications, and services are involved in the recording, processing, and storing of Quantified-Self data. For example, a sensor might send its collected data to a mobile app, which in turn uploads it to a cloud. Moreover, the app might perform various calculations on the data, visualize it in the form of a graphic, and also publish it to Twitter or Facebook, if desired. This way, new data may be generated from the original data at any time.

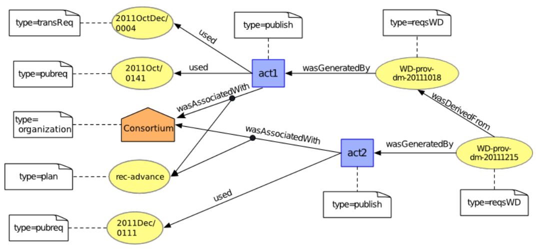
This raises the question of the reliability, accuracy, and trustworthiness of the new data that has been created by the various processing steps. By extensively recording the provenance during data processing, manipulations can be traced all the way back to the original data, thus allowing for a comparison and assessment of the accuracy and credibility of the (processed, transformed, or manipulated) data.

1.2 Motivation

The provenance of data is usually represented as a directed acyclic graph. In many visualizations, especially those following the W3C layout conventions[1], the graph is sorted topologically from left to right or top to bottom. Much like in a family tree, the "oldest" data can then be seen at the left or top and the "youngest", most recent data at the right or bottom. Two simple examples of such graphs are shown in Figure 1.1.



(a) A simple provenance graph.



(b) A slighty more complex provenance graph.

Figure 1.1: Examples of provenance graphs. (Moreau and Missier, 2013)

[1]https://www.w3.org/2011/prov/wiki/Diagrams

While these graphs may, to some extent, seem quite self-explaining to scientists, they can be rather hard to understand for laymen who are not usually concerned with graphs at all and have not been trained to read them.
Furthermore, provenance graphs can sometimes grow to enormous sizes, becoming so huge that even experts will have a hard time reading them. Since the span of immediate memory is limited to 7±2 entities at a time (Miller, 1956), graphs containing more than five to nine items will become gradually harder to interpret with every new item being added. However, 7±2 is a value that is easily reached and exceeded by even simple examples of provenance graphs (see Figure 1.2). The larger the graphs become, the more difficult it is to draw conclusions and derive new findings from the provenance data.

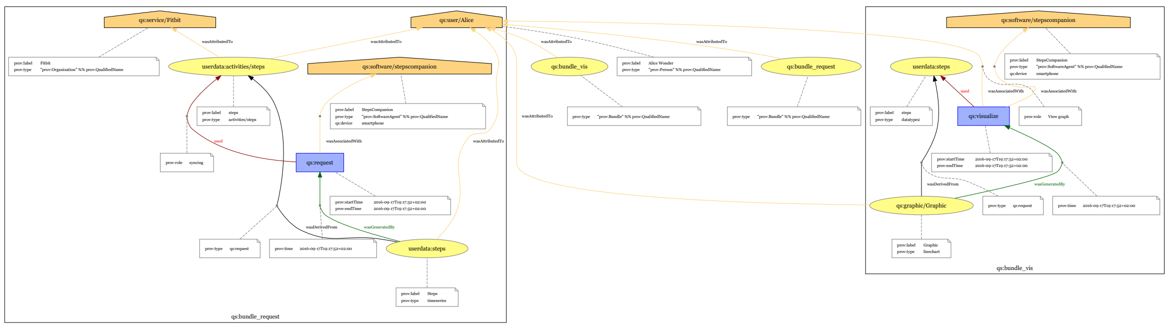


Figure 1.2: A larger, yet still rather simple provenance graph

The possibility to view their own provenance data is of no value to end users, if the visualization of that provenance is unintelligible to them. Moreover, the purpose of health-related smartphone apps and gadgets is to make accumulating, aggregating, reading and understanding health data easier, not harder. Many people use them in their leisure time, and even if they were interested in viewing the provenance of the data they have collected, it cannot be expected that they learn how to read an abstract, possibly complex graph. Instead, the visualization should be simple, self-explaining, and familiar in such a way that end users can read and understand it almost effortlessly.

For the abovementioned reasons, the useful and beneficial visualization of provenance data is a research topic in itself. As of now, abstract graphs are the only universal solution that can be applied equally to all provenance graphs. If a more customized visualization is desired, it must be developed individually for most applications in order to achieve the maximum benefit and information value.

1.3 Goals

The goal of this work was to develop a visualization of data provenance which is easy to understand and self-explaining in such a way that even non-experts – like end users of Quantified-Self apps – will be able to interpret it without any special training.

A fundamental part of the work was the mapping of structures found in provenance graphs onto visual elements. In the Quantified-Self context, these include:

- the visual appearance of the elements in the provenance graph,
- the graphical representation of temporal sequences,
- and the depiction of quantities or measurement readings (e.g. vital parameters), date and time information, etc.

The main focus was put on the concept and design of a clear and consistent visual language. A detailed style guide was developed, defining how certain "happenings" in the provenance graph should be represented graphically, which colors and shapes are to be used, etc. As proof of concept, an implementation was built in a prototypical manner, showcasing a number of working examples.

1.4 Outline

Chapter 2 describes the initial status in terms of capturing, storing, and displaying provenance in Quantified-Self applications at the DLR at the beginning of this work. It also states conditions and requirements that had to be followed while creating a new visualization.

The literature research in Chapter 3 provides an overview of existing solutions and approaches to display provenance graphs in an understandable, human-readable way.

Chapter 4 details the conception, design, and development of a new provenance visualization in the form of comics.

The design, execution, and evaluation of a reading study that was conducted for the provenance comics is described in Chapter 5.

Finally, conclusions are drawn and suggestions given for possible additions and future works in Chapter 6.

2 Initial Situation

2.1 Nominal-actual comparison

The DLR had already developed a provenance data model following the W3C PROV Recommendations (see section 2.2) and implemented the collection of provenance data into several Quantified-Self applications. However, there was no custom visualization of that provenance data yet.

To date, provenance data was stored and written to documents in accordance with the PROV-DM and PROV-N Recommendations (see section 2.2.1). The documents were then uploaded to the ProvStore[2] website, "a free service for storing, viewing and collaborating on provenance documents". (Huynh and Moreau, 2015) If visualizations of a provenance graph were needed, the functions offered by PROVSTORE were used. These include graphs like the one seen in Figure 1.2, but also Sankey, wheel, hive, and Gantt charts.

The desired condition was that there be a highly customized visualization that is readable and understandable even by laymen. The documents should still be uploaded to PROVSTORE; however, instead of the generic visualizations offered by PROVSTORE, a local program was to create the corresponding visualizations based on the information in these files. Ideally, the visualization program would fetch a PROV document from the PROVSTORE using the ProvStore API[3] and then generate the appropriate graphics locally on the client device. If possible, the program should be designed in such a way that it may later be integrated into Quantified-Self applications, so the user can view it from inside the app.

2.2 Organizational and technical conditions

2.2.1 The W3C PROV Recommendations

The whole concept and implementation had to comply with the W3C PROV Recommendations, which is a collection of definitions on both the representation and the exchange of provenance data. The heart of these definitions is the *PROV data model (PROV-DM)*[4], which mainly specifies how different types of actors (e.g. data, processes, persons, user agents) as well as their relationships towards each other (e.g. attributions, derivations, responsibilities) should be represented.

The W3C PROV model corresponds to an acyclic, directed graph like the ones already seen in Figure 1.1. The nodes of this graph are either *entities* (data), *activities* (processes), or *agents* (units responsible for activities and/or entities, e.g. persons or apps), while the edges represent their relationships. The most important concepts of the PROV Recommendations that play a role in this work are described below.

[2]https://provenance.ecs.soton.ac.uk/store/
[3]https://provenance.ecs.soton.ac.uk/store/help/api/
[4]*https://www.w3.org/TR/prov-dm/*

Entity

“An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary.” Entities are “things we want to describe the provenance of”, which covers “a broad diversity of notions, including digital objects such as a file or web page, physical things such as a mountain, a building, a printed book, or a car as well as abstract concepts and ideas.” (Moreau and Missier, 2013)

Activity

“An activity is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities.” Like entities, “activities can cover a broad range of notions: information processing activities may for example move, copy, or duplicate digital entities; physical activities can include driving a car between two locations or printing a book.” (Moreau and Missier, 2013)

Agent

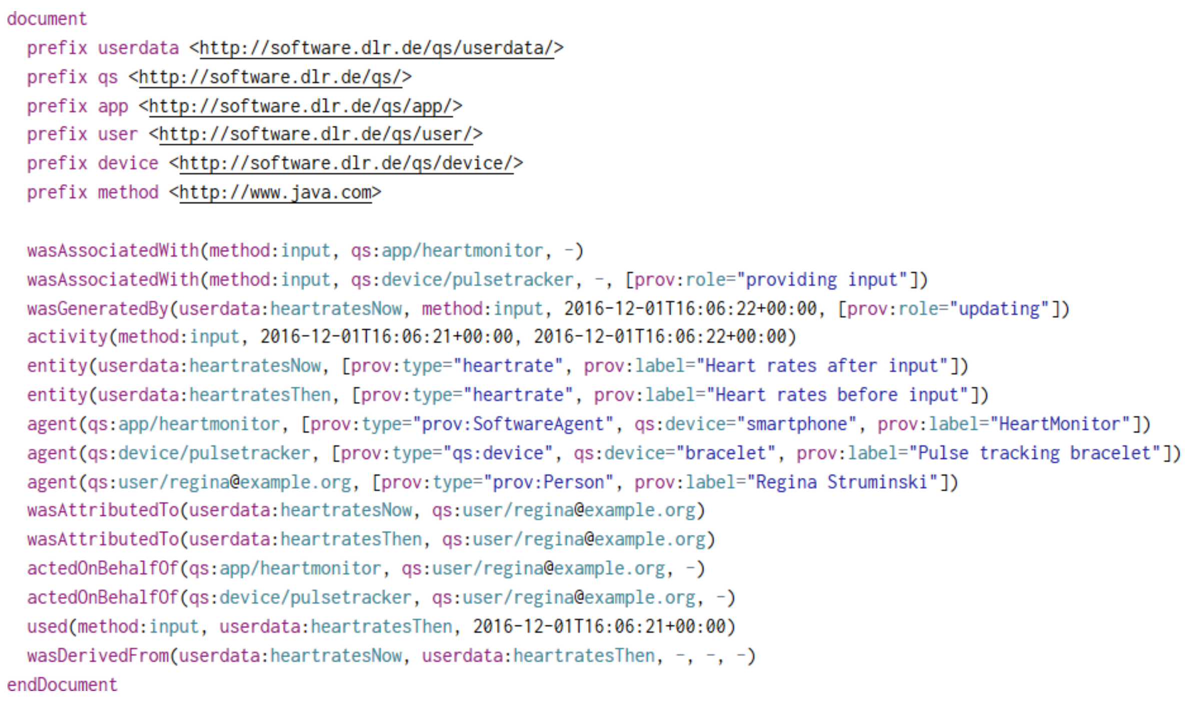
“An agent is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent’s activity. [...] Agents can be related to entities, activities, and other agents” and “are defined as having some kind of responsibility for activities.” (Moreau and Missier, 2013)

Relationships

Several types of relationships exist between agents, activities, and entities. The ones relevant in this work are listed below.

- “An activity **association** is an assignment of responsibility to an agent for an activity, indicating that the agent had a role in the activity.” (Moreau and Missier, 2013)
- “**Attribution** is the ascribing of an entity to an agent.” (Moreau and Missier, 2013)
- “A **derivation** is a transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity.” (Moreau and Missier, 2013)
- “**Generation** is the completion of production of a new entity by an activity. This entity did not exist before generation and becomes available for usage after this generation.” (Moreau and Missier, 2013)
- “**Usage** is the beginning of utilizing an entity by an activity. Before usage, the activity had not begun to utilize this entity and could not have been affected by the entity.” (Moreau and Missier, 2013)

Other useful definitions by the W3C concern "serializations and other supporting definitions to enable the inter-operable interchange of provenance information in heterogeneous environments such as the Web." (Groth and Moreau, 2013) These include the human-readable *PROV Notation*[5] (*PROV-N*, see Figure 2.1) and the *PROV-XML*[6] schema (see appendix A for the same graph in XML notation). A suggestion for JSON serialization *(PROV-JSON*[7]*)* has also been submitted. A number of further recommendations exist in PROV; however, they do not play a particular role in this work.



```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  wasAssociatedWith(method:input, qs:app/heartmonitor, -)
  wasAssociatedWith(method:input, qs:device/pulsetracker, -, [prov:role="providing input"])
  wasGeneratedBy(userdata:heartratesNow, method:input, 2016-12-01T16:06:22+00:00, [prov:role="updating"])
  activity(method:input, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  entity(userdata:heartratesNow, [prov:type="heartrate", prov:label="Heart rates after input"])
  entity(userdata:heartratesThen, [prov:type="heartrate", prov:label="Heart rates before input"])
  agent(qs:app/heartmonitor, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="HeartMonitor"])
  agent(qs:device/pulsetracker, [prov:type="qs:device", qs:device="bracelet", prov:label="Pulse tracking bracelet"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  wasAttributedTo(userdata:heartratesNow, qs:user/regina@example.org)
  wasAttributedTo(userdata:heartratesThen, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/heartmonitor, qs:user/regina@example.org, -)
  actedOnBehalfOf(qs:device/pulsetracker, qs:user/regina@example.org, -)
  used(method:input, userdata:heartratesThen, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:heartratesNow, userdata:heartratesThen, -, -, -)
endDocument
```

Figure 2.1: Example provenance document in PROV Notation

2.2.2 The Quantified-Self PROV model

A PROV model following the W3C standards had already been defined in a previous work at the DLR. (Schreiber, 2016) This model particularly contains five sub-models for use cases that regularly occur in the context of Quantified Self applications: *Input*, *Export*, *Aggregate*, *Request*, and *Visualize*. This model was to be retained; however, it made no claims of being complete. Adjustments could be made if more or different data was needed in order to create a useful and consistent visualization. The model is introduced in the following paragraphs.

[5] *https://www.w3.org/TR/prov-n/*
[6] *https://www.w3.org/TR/prov-xml/*
[7] *https://www.w3.org/Submission/prov-json/*

Input

The *Input* sub-model describes the act of obtaining new data from the user and generating structured data from it (see Figure 2.2). The input may either come from devices with sensors or from the user typing in data manually. For example, one user might have a bodyweight scale with WiFi functionality which is able to communicate a person's current weight to a smartphone app automatically after weighing, while a different user only owns a conventional weight scale and thus enters his weight by hand after weighing. In general, every Quantified-self workflow needs an input functionality in order to record data from the user.

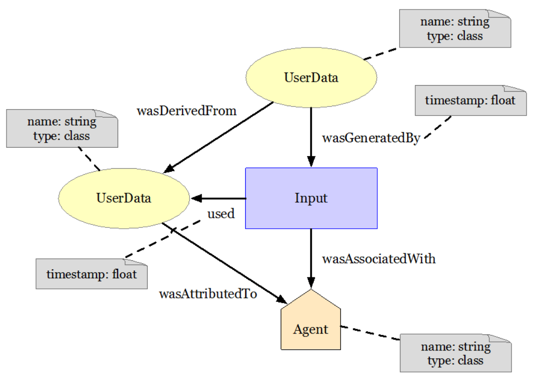


Figure 2.2: Input sub-model

Export

The *Export* sub-model specifies the functionality of converting structured data from the current system into a different format that is readable by other applications, computer systems, or devices (see Figure 2.3). Users often use more than one device; for example, a user might collect data about the number of steps taken on his smartphone and then use an export function to create a copy of his data in an Excel format like `.xlsx`.

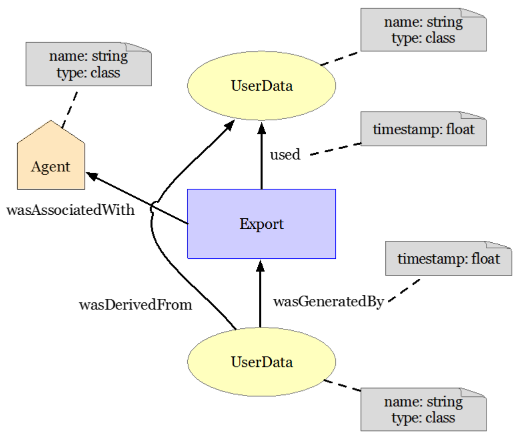


Figure 2.3: Export sub-model

Aggregate

An aggregation is the creation of new data from two or more sources of existing data. This process is covered by the *Aggregate* sub-model (see Figure 2.4). Aggregations are usually done by software, like desktop or mobile applications. They help users to a deeper knowledge and better understanding of their data and allow them to gain new insights from it. For example, users might track their daily caffeine intake and also measure their blood pressure several times a day. An aggregation would then allow them to discover correlations and find out whether or not their caffeine intake has a measurable effect on their blood pressure values.

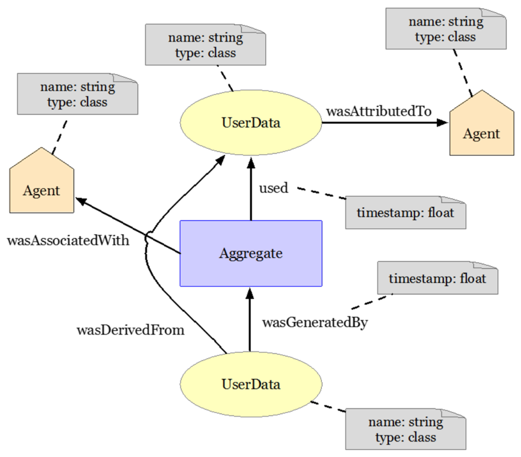


Figure 2.4: Aggregate

Request

To most Quantified-Self applications it is essential to have a request functionality similar to the one described in the *Request* sub-model (see Figure 2.5). Clients may regularly, or on user's demand, send polls for changes to a server, push newly created data to it or pull existing data from it. For example, in many applications, data is not only stored on one device (e.g. a smartphone), but also uploaded to a server ("cloud") or other web services. The user may then change the device (e.g. switch to a desktop computer) and download the same data there. Moreover, cloud synchronization also serves as a protection against loss of data, for example when the application device is lost or broken.

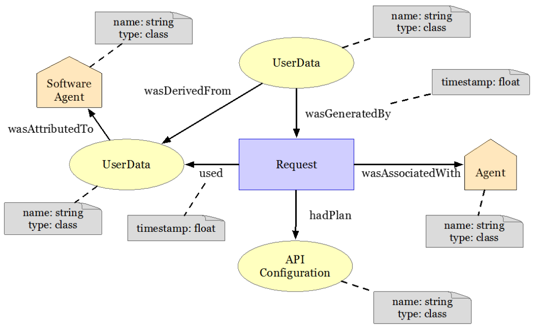


Figure 2.5: Request

Visualize

As a matter of course, Quantified-Self applications are generally capable of visualizing the collected data. Without that, it would be rather difficult for the user to gain any insights from his data. Therefore the Quantified-Self model also includes a *Visualize* sub-model (see Figure 2.6). Visualization allows users to view their data, which has often been tracked over a longer period of time, in various types of graphics. For example, a user who has collected a year's worth of bodyweight data might view the development of his weight as a weight-time graph for the last three, six, or twelve months. There might also be other graphics such as the user's position on a BMI (body mass index) scale, or a pie chart depicting his body's composition of fat and lean body mass.

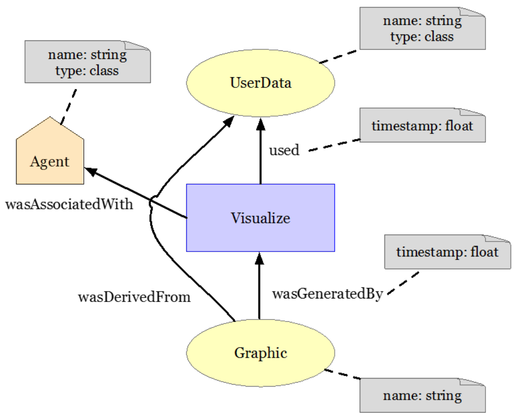


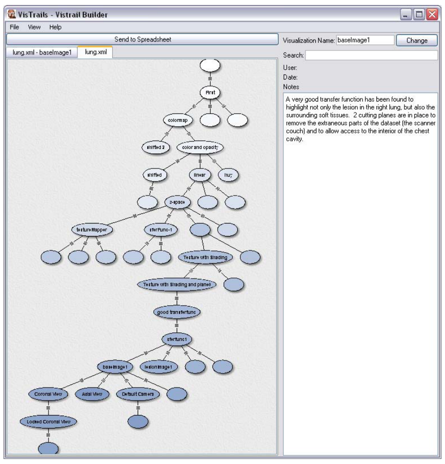
Figure 2.6: Visualize

3 Survey of existing solutions

A literature research was performed to obtain an overview over existing provenance visualizations and gather ideas and inspirations for a solution aimed at end users. The focus was put on works that explicitly pursued the goal of creating well-readable and understandable provenance visualizations; however, some interesting approaches could also be found in works that do not directly answer to that description.

3.1 Existing tools for provenance visualization

Macko and Seltzer presented PROVENANCE MAP ORBITER, a tool that simplifies large provenance graphs by combining sub-nodes into "summary nodes" at the beginning. (Macko, 2011) The user can then interactively explore the graph by zooming into these nodes. Other tools that work in a similar way include PROVENANCE BROWSER (ANAND ET AL., 2010), ZOOM*USERVIEWS (BITON ET AL., 2008), and PROVENANCE EXPLORER (HUNTER AND CHEUNG, 2007).



Visualization Name: baseImage1
Search:
User:
Date:
Notes
A very good transfer function has been found to highlight not only the lesion in the right lung, but also the surrounding soft tissues. 2 cutting planes are in place to remove the extraneous parts of the dataset (the scanner couch) and to allow access to the interior of the chest cavity.

Figure 3.1: Screenshot of VISTRAILS

VISTRAILS by Freire et al. uses a tree representation somewhat similar to a node-link diagram to visualize workflows (see Figure 3.1). Like the aforementioned tools, it can hide certain nodes to reduce visual clutter. It also makes use of color saturation to indicate the chronological order of nodes and provides a "diff interface" to help comparing different workflows. (Freire et al., 2006)
PEDIGREE GRAPH by Myers et al. also uses a tree graph to visualize data provenance. (Myers et al., 2003) To reduce visual clutter, the depth of the tree is limited to five, and nodes are visible or hidden depending on which node is currently focused.

Del Rio and Pinheiro da Silva developed PROBE-IT!, a tool for visualizing provenance from geographical information system (GIS) applications. (Rio and al, 2007) Their goal was to help scientists judge the quality of maps as well as trace and explain the reasons for possible defects or imperfections. Like many current visualization tools, PROBE-IT! displays the different processes and their connections in the form of a directed acyclic graph, similar to a node-link diagram. However, mouse-clicking one of the nodes will open a popup window containing that process's results (e.g. a map). This is to say that PROBE-IT! does not only visualize the provenance itself, but also the actual data that resulted from process executions.
A study conducted by the authors showed that scientists were better able to identify and explain several types of defects in the maps when provenance visualization was available. Improvements were particularly notable in the group of non-GIS-experts.

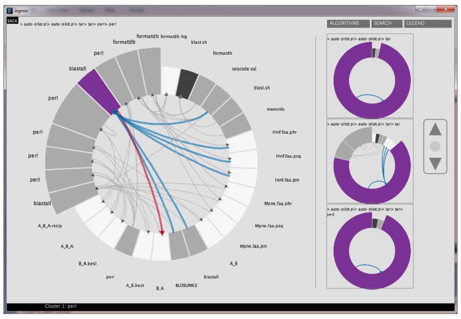


Figure 3.2: Screenshot of InProv

Borkin et. al. presented InProv, a tool that employs an interactive radial-based tree layout to display file system provenance (see Figure 3.2). It also features time-based grouping of nodes, which allows users to examine a selection of nodes from a certain period of time only. (Borkin et al., 2013)
A quantitative evaluation comparing InProv to the node-link based Provenance Map Orbiter showed that participants found InProv easier to use and were able to identify system activity more accurately.
On an interesting note, the evaluation also revealed that "women had a significantly lower average accuracy (56%) compared to men (70%)" when using Provenance Map Orbiter, while with InProv, both genders performed similarly well.

Chen et al. visualized the provenance of big and complex data, focusing on both exploration and explanation of provenance. (Chen et al., 2012) They designed several different graph representations, such as a hierarchical layout and a history chain. The visualization allows the user to define his own mappings of data attributes onto visual properties, and it uses incremental loading to deal with the large size of the data. Clustering techniques and elimination of processes and artifacts are used to simplify the graph, and there is a possibility to compare provenance graphs with each other.

3.2 Other related work

Richardson and Moreau developed the PROVglish architecture for generating natural language sentences from provenance graphs. Their goal was to create textual explanations that are suitable to communicate provenance data to end users. (Richardson and Moreau, 2016) Especially notable is the application-generic, domain-agnostic approach, meaning that PROVglish can not only handle certain applications with specific data models, but is in fact applicable to any kind of provenance data following the W3C PROV Recommendations. This is achieved by "exploiting the linguistic information informally encoded in the URIs denoting provenance resources" in RDF documents. (Richardson and Moreau, 2016)

Bach et al. designed Graph Comics to present and explain temporal changes in dynamic networks to both expert and non-expert audiences. (Bach et al., 2016) Based on traditional node-link diagrams, which are commonly used to visualize static networks, they added a temporal dimension by creating series of panels to illustrate how a dynamic network changes over time.
A subsequent qualitative study suggested that "a general audience is quickly able to understand complex temporal changes through graph comics, provided with minimal textual annotations and no training". (Bach et al., 2016)

3.3 Result

A majority of tools found in literature visualize provenance graphs using ordinary node-link diagrams or tree representations similar to node-link diagrams. This is probably owed to the fact that most of these visualization tools are aimed at an expert audience, like scientists, researchers, data analysts, and provenance experts in general. To the average end user of Quantified-Self applications, however, these diagrams may often be difficult to understand or even incomprehensible.

In terms of visualization, the only work that stood out was InProv, whose novel approach using radial layouts differs largely from the usual node-link diagrams. However, the graphics that InProv generates are still very abstract and graph-like, rendering them suitable for experts rather than for end users.

At the time of writing, there seemed to be no self-explaining, easy-to-understand visualizations of provenance data that were targeted at non-experts. One tool aimed at end users was PROVglish; however, it only generates textualizations, not graphical visualizations.

4 Prov Comics

4.1 Selected solutions

From the survey of current literature in Chapter 3, the works that were explicitly dedicated to the visualization of data provenance did not yield much inspiration for the creation of a simple, self-explaining provenance visualization directed towards end users. Of all the works, just the two that were *not* directly related to the visualization of provenance led to a new idea.

The first one was PROVglish by Richardson and Moreau. (Richardson and Moreau, 2016) The textual explanations generated by PROVglish were actually quite close to the objective of this work. The difference was only in the medium: While PROVglish uses language to create easily-understandable representations of data provenance, this work was aimed at developing graphical representations. In simplified terms, one might employ similar algorithms as PROVglish and just have them output pictures instead of words.
However, since creating an application-generic graphical visualization would have been a lot more difficult than creating application-generic sentences, the visualizations developed in this work only consider provenance data from the Quantified-Self domain. (Further details on the problems offered by a domain-agnostic approach are given in section 6.2.3.)

The second inspiration was drawn from the Graph Comics by Bach et al. (Bach et al., 2016) Just like the changes in a network, provenance too has a temporal aspect: origin, manipulation, transformation, etc. happen over time and can thus be narrated like a story. For example, when looking at the provenance of a graphic that was created by some application, one might find that *first* (i.e. near the root of the provenance tree), a sensor recorded some data; *later*, the data was exported to another file format; and *after that*, a graphic was generated based on this exported file. The directed, acyclic provenance graph guarantees that, while moving through its nodes, one always moves linearly forward or backward in time. It should thus be possible to derive a story line from the graph and create a sequence of pictures for it.

A general advantage of comics over conventional visualizations, like node-link diagrams, is their familiarity: Almost anybody has probably read some comics in their life. No training is required to read them, and they are able to transport meaning with minimal textual annotation. They are easy to interpret and not as strenuous to read as, for example, a graph or a long paragraph of continuous text.

These two approaches in combination led to the idea of a comic-style visualization for provenance data. Similarly to PROVGLISH, provenance documents may be searched for agents, entities, and activities, their relations towards each other, as well as other descriptive keywords (e.g. attribute values) specified by the Quantified-Self model. Instead of outputting the results as text, they can be used to generate a number of comic panels containing predefined and pre-rendered graphics that convey the sequence of "happenings" in the provenance graph in a story-like manner.
The conception, design, and development of these PROV COMICS is described in the next sections.

4.2 Concept

In order to create clear, unambiguous graphics, and also to later generate the comics automatically, a consistent mapping of PROV elements onto visual representations needed to be designed beforehand. Also, a predefined sequence of comic panels had to be assigned to each of the five basic use cases (i.e. the sub-models explained in section 2.2.2). This would later allow for a script to simply "translate" the provenance data into corresponding pictures.

A lot of thought was put into readability and usability aspects in order to make the comics as accessible and comprehensible as possible. In terms of readability, readers should quickly and effortlessly be able to recognize and identify the different actors in the story. No cognitive capacities should be unnecessarily "wasted" trying to decipher the visual language. While a "conventional", entertaining comic may invite a reader to delve into its pictures and admire their details, the main goal of the PROV COMICS is to quickly convey a message. This also had an impact on some of the decisions made about the graphical representations: All pictures are rather simplistic, minimalist, and low in detail. A "flat" design was employed, meaning that color gradients and other decorative elements (e.g. borders) were reduced to an absolute minimum. The less there is to be seen in the pictures, and the fewer distractions they contain, the easier it is for viewers to focus their attention on the important parts that drive the story forward.

As for usability, and especially the choice of colors and shapes, it had to be kept in mind during the design phase that the comics would later be displayed on a monitor (e.g. a computer or smartphone screen) and possibly also viewed by readers with color vision deficiencies. All graphics still needed to be well discernible and distinguishable under these conditions. When it came to choosing colors and contrasts, the WEB CONTENT ACCESSIBILITY GUIDELINES (WCAG)[8] and their *Success Criteria (SC)* served as a general guideline.

For the abovementioned reasons, it has been made sure that objects are never only distinguishable by one single feature. For example, as required by the WCAG SC 1.4.1 (USE OF COLOR) (Cooper et al., 2016b), two items never differ from each other only in their color.

[8] *https://www.w3.org/TR/WCAG20/*

Rather, all important actors in the comics exhibit three distinctive features: **shape**, **color**, and **icons or texts**. These three features, as well as a few other design considerations, are described in the following sections.

4.2.1 Shapes

Shapes have been designed and selected according to several criteria. As mentioned above, shapes must not exhibit a lot of detail. They should use a “flat” look, i.e. avoid textures, decorations, shadows, and three-dimensional elements.

Table 4.1 gives an overview over the shapes that have been selected to reflect the different types of elements in the DLR’s PROV data model. Note that activities are not directly listed here. Unlike agents or entities, activities are actions that take place over time. Thus they will not be depicted as a single graphic; instead, they represent a temporal progress and only become visible through the sequence of events in the next three to five panels of the comic.

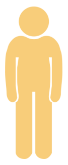
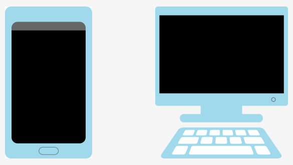
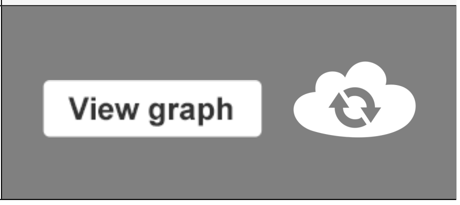
Element type	**Shape**	**Example**
Agent type: `Person`	human silhouette	
Agent type: `SoftwareAgent`	smartphone, computer, ... *(depending on the agent’s ”device” attribute)*	
Agent type: `Organization`	office building	
Entity	file folder, document, chart, ... *(depending on the entity’s “type” attribute)*	
Activity-related objects	button, icon, ... *(depending on the activity’s name and/or “role” attribute)*	

Table 4.1: Shapes defined for different types of PROV elements

4.2.2 Icons, letters, and labels

As a second distinctive feature, all main actors in the comics carry some kind of symbol on them, whether it be an icon, a single letter, or a whole word (see Figure 4.1).

- Agents of the `Person` type always wear the first letter of their name on the chest.
- `Organization` agents have their name written at the top of the office building picture.
- `SoftwareAgents` display an application name on the screen.
- Entities are always marked by an icon representing the type of data they contain.

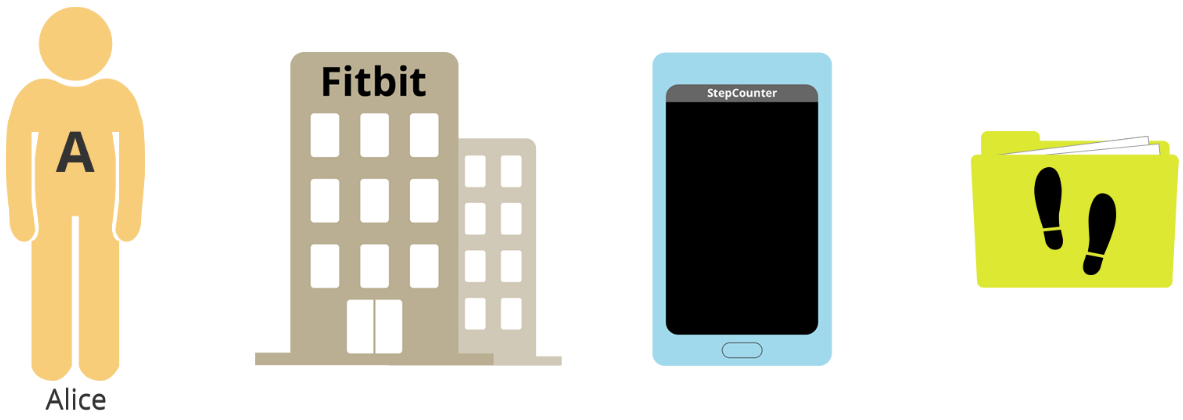


Figure 4.1: Agents and entities using three distinctive features (shape, color, and icon or text)

A few icons have been defined for some types of data that are common in the Quantified-Self domain. These are listed in Table 4.2.

Data type	Icon	Description
Blood pressure	♡	a heart outline with a pressure indicator
Heart rate	♥	a heart containing an ECG wave
Sleep	☾★	a crescent moon with stars
Steps	👣	a pair of footprints
Weight	kg	a weight with the abbreviation “kg” cut out

Table 4.2: Icons for some typical Quantified-Self data types

4.2.3 Colors

Colors for objects of the same type

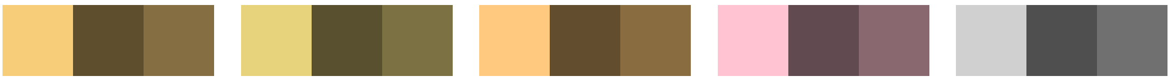
Two alternative color shades have been defined for agents and entities in case that two or three objects of the same type need to appear at once. At the moment, this never actually happens in the current implementations and example PROV files; but since it may someday be the case, these alternative shades are suggested.

The first alternative was determined by reducing the main color’s lightness (in the HSL color space) by 60%, the second alternative by reducing the lightness by 30%-45%. Table 4.3 shows all colors defined for agents and entities along with their two alternatives, while Figures 4.2, 4.3, and 4.4 examplarily simulate the effect of different types of color blindness on agent and entity colors.[9]

[9]Simulations generated by http://www.color-blindness.com/coblis-color-blindness-simulator/

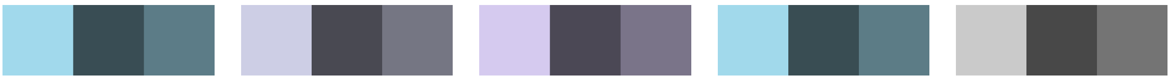
	Color		Example
	Agent (type: `Person`)		
Default color	`hsl(40°, 51%, 97%)` `lab(85, 8, 47)`	`rgb(248, 205, 122)` `#f8cd7a`	🟨
Alternative 1	`hsl(40°, 51%, 37%)` `lab(34, 3, 21)`	`rgb(94, 78, 46)` `#5e4e2e`	🟫
Alternative 2	`hsl(40°, 51%, 52%)` `lab(48, 5, 29)`	`rgb(133, 110, 65)` `#856e41`	🟫
	Agent (type: `SoftwareAgent`)		
Default color	`hsl(195°, 32%, 93%)` `lab(83, -15, -15)`	`rgb(161, 217, 236)` `#a1d9ec`	🟦
Alternative 1	`hsl(195°, 32%, 33%)` `lab(31, -7, -7)`	`rgb(57, 77, 84)` `#394d54`	⬛
Alternative 2	`hsl(195°, 32%, 53%)` `lab(50, -10, -10)`	`rgb(92, 124, 135)` `#5c7c87`	🟦
	Agent (type: `Organization`)		
Default color	`hsl(43°, 21%, 73%)` `lab(72, 1, 16)`	`rgb(186, 175, 147)` `#baaf93`	⬜
	`hsl(42°, 12%, 82%)` `lab(81, 1, 10)`	`rgb(209, 201, 183)` `#d1c9b7`	⬜
Alternative 1	`hsl(43°, 21%, 13%)` `lab(12, 0, 4)`	`rgb(33, 31, 26)` `#211f1a`	⬛
	`hsl(42°, 12%, 22%)` `lab(23, 0, 3)`	`rgb(56, 54, 49)` `#383631`	⬛
Alternative 2	`hsl(43°, 21%, 43%)` `lab(44, 1, 10)`	`rgb(110, 103, 87)` `#6e6757`	🟫
	`hsl(42°, 12%, 52%)` `lab(54, 0, 7)`	`rgb(133, 128, 117)` `#858075`	🟫
	Entity		
Default color	`hsl(64°, 79%, 91%)` `lab(89, -19, 79)`	`rgb(220, 232, 49)` `#dce831`	🟨
Alternative 1	`hsl(64°, 79%, 31%)` `lab(32, -8, 34)`	`rgb(75, 79, 17)` `#4b4f11`	🟩
Alternative 2	`hsl(64°, 79%, 51%)` `lab(52, -12, 50)`	`rgb(123, 130, 27)` `#7b821b`	🟩

Table 4.3: Determining alternative color shades for entities



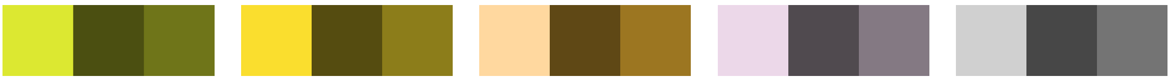
(a) Normal (b) Protanopia (c) Deuteranopia (d) Tritanopia (e) Achromatopsia

Figure 4.2: `Person` agent color shades and how they are seen by colorblind people



(a) Normal (b) Protanopia (c) Deuteranopia (d) Tritanopia (e) Achromatopsia

Figure 4.3: `SoftwareAgent` color shades and how they are seen by colorblind people



(a) Normal (b) Protanopia (c) Deuteranopia (d) Tritanopia (e) Achromatopsia

Figure 4.4: Entity color shades and how they are seen by colorblind people

In a previous approach, colors had been rotated by 180°, 90°, and 270° to obtain well-matched second, third and even fourth colors. However, two problems arose: First of all, the whole comic would generally have become very colorful, which would possibly have led to confusion. Depending on the situation, there might, for example, have been a blue person that owns a blue phone and a pink entity, while at the same time a pink person is present owning a blue entity. Some similar items would have had very dissimilar colors, while some dissimilar items would have had very similar colors. Apart from causing a certain visual inconsistency, this might also have suggested to the reader that there were some deeper meaning to the colors, other than discriminability. For example, the reader might have thought that similar colors indicate a grouping of some kind (e.g. that a pink entity belongs to a pink person).

Moreover, colors from a certain rotation were often too similar to colors from another rotation. For example, it can be seen in Figure 4.5 that for each of the human silhouettes, a smartphone icon with a very similar color can be found in one of the color rotations. Also, the `Person` and entity colors in all three rotated versions are not as well distinguishable as in the original version.

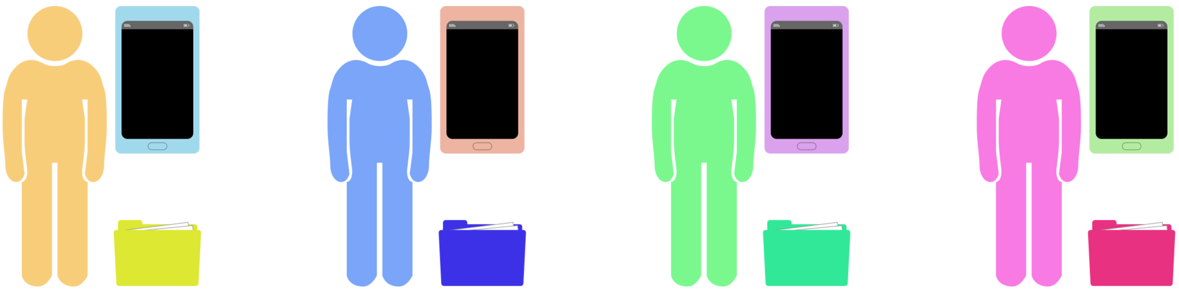


Figure 4.5: Agent and entity colors rotated by 0°, 180°, 90°, and 270° (left to right)

Hence, the rotation approach was abandoned, and colors were varied based on lightness as described above. It is thus ensured that similar elements always use similar color hues, yet remain well distinguishable even by people suffering from Protanopia, Deuteranopia, Tritanopia, or even Achromatopsia.

Colors for objects of different types

The distinctiveness between the colors of *different* object types is not as important as that between colors of the *same* types of objects. That is to say: Color is more important for distinguishing two items that have the same shape than it is for two items with different shapes. Thus the selection and discriminability of colors need not be handled as strictly for different types of actors.

Figure 4.6 shows that especially the default colors of `Person` agents and entities are not well distinguishable by readers suffering from color vision deficiencies. However, since shape *and* icon or text will be different, the weak color difference is neglectable. Figure 4.7 shows that items are still well distinguishable due to their shapes and icons.

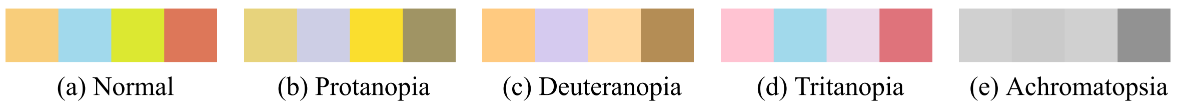


Figure 4.6: Default colors for `Persons`, `SoftwareAgents`, entities, and a "button press" effect and how they are seen by colorblind people

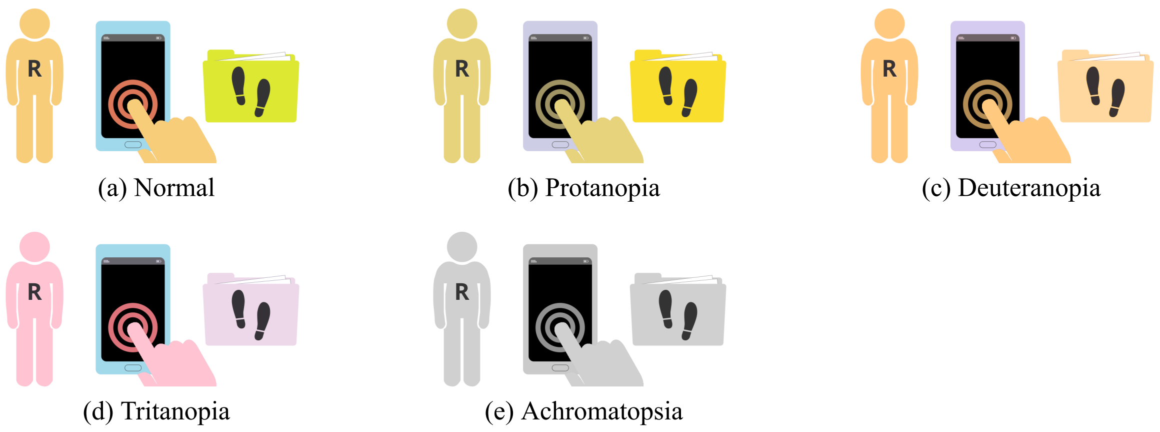


Figure 4.7: Default colors and shapes for different objects and how they are seen by colorblind people.

Text and icon colors

In a number of cases, agents and entities will be labeled with texts, letters or icons. To keep those recognizable on different background colors, a simple rule of thumb has been established using the colors' equivalents in the Lab color space:

- If a color's L (lightness) value is between **0 and 49**, the text or icon color is **white**.
- If a color's L value is between **50 and 100**, the text or icon color is **black**.

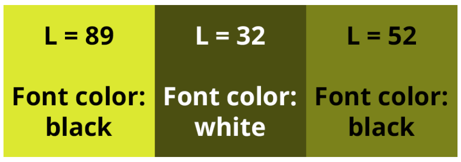
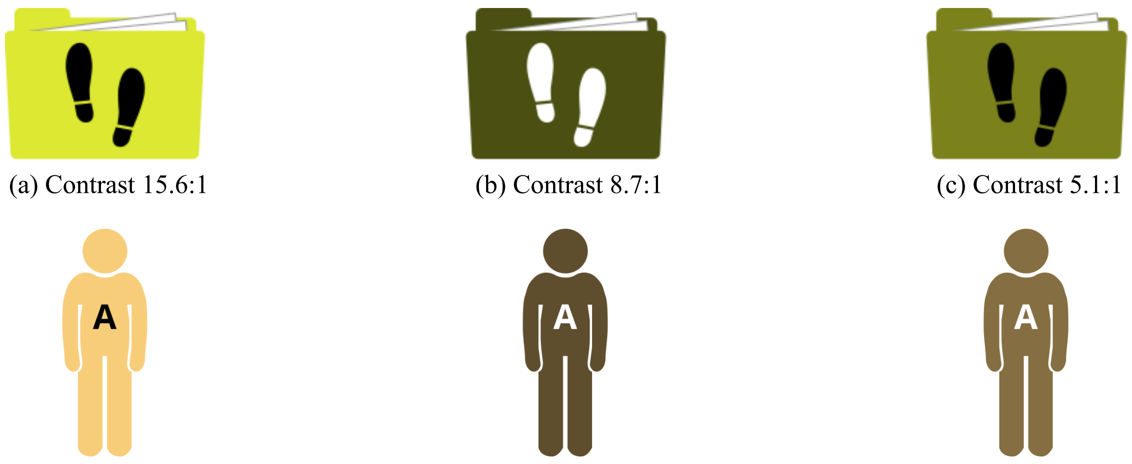


Figure 4.8: Black and white text depending on the background's lightness value.

Figure 4.8 shows an example of the application of this rule. By choosing the font color this way, a contrast ratio of at least 3:1 (often a lot higher) is achieved, which is "the minimum level recommended by ISO-9241-3 and ANSI-HFES-100-1988 for standard text and vision". (Cooper et al., 2016a) The WCAG's SC 1.4.3 (Minimum Contrast) requires a ratio of 4.5:1 for standard text, and 3:1 for "large-scale text and images of large-scale text", with "large-scale text" having a size of at least 18 point, or 14 point and bold style. The even stricter SC 1.4.6 (Enhanced Contrast) requires a ratio of 4.5:1 for large-scale text and 7:1 for standard text. (Cooper et al., 2016a)

The majority of icons and letters used in the PROV Comics qualify as large-scale text. By choosing the font or icon color according to the simple "black or white" rule proposed here, it is guaranteed that a contrast ratio of at least 3:1 is always achieved. In fact, when combined with the previously defined agent and entity colors, this rule yields a contrast ratio of at least 4.5:1 for all graphics containing text or icons. Thus, they even fulfill the stricter SC 1.4.6 (Enhanced Contrast) for large-scale text. Figure 4.9 shows some example graphics with high-contrast icons or letters.[10]



(a) Contrast 15.6:1 (b) Contrast 8.7:1 (c) Contrast 5.1:1

(d) Contrast 14:1 (e) Contrast 8.1:1 (f) Contrast 4.9:1

Figure 4.9: Examples of entities and agents with icons passing the WCAG SC 1.4.6 (Enhanced Contrast).

[10]Contrast ratios calculated by http://leaverou.github.io/contrast-ratio/

4.2.4 Panels and layout

All panels are perfect squares. Horizontally, they are separated from each other by a white space of 10 % of the panel size, while the vertical distance between rows of panel is 20 % of the panel size. For example, 600x600 pixel panels have 60 pixels of white space between them horizontally, and 120 pixels of white space vertically.
By arranging them this way, panels are grouped into rows, helping the reader determine the correct reading direction. This is explained by the gestalt law of proximity: Objects that are close to each other are perceived as a group. (Böhringer et al., 2008, p. 42)

However, no requirements are made as to how many panels each row should contain. Due to the fact that the comics are to be viewed on different devices the layout needs to be scalable. While a row may consist of four or five panels on a desktop or tablet computer, there might only be enough space for one panel per row on a smartphone.

The panels have black borders, the width of which should amount to 1 % of the panel size. For example, a 600x600 pixel panel should use a 6 pixel border. In case a caption or introductory text is added to the top of a panel, it is separated from the rest of the panel by a bottom border with the same properties.
Borders group the different graphics inside a panel together, so they are perceived as one large image. This is an application of the law of closure, which states that objects in a framed area are perceived as one unit. (Böhringer et al., 2008, p. 44)

4.2.5 Captions and text

It is generally desired to include as little text as possible in the comics. The vast majority of information should be conveyed by the graphics in order to provide an effortless "reading" experience. However, in certain cases, a few words are useful to support the interpretation of symbols. For example, when up- or downloading data, the words "Uploading..." or "Downloading..." are added below the cloud icon. These short annotations take only little cognitive capacity to read, but may greatly facilitate the understanding of certain icons.
Buttons also use textual labels, as it is very difficult to convey the actions they represent in the form of graphics. The labels should be as short as possible though, preferably consisting of only one or two words (e.g. "View graph", "Export CSV", etc.).
Several examples of textual annotations and button labels can be seen in the final prototypes described in section 4.3.3 and in the comics created for the reading study (see appendices O through S).

Captions should be used to provide information about the date and time the different activities took place. Every comic should begin with such a caption in the very first panel to give the reader temporal orientation. Whenever a significant amount of time[11] has passed between two activities, a caption may be used again to communicate this to the reader. For an example, see the third panel in Figure 4.20. The text of these captions may vary; aside from "Later that day", they may as well read "Five minutes later", or give an exact date and time.

[11]No exact definition will be given as to what a "significant amount of time" is. This may vary depending on the activities that are to be depicted. For example, when the user switches devices, an interruption of a few minutes is not unusual. It might thus be ignored, and the two activities considered to have taken place in quick succession. In other cases, however, a break of five minutes might be noteworthy.

4.2.6 Level of detail

PROV Comics are characterized by extreme simplicity and reduction to the essentials. The reader should never have to look for the important parts of the image. Thus, only relevant items are pictured; no purely decorative graphics are used. This includes the background, which is plain white at all times. No surroundings or other possible distractions are ever shown.
Scott McCloud calls this "amplification through simplification": By eliminating details, reducing images to their essential meaning, and focusing on specific elements, the emphasis will be put on the actual information. (McCloud, 2011, p. 30)

All shapes described in section 4.2.1 have been designed following that guideline. They only contain just enough detail to make them recognizable. This is of special interest regarding the human silhouette depicting the `Person` agent: When looking at a photograph or very detailed drawing of a person, one tends to see it as "someone else". However, when shown a very simple and abstract image of a person, viewers are more likely to identify with it. (McCloud, 2011, p. 36)

4.2.7 Recurring image structures

As already mentioned earlier, activities will not be represented by a single graphic, but by a sequence of three to five comic panels. Similar activities should be illustrated by similar sets of panels, making use of recurring image compositions. For example, the activities of the data sub-models *Export*, *Aggregate*, and *Visualize* are comparable in that they take one kind of data and create a different kind of data from it. They can thus be visualized in a very similar manner as will be shown later in Figures 4.14, 4.15, and 4.19.
Using recurring image structures whenever possible adds to the comics' consistency, comprehensibility and learnability: Once readers have understood the *Export* panels, for example, they will easily be able to understand *Aggregate* and *Visualize* panels, too.

4.2.8 Commonly known symbols

Some of the graphics used in the comics rely on the reader's (computer) experience. For example, in recent years, the "cloud" icon has become a widely known symbol for external data storage space. The icon is frequently accompanied by an "arrow circle", indicating that data is being synchronized (i.e. up- or downloaded) or refreshed.

Conventions like these are useful when it comes to depicting rather abstract items. Concrete objects, like a person, a smartphone, or a computer, can easily be drawn as a simplified graphic, but it is not as easy with more abstract notions like "data". However, "sheet of paper" and "document folder" icons have been used for many years to symbolize data and collections of data. The graphics representing exported files, collections of Quantified-Self data, but also data transmission and synchronization build upon this long tradition.

4.3 Static prototypes

Much like in UI prototyping, different types of drafts were created in several iterations. In the course of time, some guidelines emerged that should generally be applied to each of the five use cases (or sub-models). In particular, a default sequence of comic panels has been defined for each of the use cases. The development of these static prototypes, starting from the very first drafts up to the final design, is presented in this section.

4.3.1 Prototype dimensions

Prototypes can take various forms, depending on their immediate goal, and may be characterized by specifying several dimensions beforehand, such as functional breadth and depth, interactivity, etc. (Richter and Flückiger, 2013, p. 53) For the static prototypes, it was determined that they be rather broad, depicting each of the five use cases as well as a few combinations of two or more use cases. As for the depth, however, no functionality and interactivity was required at this point; only static pictures needed to be created.

A number of real-world as well as example provenance documents, compliant with the W3C PROV standards, was already available on the PROVSTORE website. However, they did not cover all of the aforementioned use cases and their possible combinations. In order to design and test the new visualization, it was therefore necessary to create further PROV documents covering at least all five sub-models as well as different fields of application. For example, the available PROV documents were almost exclusively concerned with bodyweight data. Since the comics should also cover other domains of application, examples concerning the tracking of steps, pulse, blood pressure etc. were created.

As to the look of the prototypes, the earliest versions did not follow any specific design principles yet, but used stock graphics and images provided by storyboarding tools to determine how the use cases might generally be depicted. Prototypes from later iterations, however, comply with the design guidelines defined in the concept in section 4.2.

4.3.2 Early drafts

The concepts described in section 4.2 were not developed all at once, but in an iterative fashion. Thus, the earliest drafts feature some elements that were later abandoned; for example, they still used decorative background images.

The very first drafts went rather deep into the technical details, i.e. they depicted happenings at method level. For example, one comic that combined a *Request* (download) and *Visualize* process used differently colored Python icons to represent three Python methods (see Figure 4.10). The first method (`time_series`) fetched step data from FITBIT, a second method (`generateDict`) "translated" that data into a format used locally by the application, and the third method (`matplotlib_plot`) finally created a graphic from that data.

It quickly became clear that, while this comic might be useful to developers who wish to reconstruct the sequence of operations performed by an application, it was way too detailed and low-level to be understood by end users. Even though methods and usage of libraries were depicted in a rather metaphorical style, an average user, who might not even know what a "method" or "library" is in a programming context, would probably have had great difficulties understanding this comic.

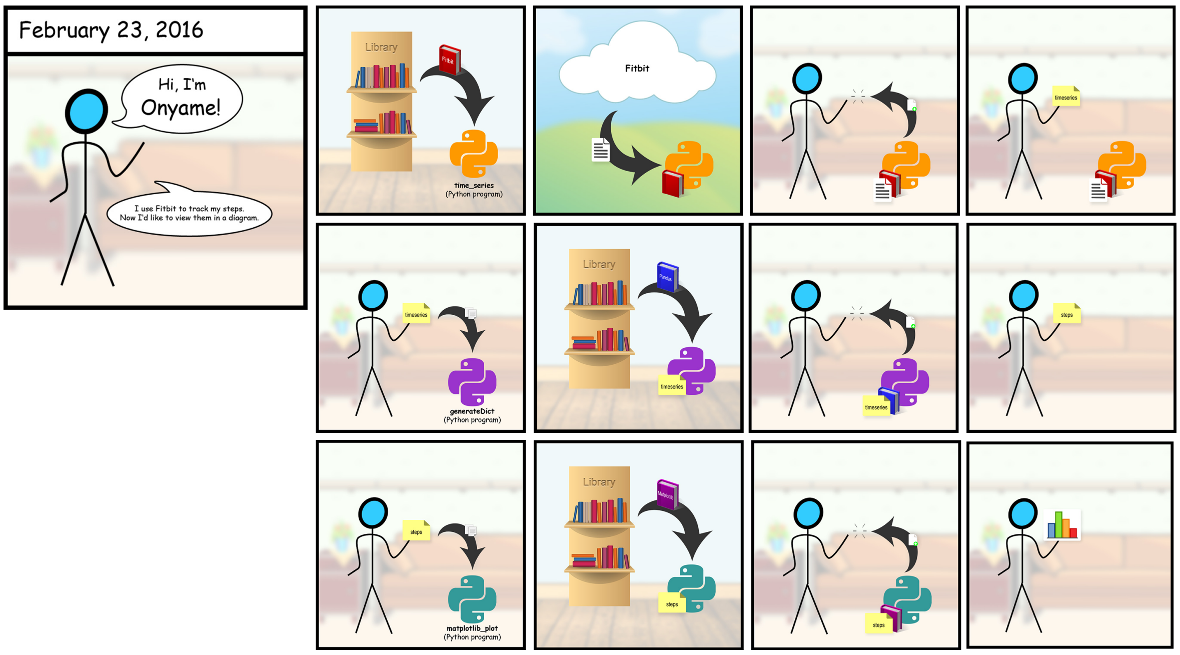

February 23, 2016
Hi, I'm Onyame!
I use Fitbit to track my steps. Now I'd like to view them in a diagram.
Library
Fitbit
time_series (Python program)
Fitbit
timeseries
generateDict (Python program)
timeseries
steps
matplotlib_plot (Python program)
steps

Figure 4.10: A very detailed draft depicting a visualization process at method level.

Other early drafts that were created using StoryboardThat[12] already came closer to the degree of abstraction that was desired for a visualization aimed at end users (see Figure 4.11). Note that this version also contained rather detailed texts under each panel. In later versions, the use of descriptive text was reduced to an absolute minimum.

4.3.3 Final prototypes

This section shows the final prototypes and describes the default panel sequences that have been defined for the different use cases. These are explained in great detail, since an exact specification is needed in order to have the comics be generated automatically.

To avoid any misunderstandings, two expressions need to be clarified: In the following subsections, the devices that initially track and record data (e.g. wearables like step tracking bracelets, or WiFi-enabled bodyweight scales) will be referred to as *tracking devices*. The devices that run the Quantified-Self application (e.g. a smartphone or a desktop computer), and to which the data will be transferred in order to be stored for a longer time, are called *application devices*.

Input Inputs may occur either manually or automatically. A mood tracking application, for example, may ask the user how he feels several times a day, and the user selects a value manually from a scale that ranges from "very good" to "very bad". In contrast, a user with a step counting bracelet will never have to enter any values by hand; the bracelet will usually send the number of steps it counted to a smartphone or other device via Bluetooth.

[12]http://www.storyboardthat.com/

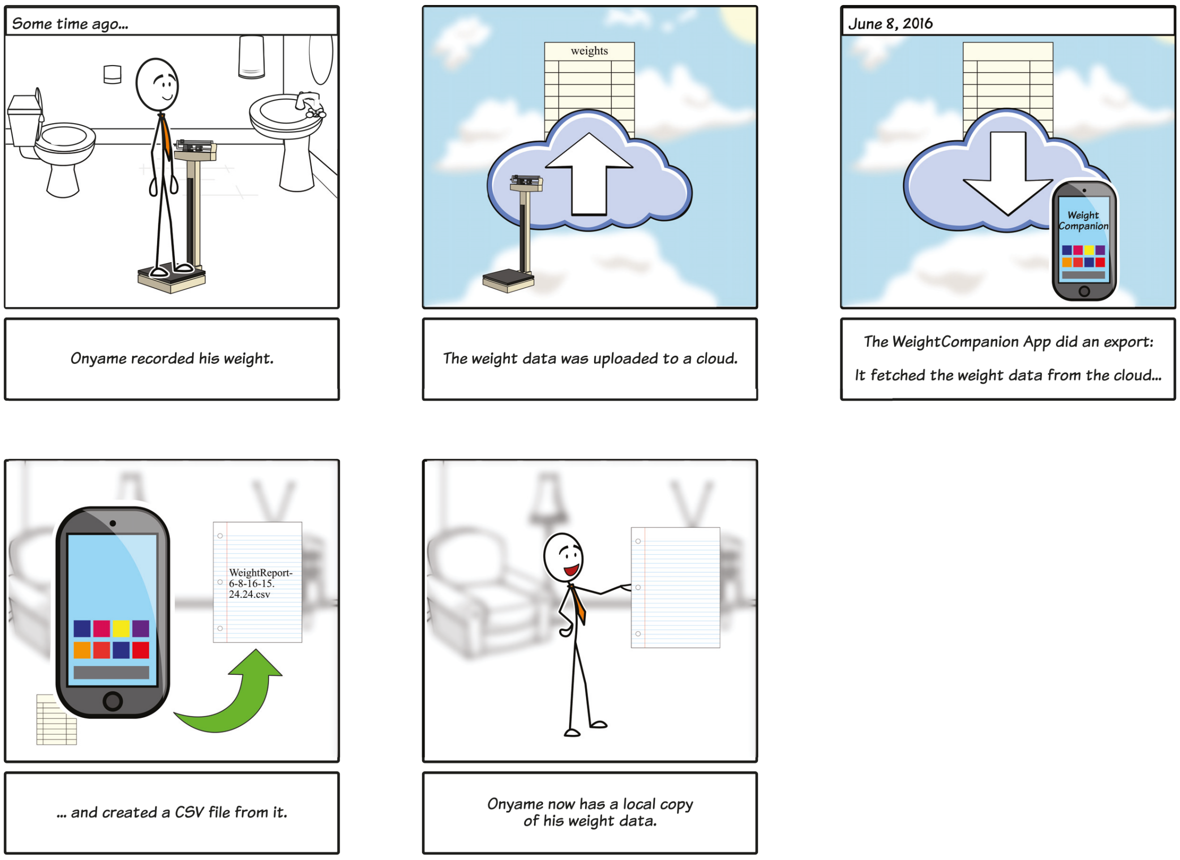

Some time ago...
Onyame recorded his weight.
weights
The weight data was uploaded to a cloud.
June 8, 2016
Weight Companion
The WeightCompanion App did an export:
It fetched the weight data from the cloud...
WeightReport-6-8-16-15.24.24.csv
... and created a CSV file from it.
Onyame now has a local copy of his weight data.

Figure 4.11: Early draft created with StoryboardThat

The input use case is depicted in four to five panels:

1. **A panel showing a comic figure performing the activity that was recorded.** The comic figure is shown using the device that originally measured or tracked the data (e.g. bracelet, blood pressure meter). If applicable and necessary, an icon on the device indicates the type of data that was recorded; for example, a step counting bracelet may be recognized by a footprint icon, while a pulse tracking bracelet uses a heartbeat icon.
2. *In case the data came from a bracelet or similarly small device:* A panel showing a zoomed-in view of that device, so the device as well as its icon are better recognizable.
3. **A panel showing the figure using the application device that will hold the data (e.g. smartphone, computer).** The tracking device is also still visible. This is to provide the reader with some orientation and context for the following panels.
4. **A panel showing the transmission (or manual input) of the data to the application device.** This is a close-up view of either the application device only (in case of manual input) or both the tracking and the application device (in case of automatic transmission). The application name is displayed in the title bar to give the reader additional orientation. An automatic transmission is symbolized by a graphic representing radio waves moving from the tracker to the application device. Also, the application displays a word like “syncing” or “receiving” and a corresponding icon.

A manual input is expressed by one (or both) hand(s) typing in values into a text field using a real or soft keyboard, depending on the device (e.g. smartphone, computer).

5. **A panel indicating that the transmission (or manual input) is complete and the data has been saved.**
   This is represented by an image of a folder on the application device. The folder is marked by an icon that indicates the type of data. Also, a word like "complete" or "done" is displayed. In case a tracking device was the input source, it is still visible near the application device.

Figure 4.12 shows an example of a user wearing a step tracking bracelet which sends its data to an app on the user's phone. Figure 4.13 shows an example of a user measuring his blood pressure with a conventional blood pressure meter and then manually typing the values into an application on a desktop computer.

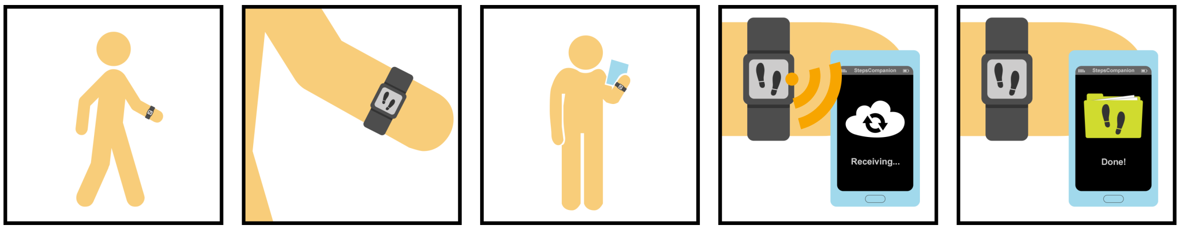


Figure 4.12: Comic depicting automatic data input from a step counter.

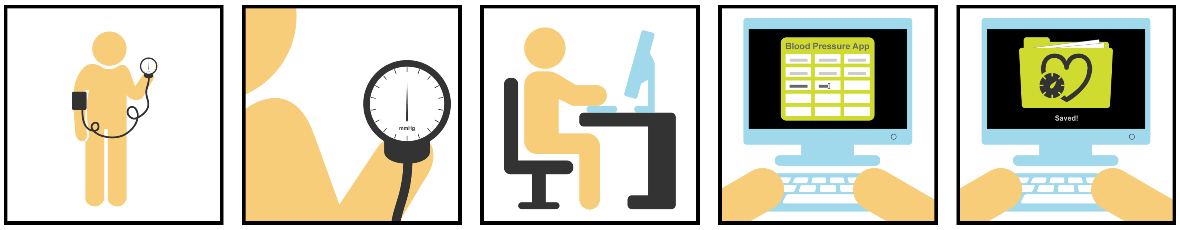


Figure 4.13: Comic depicting manual input of blood pressure values.

Export Exports are usually triggered manually by users in order to save their collected data in a different file format that can be read and used by other applications. Useful formats may include CSV, XML, JSON, as well as image formats. In the DLR's current implementations, the CSV format plays the most important role.

The export use case is depicted in three to four panels:

1. *In case the export takes place at the very beginning of a comic,*
   or the comic figure was not using this application device in the previous picture(s):
   A panel showing the figure using the device (e.g. smartphone, computer). This is to provide the reader with some orientation and context for the following panels.

2. **A panel showing the option to export data in another format.**
   This is a close-up view of the application device featuring an image of a folder. The folder is marked by an icon that indicates the type of data. Beneath the folder, there is a button with a text like "Export *[format]*", with *format* being the target data type, e.g. "CSV".

3. **A panel showing the pressing of the export button.**
   The figure's second hand comes into view, with the index finger touching the button. Red concentric circles around the finger add to the impression that the button is being tapped/pressed.

4. **A panel showing the new exported data on the device.**
   The application now displays a document icon with a textual representation of the export file type. An "appearance" effect around the icon suggests that this is a new file that has just been created.

Figure 4.14 shows an example of a user exporting step count data to a CSV file.

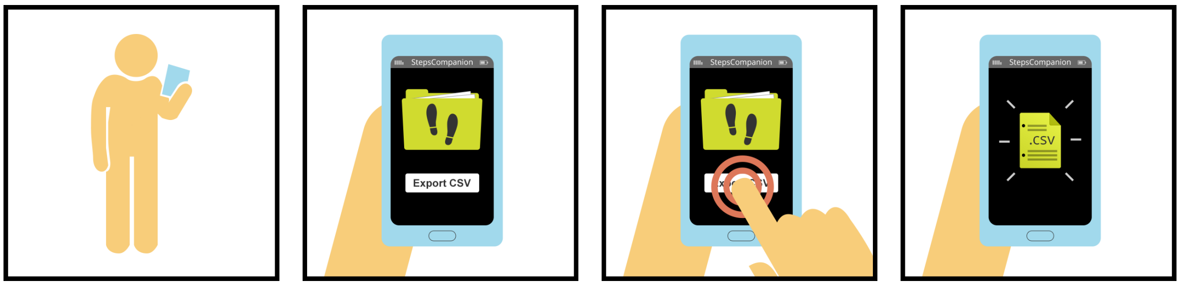


Figure 4.14: Comic depicting the export of step data into a CSV format.

Aggregate Aggregations are usually triggered by a software, for example when the user chooses to view a graph featuring two different kinds of data at once in order to understand their relationships and unveil correlations between them. In order to create and display such a graph, the software first needs to aggregate those two sets of data. For the DLR's current implementations, aggregations of two data sources are sufficient.

Similarly to export, aggregations are depicted in three to four panels:

1. *In case the aggregation takes place at the very beginning of a comic, or the comic figure was not using this application device in the previous picture(s):*
   A panel showing the figure using the device (e.g. smartphone, computer). This is to provide the reader with some orientation and context for the following panels.

2. **A panel showing the option to aggregate two sets of data.**
   This is a close-up view of the application device featuring images of two folders with a plus sign between them. The folders are marked by icons that indicate the two types of data. Beneath the folders, there is a button with the word "Combine". This word was chosen because the average end user may not – or not exactly – know what an "aggregation" is; "combining", however, should be understandable by most people.

3. **A panel showing the pressing of the aggregation button.**
   The figure's second hand comes into view, with the index finger touching the button. Red concentric circles around the finger add to the impression that the button is being tapped/pressed.

4. **A panel showing the new data set resulting from the aggregation.**
   The application now displays a folder icon that includes the icons of both original data sets.

Figure 4.15 shows an example of a user aggregating step count and heart rate data, resulting in a new set of data.

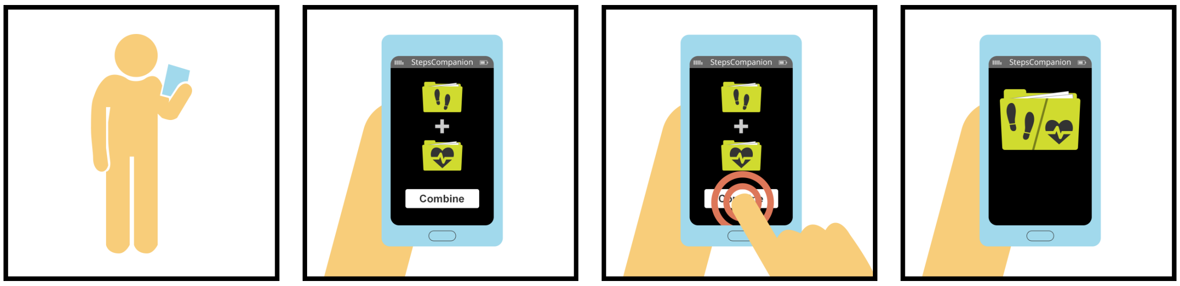


Figure 4.15: Comic depicting the aggregation of step count and heart rate data into a new set of data.

Request Although requests may also be triggered manually, they are most often done automatically by a software. Many applications allow users to set up "cloud" synchronization once, and from that moment on, their data will automatically be sent to an organization's server for storage in regular intervals. Also, the application will regularly perform automatic polls for changes or the existence of new data on the server.

Requests are depicted in three to four panels. Since there are two main types of requests (upload and download), both will be described individually. No request buttons and fingers pressing those buttons are used in the following descriptions and examples, because the usual scenario is that of automatic synchronization.

Download:

1. *In case the request takes place at the very beginning of a comic, or the comic figure was not using this application device in the previous picture(s):*
   A panel showing the figure using the device (e.g. smartphone, computer). This is to provide the reader with some orientation and context for the following panels.
2. **A panel indicating that data on the application device is missing or outdated.**
   This is a close-up view of the application device featuring a grayed-out image of a folder with a "not available" icon on it. In the background, an office building can be seen, wearing the name of the remote organization. The building is disproportionately smaller than the application device to create the impression that it is somewhat far away ("remote"). An image of a folder with the usual data type icon is attached to the lower right corner of the building to indicate that the data is currently stored on the organization's servers.
3. **A panel showing the transmission of data from the organization to the application device.**
   The application is displaying a "cloud sync" icon as well as the word "downloading". An arrow-headed line is now running from the building to the device. The line is gradually becoming thicker towards the bottom of the picture to add to the impression of depth and perspective. A second, smaller version of the folder is shown "wandering" along the arrow towards the application device to indicate that a copy of the data is being transmitted from the organization to the device.

4. **A panel indicating that the data has been copied from the organization to the device.**
   Two identical folders are now seen on the device as well as the building to make clear that both now contain identical data. The application displays a word like "done" or "complete".

Figure 4.16 shows an example of a user downloading step count data from an organization using a smartphone.

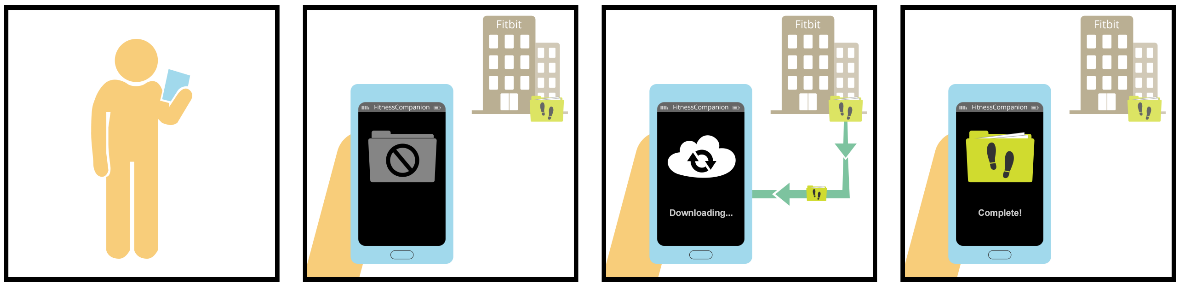


Figure 4.16: Comic depicting the download of step data from a "cloud", i.e. an organization's server.

Upload:

1. *In case the request takes place at the very beginning of a comic, or the comic figure was not using this application device in the previous picture(s):*
   A panel showing the figure using the device (e.g. smartphone, computer). This is to provide the reader with some orientation and context for the following panels.

2. **A panel showing data on the application device and a remote organization.**
   This is a close-up view of the application device featuring a folder with an icon indicating the type of data. In the background, an office building can be seen, wearing the name of the remote organization. The building is disproportionately smaller than the application device to create the impression that it is somewhat far away ("remote").

3. **A panel showing the transmission of data from the application device to the organization.**
   The application is displaying a "cloud sync" icon as well as the word "uploading". An arrow-headed line is now running from the device to the building. The line is gradually becoming thinner towards the top of the picture to add to the impression of depth and perspective. A small version of the folder seen in the previous panel is shown "wandering" along the arrow towards the office building to indicate that a copy of the data is being transmitted from the device to the organization.

4. **A panel indicating that the data has been copied from the device to the organization.**
   A copy of the data folder is attached to the bottom right corner of the building, while the original folder is also still visible in the application. Both folders use identical icons to make clear that they contain identical data. The application displays a word like "done" or "complete".

Figure 4.17 shows an example of a user uploading blood pressure data to an organization using a desktop computer.

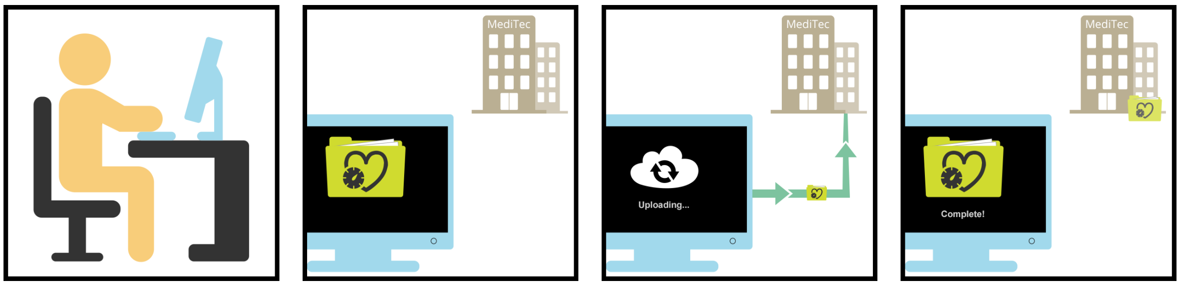


Figure 4.17: Comic depicting the upload of step data to a "cloud", i.e. an organization's server.

Visualize Visualization is usually triggered manually by users in order to view their collected data in various possible types of graphs, charts, or diagrams. A visualization may, for example, be a line, bar, or pie chart, but also a point cloud or scatter diagram. As of now, the comics do not show the actual, "real" data when displaying a visualization activity, as this would require to either include the actual data in the provenance document, or provide some kind of link to the actual data source. Thus, only a few exemplary graphics (different line charts and a bar chart) have been designed to serve as a symbol of the actual diagram that was created by a software (see Figure 4.18).

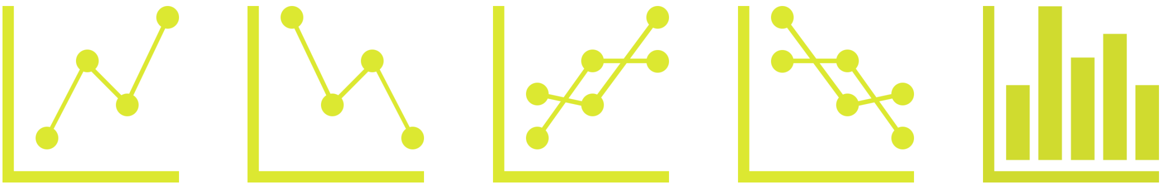


Figure 4.18: Graphics to symbolize diagrams.

The visualization use case is depicted in three to four panels:

1. *In case the visualization takes place at the very beginning of a comic, or the comic figure was not using this application device in the previous picture(s):*
   A panel showing the figure using the device (e.g. smartphone, computer). This is to provide the reader with some orientation and context for the following panels.

2. **A panel showing the option to view data in a diagram.**
   This is a close-up view of the application device featuring an image of a folder. The folder is marked by an icon that indicates the type of data. Beneath the folder, there is a button with a text like "View graph", "View diagram" or similar.

3. **A panel showing the pressing of the "view" button.**
   The figure's second hand comes into view, with the index finger touching the button. Red concentric circles around the finger add to the impression that the button is being tapped/pressed.

4. **A panel showing a diagram displayed by the application.**
   The application now displays a graph or other type of diagram. In many cases, this will be an X-versus-time graph, X being the type of data to be displayed.
   At the moment, this diagram is only a dummy, meaning that it is a static image that does not reflect the actual values from the data set. It may, however, be an idea for future development to generate this graphic according to the actual original data.

Figure 4.19 shows an example of a user viewing weight data as a graph.

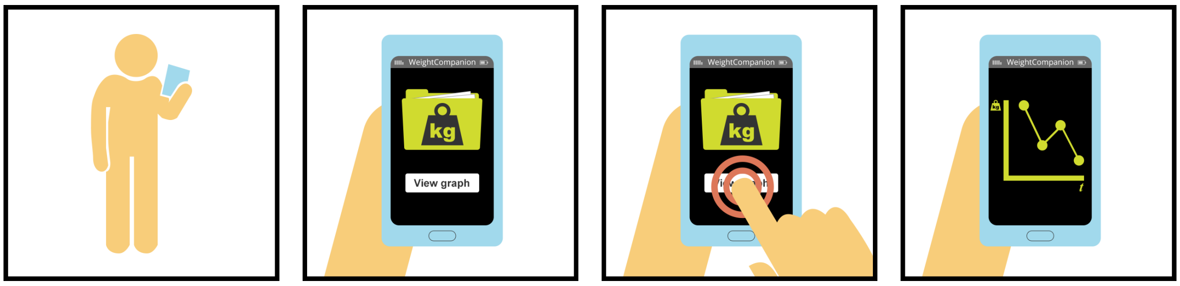


Figure 4.19: Comic depicting the visualization of weight data.

4.3.4 Combinations of and transitions between use cases

The five panel sequences defined above may be appended to each other to represent a series of actions. In some cases, small modifications may or should be made. These are described below.

Direct chaining

In many cases, the different use cases may directly be chained together without any modifications, according to the definitions made in section 4.3.3. As an example, Figure 4.20 displays the transition from one *Request* (upload) to another (download). Since the application device has been switched in between, the second *Request* sequence begins with the first "orientation" panel, as described in section 4.3.3.

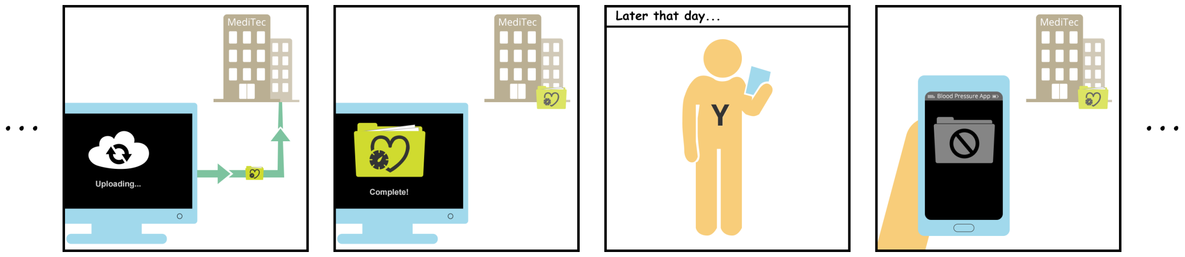


Figure 4.20: Transition from *Request* (upload) to *Request* (download).

Similarly, an *Input* followed by an upload *Request* is shown in Figure 4.21. Since the same device is used for the *Request*, immediately after the *Input* action has been finished, no "orientation" panel is needed, and the *Request* sequence may start from panel #2, as described in section 4.3.3.

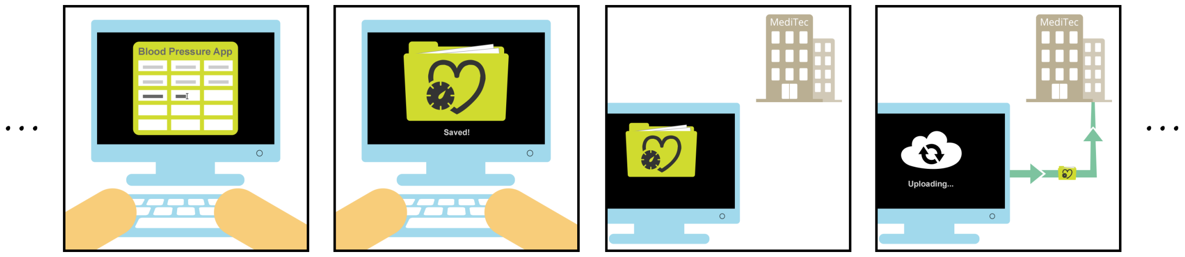


Figure 4.21: Transition from *Input* to *Request* (upload).

Appending Export and Visualize

A small adaptation should be made when appending *Export* or *Visualize* to a previous sequence. For example, when combining *Aggregate* and *Visualize*, and there is no switching of application devices in between, this means that the *Visualize* sequence will start from panel #2, as described in section 4.3.3. In this case, the button indicating the option to visualize data should be adapted to have an "appearance" effect, as seen in the third panel in Figure 4.22. This applies equally when appending an *Export* sequence.

The small adaptation was shown to be necessary after two readers in a quick intermediate test had exhibited difficulties understanding the transition from one sequence to the other. They were confused as to where the "view graph" button suddenly came from. The problem has been solved by simply adding a few short, radial lines around the button, indicating that the button has just become newly available. In the actual reading study (see section 5), none of the readers reported any difficulties concerning this transition.

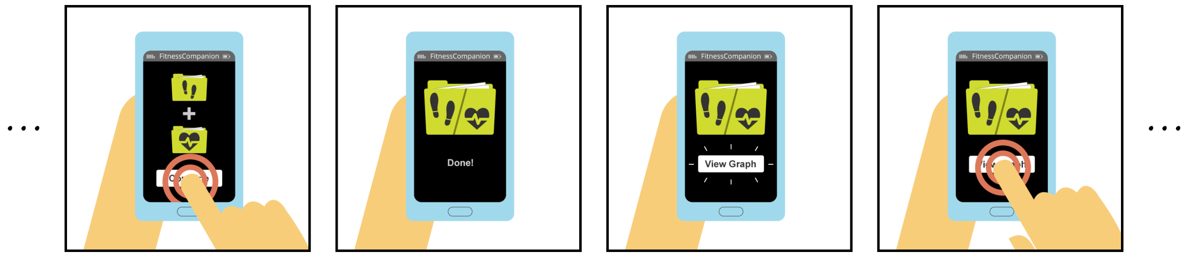


Figure 4.22: Transition from *Aggregate* to *Visualize* with "appearance" effect on the button.

Appending Aggregate

When appending *Aggregate* to an existing comic, it should be made sure that the two sets of data have already been depicted as residing on the device in an earlier panel, as shown in Figure 4.23. Otherwise, a reader might be confused as to where the second (or both) data sets come from.

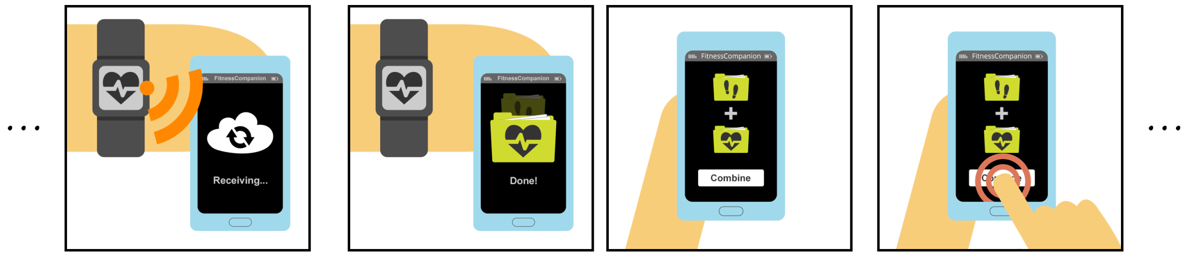


Figure 4.23: Transition from *Input* to *Aggregate*.

4.4 Dynamic prototype

After the detailed concept for the different comic sequences had been developed, a dynamic prototype was built to provide a proof of concept. Since the current data models did not provide all information necessary to generate appropriate images automatically, some adjustments had to be made to the models. These adjustments, as well as the basic operating principles of the prototype, are described below.

The prototype is enclosed on CD. However, since it may be further developed, the most recent version can be found at http://www.struminski.de/provcomics/.

4.4.1 General workflow

The prototype was built in the form of a website that can be run locally in the browser. No web server is needed to host the prototype website. The workflow is as follows: The user enters a PROVSTORE username into an input field. That user's documents are then looked up in the PROVSTORE and listed in a drop-down box. The user may now choose a document from the list. The website immediately fetches the document and creates a comic from it, which is then displayed.

4.4.2 Technologies

The employed techniques obviously include HTML and CSS, but also JavaScript as well as the jQuery library.

The website makes use of the ProvStore jQuery API[13] to list documents available from the PROVSTORE. The jQuery plugin URI.js[14] was added to provide deep-link support, i.e. the ability to link to a certain comic.

For the comics themselves, a `ProvComics.js` script was written, defining three `proto-types` ("classes"):

ProvComic serves as a frame to contain all comic panels. It is also the general starting point for creating a PROV COMIC inside a given HTML element. For example, if there is a `<div id="comic">` tag in the HTML, a new PROV COMIC may be started within the `div` element by declaring `var comic = new ProvComic("#comic")`.

Panel represents a single comic panel and has all necessary abilities to create any of the panels described in the concept. For example, it provides functions to add captions, `Persons`, `SoftwareAgents`, `Organizations`, different types of entities, etc.

PanelGroup represents a predefined sequence of panels, as described in section 4.3.3. They make it easier to insert recurring panel sequences. For example, it provides a function to add all panels depicting a download *Request* at once.

Fetching the provenance document from the PROVSTORE, reading it, and calling the necessary functions in `ProvComic.js` is done by a separate script. The PROV documents are retrieved in XML format. The JSON format would have been equally suitable; however, with JSON, child objects and properties are always sorted alphabetically. Thus, the XML format was chosen in case the original node order should ever become relevant – even though this should never be the case.

Since this is a prototype, it has only been tested against a number of example PROV documents specially created to represent the five Quantified-Self use cases.

[13]https://provenance.ecs.soton.ac.uk/store/help/api/#jquery
[14]http://medialize.github.io/URI.js/

4.4.3 Translating PROV data into comics

Activities are the centerpiece when it comes to converting a PROV document into a comic. The activities prescribe which of the panel sequences defined in section 4.3.3 need to be displayed. It is then only a matter of routine to fill in the correct graphics for agents and entities.
Thus, the script first looks for activities in the PROV document to determine the general panel sequence. If there is more than one activity, the correct order can be derived from the timestamps that activities usually have. In case no timestamp is available, the order of appearance in the PROV document may be used, which is preserved in the XML versions of the PROV files.
The script then reads the attributes of involved agents and entities to decide which graphics to include in these panels. For example, the attributes indicate whether to display a smartphone or a computer, a folder or a single document, a steps icon or a weight icon, etc.

The example PROV documents that have been created during the prototyping phase (see section 4.3.1) already contain all the information necessary to automatically generate comics from them. For example, looking at the PROV document given in Figure 4.24a, one can see that

- two agents play a role:
    - a user labeled `Regina Struminski` (who is a `Person`) and
    - an app labeled `StepCounter` (which is a `SoftwareAgent` running on a `smartphone`).
    - The app is acting on the user's behalf (i.e. the user "tells" the app to do something).
- two entities play a role:
    - `steps` (containing data of the `steps` type) and
    - `graphic/diagram` (which is a `linechart`).
- the activity that is being conducted is `visualize`.
    - It uses `steps`.
    - It creates `graphic/diagram`.

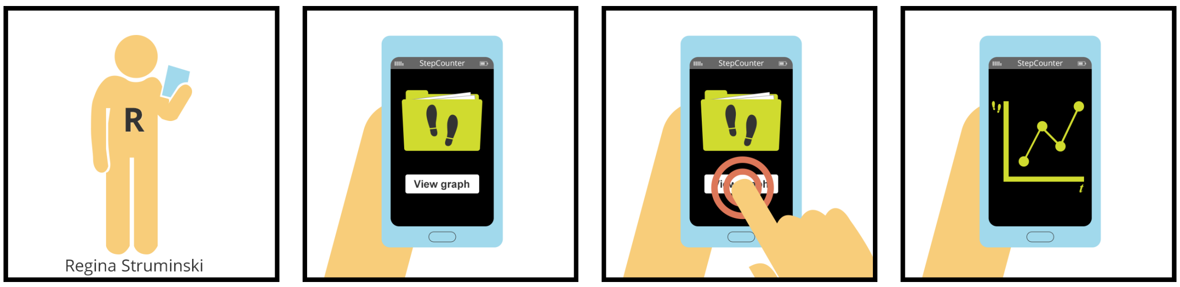
With this information, a script may generate a comic like the one shown in Figure 4.24b directly from the provenance document.

4.4.4 Adjustments to the PROV data model

The existing PROV data model as described in section 2.2.2 did not contain all information necessary to automatically generate an understandable comic. Most importantly, three of the five sub-models contained only one agent. It is not specified whether this agent is meant to be a `Person` (i.e. the user) or a `SoftwareAgent` (i.e. a smartphone app or similar).

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix graphic <http://software.dlr.de/qs/graphic/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  wasGeneratedBy(qs:graphic/diagram, method:visualize, 2016-12-01T16:06:22+00:00, [prov:role="displaying"])
  activity(method:visualize, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  entity(qs:graphic/diagram, [prov:type="linechart", prov:label="Line chart"])
  entity(userdata:steps, [prov:type="steps", prov:label="Steps database"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  agent(qs:app/stepcounter, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="StepCounter"])
  wasAttributedTo(qs:graphic/diagram, qs:user/regina@example.org)
  wasAttributedTo(userdata:steps, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/stepcounter, qs:user/regina@example.org, -)
  used(method:visualize, userdata:steps, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(qs:graphic/diagram, userdata:steps, -, -, -)
  wasAssociatedWith(method:visualize, qs:app/stepcounter, -)
endDocument
```

(a) PROV-N code for a visualization use case



(b) A comic corresponding to the PROV-N code seen in Figure 4.24a

Figure 4.24: PROV-N document and its corresponding comic

In any case, one agent is missing: A script will need to know both the `Person` *and* the `SoftwareAgent` in order to generate a useful comic.

Several models were also lacking correct attributions of entities to agents. The *Aggregate* model did non seem to make much sense using only a single source of data, and the *Request* model should actually be divided into an upload and a download request.

All changes made to the five sub-models are described and depicted below.

Input

In the existing *Input* model, it seems as if the user himself is performing the `Input` activity. However, inputs are really not done by a `Person` alone, but with the help of a `Soft-wareAgent` that acts on the `Person`'s behalf. A `Person` does not write data into a file and saves it; the `SoftwareAgent` does that.

This is actually a question of granularity, i.e. exactly how specific the data model should be and how far it should go into detail. The current model is not wrong, but for the PROV COMICS, its granularity is not fine enough. To automatically generate a useful comic, the PROV document must at least contain the `SoftwareAgent` that the `Person` used to input data – which may, for example, have been a Quantified-Self app running on a smartphone. This information is necessary to decide what type of device should be pictured in the comic.

The new model thus contains two agents (as shown in Figure B.1b in appendix B):

1. a `Person`, whom the "before" and "after" versions of the data are attributed to, and
2. a `SoftwareAgent` that uses the "before" data and derives the "after" data from it.

Both agents are associated with the `Input` activity: The `Person` induces or initiates it, while the `SoftwareAgent` actually carries it out. Sometimes even a third agent may be present. For example, a step counting bracelet transmitting its data to the `SoftwareAgent`, would take a similar position in the model as the `SoftwareAgent`, acting on behalf of the `Person` and associated with the `Input` activity (see Figure B.1c in appendix B).

Export

Like the *Input* model, the *Export* model has been adjusted to contain a `SoftwareAgent` that acts on a `Person`'s behalf. A `Person` does not actually create an export file, but instead induces a `SoftwareAgent` to do so. Thus, both agents are associated with the activity. The `SoftwareAgent` uses existing data, which is attributed to the `Person`, and derives new data (i.e. the exported file) from it. This new data, or entity, is again attributed to the `Person`. The new *Export* model is depicted in Figure C.1b in appendix C.

Aggregate

The existing *Aggregate* model already contained a second agent to represent the software that is actually doing the aggregation. However, it was not clear that the aggregation usually requires two or more data sources. The updated model contains a second entity serving as input for the aggregation process. Thus, two entities attributed to a `Person` are used by the `Aggregation` activity (which is performed by the `SoftwareAgent` on the `Person`'s behalf). The result is a new entity that is also attributed to the `Person`. The new *Aggregate* model is depicted in Figure D.1b in appendix D.

Request

For the *Request* model to be precise, it is necessary to have two slightly different variants: one for an upload request, and one for a download request. The difference is only in the attributions: When uploading, the source data is attributed to the `Person` only, and the result data is attributed to both the `Person` and the `Organization` (see Figure E.1b in appendix E). When downloading, the source data is attributed to both the `Person` and the `Organization`, and the result data is attributed to the `Person` only (see Figure E.1c in appendix E). A `role` attribute was added to the `wasGeneratedBy` relation, which is to have a value of "uploading" or "downloading". This makes it less complicated for a script to determine which one of the two actions is being performed.

The "API Configuration" plan was removed from the model for several reasons. First of all, it did not seem to fit the definition of a plan: "A plan is an entity that represents a set of actions or steps intended by one or more agents to achieve some goals.". (Moreau and Missier, 2013) Thus, it usually describes some kind of workflow; an "API Configuration", however, is not a workflow or a set of actions.
Secondly, the plan had not been included correctly: There is no relation `hadPlan` in PROV-DM. This property is part of the *PROV Ontology (PROV-O)*, which "defines the OWL2 Web Ontology Language encoding of the PROV Data Model" (Lebo et al., 2013). The correct way of including a plan in a PROV data model would have been a ternary association between the activity, the agent, and the plan. (In the provenance graph, this would have become visible as a multi-edge pointing from the *Request* activity to the agent as well as to the plan.)[15]
Lastly, the PROV COMICS do not need this information at all, and it is questionable whether it is actually needed by any other application at the moment. At this time, it seems more reasonable not to include plans in the Quantified-Self data model.

Visualize

Like the *Input* and *Export* models, the *Visualize* model has been adjusted to contain a `SoftwareAgent` that acts on a `Person`'s behalf. A `Person` does not actually create a visualization himself, but instead induces a `SoftwareAgent` to do so. Thus, both agents are associated with the activity. The `SoftwareAgent` uses existing data, which is attributed to the `Person`, and generates a graphic from it. This new graphic entity is again attributed to the `Person`. The new *Visualize* model is depicted in Figure F.1b in appendix F.

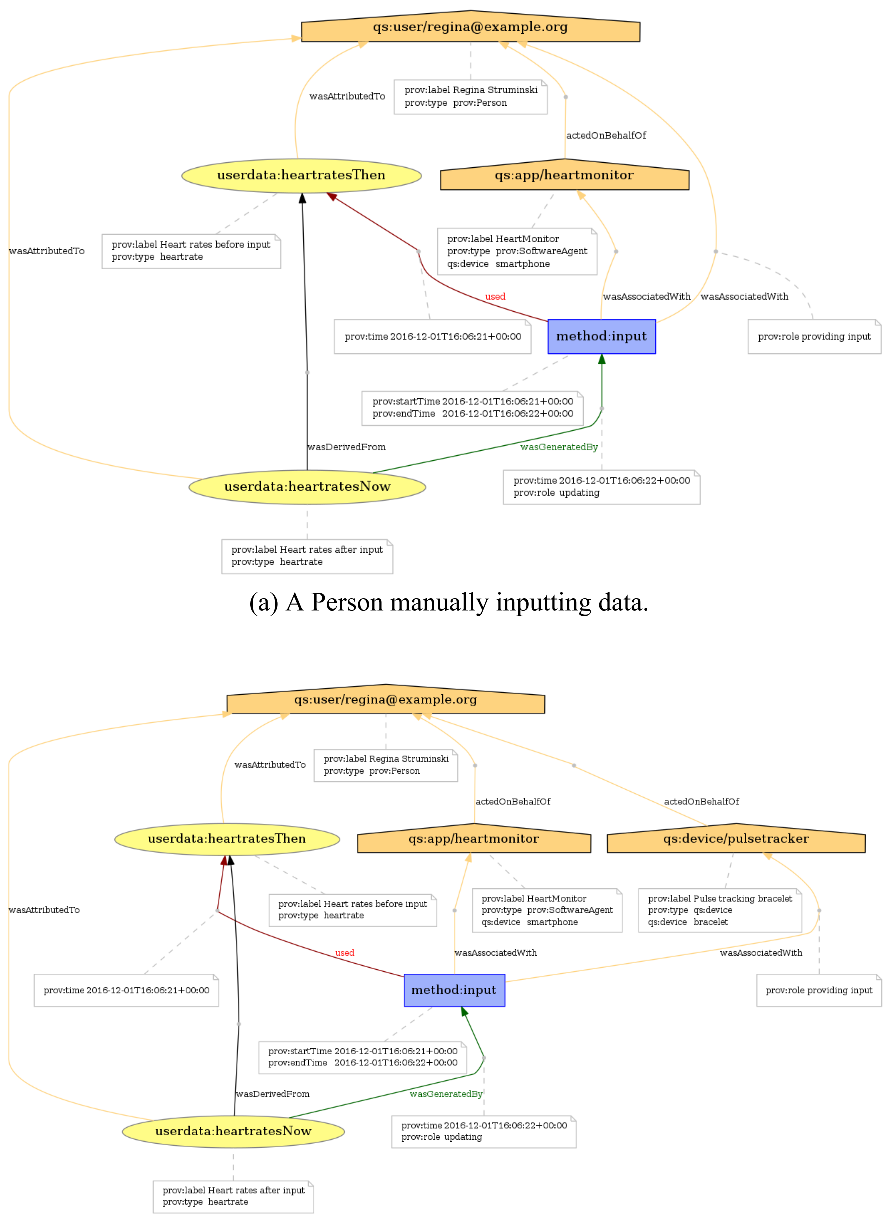
4.4.5 Example PROV documents

Since the models had been updated, the existing PROV documents were now outdated. Also, they contained inconsistencies in some places and lacked information that was important for the automatic generation of the comics (cf. adjustments made in section 4.4.4). Thus, a number of new examples had to be created. At least one PROV document was written for each of the five Quantified-Self sub-models to serve as input files for the prototype. Two example files were needed for both *Input* and *Request*.
The PROV-N sources to all examples can be found in Appendices G through M.

[15]See https://www.w3.org/TR/prov-dm/#section-example-two for further details and examples.

Input

For the *Input* model, two examples were created, since the scenario may vary in whether the input is done manually by the user or automatically by a tracking device communicating with the application device. This distinction has great impact on how the *Input* activity will be depicted in the comic. An example of manual input might be a user who measures his pulse by palpating it at his wrist, then enters it into a smartphone app using a number pad. In contrast, an automatic input might be done by a pulse-tracking bracelet that sends its data to the smartphone app without any user interaction. In the latter case, a third agent – apart from the user and the app – is present. An example provenance graph has been created for each of these cases (see Figure 4.25).


qs:user/regina@example.org
prov:label Regina Struminski
prov:type prov:Person
wasAttributedTo
actedOnBehalfOf
userdata:heartratesThen
qs:app/heartmonitor
prov:label Heart rates before input
prov:type heartrate
prov:label HeartMonitor
prov:type prov:SoftwareAgent
qs:device smartphone
used
wasAssociatedWith
prov:time 2016-12-01T16:06:21+00:00
method:input
prov:role providing input
prov:startTime 2016-12-01T16:06:21+00:00
prov:endTime 2016-12-01T16:06:22+00:00
wasDerivedFrom
wasGeneratedBy
userdata:heartratesNow
prov:time 2016-12-01T16:06:22+00:00
prov:role updating
prov:label Heart rates after input
prov:type heartrate
qs:device/pulsetracker
prov:label Pulse tracking bracelet
prov:type qs:device
qs:device bracelet

(a) A Person manually inputting data.

(b) A tracking device automatically inputting data.

Figure 4.25: Example provenance graphs for the *Input* sub-model.

Export

For the *Export* model, only one example document was needed: A user instructs a software application to export his existing step data to a CSV file (see Figure 4.26).

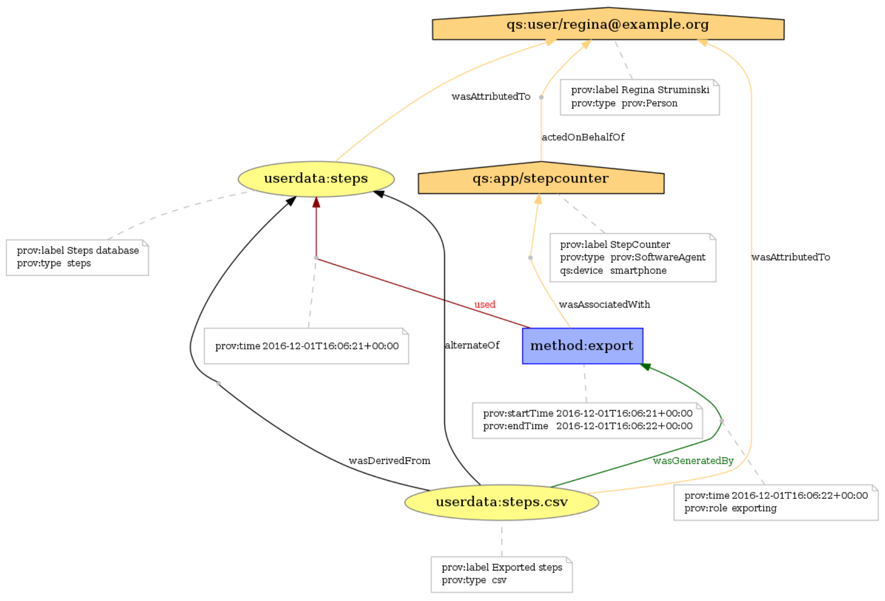

qs:user/regina@example.org
wasAttributedTo
prov:label Regina Struminski
prov:type prov:Person
actedOnBehalfOf
userdata:steps
qs:app/stepcounter
prov:label Steps database
prov:type steps
prov:label StepCounter
prov:type prov:SoftwareAgent
qs:device smartphone
wasAttributedTo
used
wasAssociatedWith
prov:time 2016-12-01T16:06:21+00:00
alternateOf
method:export
prov:startTime 2016-12-01T16:06:21+00:00
prov:endTime 2016-12-01T16:06:22+00:00
wasDerivedFrom
wasGeneratedBy
userdata:steps.csv
prov:time 2016-12-01T16:06:22+00:00
prov:role exporting
prov:label Exported steps
prov:type csv

Figure 4.26: Example provenance graph for the *Export* sub-model.

Aggregate

The *Aggregate* example file features two data entities attributed to a user, which contain steps and pulse data respectively. The user instructs a software application to export his existing step data data to a CSV file (see Figure 4.27).

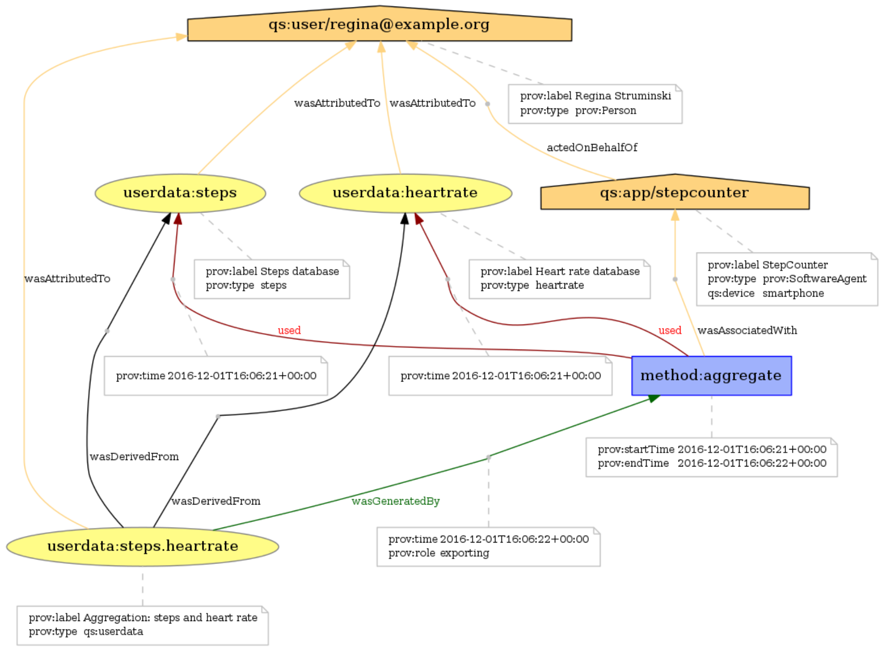
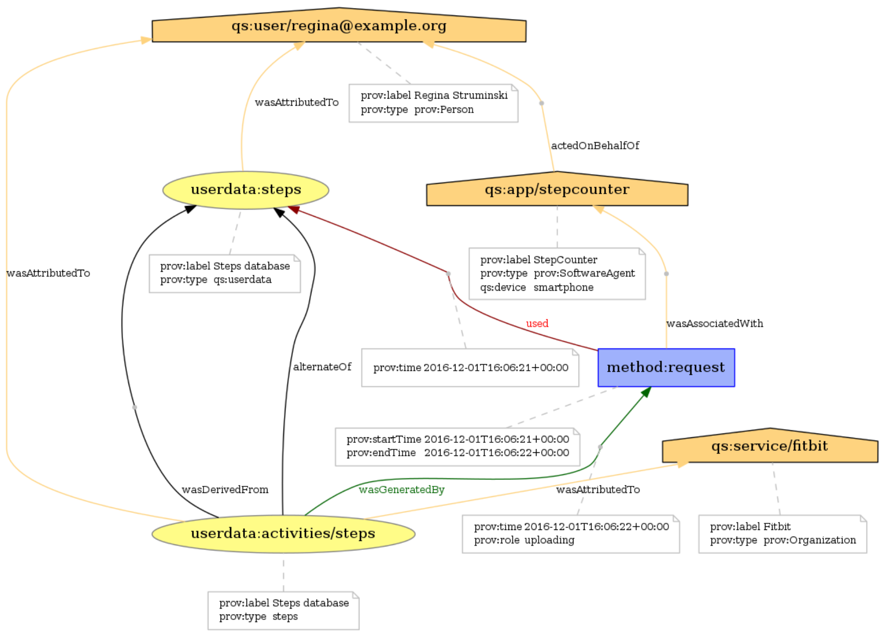

qs:user/regina@example.org
wasAttributedTo
wasAttributedTo
prov:label Regina Struminski
prov:type prov:Person
actedOnBehalfOf
userdata:steps
userdata:heartrate
qs:app/stepcounter
wasAttributedTo
prov:label Steps database
prov:type steps
prov:label Heart rate database
prov:type heartrate
prov:label StepCounter
prov:type prov:SoftwareAgent
qs:device smartphone
used
used
wasAssociatedWith
prov:time 2016-12-01T16:06:21+00:00
prov:time 2016-12-01T16:06:21+00:00
method:aggregate
prov:startTime 2016-12-01T16:06:21+00:00
prov:endTime 2016-12-01T16:06:22+00:00
wasDerivedFrom
wasDerivedFrom
wasGeneratedBy
userdata:steps.heartrate
prov:time 2016-12-01T16:06:22+00:00
prov:role exporting
prov:label Aggregation: steps and heart rate
prov:type qs:userdata

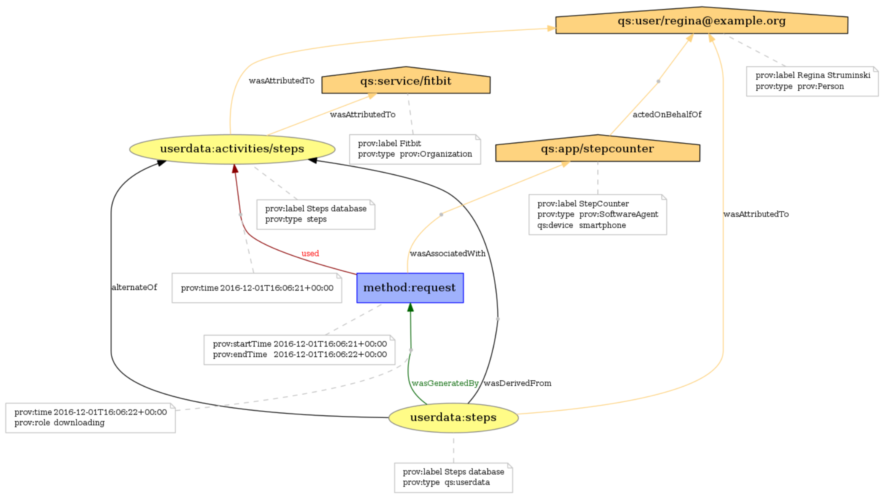
Figure 4.27: Example provenance graph for the *Aggregate* sub-model.

Request

Requests differ slightly depending on whether they are upload or download requests. Although these differences are only very small, they have a huge impact on how the request will be depicted in the comics. Thus, two example documents were created for this sub-model: one of a user having the software application upload step data to FITBIT, and one downloading step data from FITBIT back to the application device.


qs:user/regina@example.org
prov:label Regina Struminski
prov:type prov:Person
wasAttributedTo
actedOnBehalfOf
userdata:steps
qs:app/stepcounter
prov:label Steps database
prov:type qs:userdata
prov:label StepCounter
prov:type prov:SoftwareAgent
qs:device smartphone
wasAttributedTo
used
wasAssociatedWith
alternateOf
prov:time 2016-12-01T16:06:21+00:00
method:request
prov:startTime 2016-12-01T16:06:21+00:00
prov:endTime 2016-12-01T16:06:22+00:00
qs:service/fitbit
wasDerivedFrom
wasGeneratedBy
wasAttributedTo
userdata:activities/steps
prov:time 2016-12-01T16:06:22+00:00
prov:role uploading
prov:label Fitbit
prov:type prov:Organization
prov:label Steps database
prov:type steps

(a) Step data being uploaded to FITBIT.


qs:user/regina@example.org
prov:label Regina Struminski
prov:type prov:Person
wasAttributedTo
qs:service/fitbit
actedOnBehalfOf
wasAttributedTo
userdata:activities/steps
prov:label Fitbit
prov:type prov:Organization
qs:app/stepcounter
prov:label Steps database
prov:type steps
prov:label StepCounter
prov:type prov:SoftwareAgent
qs:device smartphone
wasAttributedTo
used
wasAssociatedWith
alternateOf
prov:time 2016-12-01T16:06:21+00:00
method:request
prov:startTime 2016-12-01T16:06:21+00:00
prov:endTime 2016-12-01T16:06:22+00:00
wasGeneratedBy
wasDerivedFrom
prov:time 2016-12-01T16:06:22+00:00
prov:role downloading
userdata:steps
prov:label Steps database
prov:type qs:userdata

(b) Step data being downloaded from FITBIT.

Figure 4.28: Example provenance graphs for the *Request* sub-model.

Visualize

The *Visualize* example is very similar to the *Export* example: A user instructs a software application to create a new entity, namely a line chart diagram, from existing step data (see Figure 4.29).

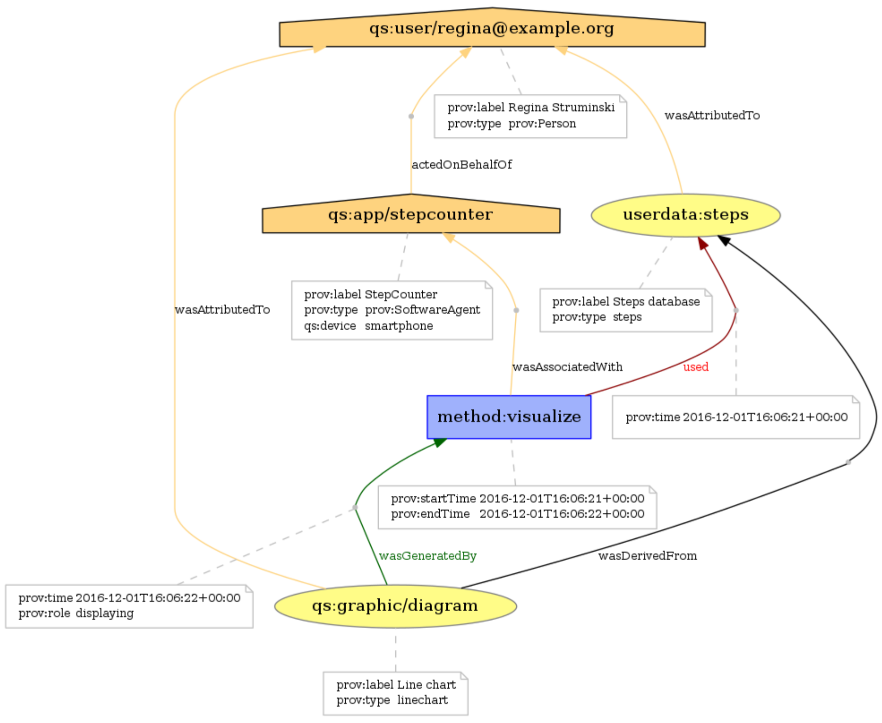


Figure 4.29: Example provenance graph for the *Visualize* sub-model.

5 Reading study

A small user study was conducted to evaluate the clarity and comprehensibility of the PROV Comics. Ten test subjects were shown a number of comics and asked to re-narrate the story as they understood it. Section 5.2 describes the comics that were shown to the test persons, while section 5.1 details the planning and conduct of the study. Section 5.3 compiles the results and draws some conclusions.

5.1 Study design and conduct

In order to construct a useful survey, a few considerations need to be made. These include the question that the study should answer, the basic type of study (qualitative or quantitative), the timing of the inquiries, the selection and number of test subjects, and the tools or instruments that are to be used. (Richter and Flückiger, 2013, p. 90) The decisions made for each of these aspects, as well as the construction of the questionnaires and the actual conduct of the study are described below.

5.1.1 Research question

The general question that was to be answered by the study is whether the PROV Comics are comprehensible to average end users: Are the selected graphics and the visual language they form understandable? Do users understand the history of their own data, i.e. when and how their data originated, what conversions and transformations it underwent, and who had access to or control over it in the course of time?

The study was also to reveal misunderstandings that may arise from a lack of technical knowledge on the reader's part and help determine passages where the images are not explanatory enough and need to be improved or extended.

5.1.2 Qualitative vs. quantitative

Quantitative studies usually collect numerical data that can easily be compared and analyzed using statistics tools. Since questions are rather closed and procedures are very standardized, large numbers of test subjects may be interviewed, thus yielding a large amount of data. However, not all scientific subjects can be measured in numbers; for these subjects, a quantitative study may not be the right choice.
Qualitative studies generally involve less participants. Questions are rather open and procedures are flexible and exploratory. Instead of numerical data, results consist of backgrounds, relations, connections, causes, and even subjective statements that participants make. As a consequence, results may be more detailed and individual than in a quantitative study, but their analysis requires much more effort, because qualitative content analyses have to be done.

It was decided that a qualitative study was the better choice in order to find out whether or not the Prov Comics are comprehensible. Different people may understand the comics in different ways, or have different problems when reading them. These can hardly be compared or measured in numbers, and creating a standardized questionnaire with closed questions would have been very difficult. Moreover, it would probably have led to further problems; for example, if asking about certain features of the comics using single or multiple choice questions, the question itself as well as the available answers might have provided hints and suggested something to the participants that they actually did not understand by themselves when they first read the comics.

Due to these considerations, it was deemed reasonable to have test readers speak freely about the comics and perform a qualitative analysis afterwards. However, to later make the test readers' answers accessible to statistics and comparison, a list was created for each of the comics, containing 10 to 23 findings that participants might discover and verbalize. It was thus possible to gain quantitative data by calculating the percentage of discovered findings.

5.1.3 Questionnaire construction

Further aspects that are relevant when creating a survey or questionnaire include, for example, the type of questions, the use of scales, instructions for the test person, etc. (Richter and Flückiger, 2013, p. 90f)

Since the decision had been made for a qualitative study, only five questions regarding statistical information were of quantitative or closed nature (e.g. gender, age, level of technical experience; see appendix N). For these closed questions, no detailed instructions were necessary, as they were rather self-explaining. Examples were given to clarify the different options in single- and multipe-choice questions. Participants' technical experience was measured on a four-part scale ranging from "no experience" to "expert", so the answers could later be represented by numbers in order to calculate the average technical experience. Answering these introductory questions was expected to take no longer than five minutes.

The large rest of the inquiry consisted of the single open question of how participants interpreted the comics. Five comics were prepared, each displaying a series of activities that took place using a different software and/or in a different field of the quantified-self domain (e.g. body weight, blood pressure, step counter). For each of the comics, an introduction as well as some instructions were given. The introduction was written on an extra sheet of paper that preceded each comic. It was expected that reading all introductions and comics as well as re-narrating their stories would take about 20 minutes.

Further details about the test comics will be given in section 5.2.

5.1.4 Timing

Test readers were interviewed one at a time, and each reader was interviewed only once; there were no repeated interviews with the same persons. Test subjects were not divided into groups for parallel examination; instead, all participants were shown the same comics in the same order. The interviews took about thirty minutes each and were conducted over a period of several days.

5.1.5 Selection of test subjects

No special background was required of the test persons; on the contrary, it was desired that they have no previous knowledge about data provenance and no special expertise in the Quantified-Self domain. No limitations were set in terms of age, gender, or occupation.

Participants were chosen from the group of possible future readers of Quantified-Self-related PROV Comics. Thus, they needed to be possible users of health or fitness tracking applications. However, it did not matter whether they actually had previous experience in the Quantified-Self domain. Participants were desired to be at least casual users of smartphone and/or desktop apps on a beginner level.

As for the number of test readers, values common in the field of usability testing were taken as a guideline: For qualitative statements that lead to first insights as to where improvements and adjustments need to be made, a number of four to six test subjects is sufficient; for systems with higher requirements, seven to fifteen persons should be questioned. (Richter and Flückiger, 2013, p. 86) Based on these values, a total of ten test subjects was interviewed, four of which were female. The average age was 37.7 years.

5.1.6 Tasks, rules and instruments

For each participant, all sheets shown in Appendices N through S were printed out and handed to them on paper. Participants were not told anything about the actual subject of the comics (data provenance).

In order to obtain comparable results, all test subjects were asked to fulfill the exact same tasks for each of the five comics in two phases: a **reading** and a **conversation phase**. In the reading phase, test persons were asked to first read the introduction to the scenario they should imagine themselves in when reading the comic (i.e. using a quantified-self app and then making use of a "Origin of this data" feature). After that, they were encouraged to take their time reading the comic and signal when they felt they had understood the story. In the conversation phase, participants were asked to re-narrate the comic's story in a detailed manner. This procedure was conducted for all five comics with each test person.

To avoid influencing the interpretation process in any way, the examiner did not talk to participants during the reading phase. This was to ensure that test persons were not given any hints or explanations that might have biased the results. In the subsequent conversation phase, however, the examiner was allowed to ask clarifying questions about the participant's interpretation of the comic, if necessary.

A smartphone running a dictaphone app was used to record the participants' re-narrations of the comics. Any further comments and notes on certain parts of the comics were written down directly on the paper, near the panel in question. The recordings were later used to assess which findings participants verbalized when telling the story.

5.1.7 Debriefing

After all comics had been worked through, any difficult parts were revisited and analyzed in an informal conversation. Participants were encouraged to comment freely on the comics, giving their own opinion and suggestions for improvements. They were also asked as to what they thought the actual point of the comics might be.

5.2 Test comics

Note that some of these test comics do not yet fulfill all of the design rules and guidelines determined in the concept (section 4.2). That is because the test comics were partially developed in parallel to the concept, when some of the rules had not been postulated yet. In fact, some rules were only established after the first interviews had already been conducted and revealed problematic parts.

The first three test comics each depict a combination of two use cases (e.g. *Input* and *Visualize*). The fourth and fifth comics are a little longer, combining three to four use cases.

5.2.1 Comic #1

In the introductory text, test persons are asked to imagine that they wear a **step counting bracelet** everyday. A smartphone app allows users to sync data with the bracelet and **view the steps they have taken**. They now press a button labeled "Origin of this data" in the app, whereupon the comic on the next page is displayed.

The comic is a combination of the *Input* and *Request* (upload) use cases. It shows a user walking while wearing a step tracking bracelet on his arm. He then takes his smartphone, and the data is transmitted from the bracelet to the phone. The data is then uploaded to FITBIT, so in the end, both the user and the company have a copy of the data.

The comic including the introduction can be seen in appendix O. A list of expected findings is given in appendix U.

5.2.2 Comic #2

In the introductory text, test persons are asked to imagine that they wear a **sleep tracking bracelet** everyday. A smartphone app allows users to sync data with the bracelet and **view their sleep data** anytime. They now press a button labeled "Origin of this data" in the app, whereupon the comic on the next page is displayed.

The comic is a combination of the *Input* and *Export* use cases. It shows a user sleeping while wearing a sleep tracking bracelet on his arm. The next morning, he takes his smartphone, and the data is transmitted from the bracelet to the phone. An option to export the sleep data in CSV format appears; the user taps that option, and a CSV file is created on the phone.

For the sake of testing, one of the alternative color shades for the human silhouette as used, even though there is no second person present in this comic.

The comic including the introduction can be seen in appendix P. A list of expected findings is given in appendix V.

5.2.3 Comic #3

In the introductory text, test persons are asked to imagine that they own a **WiFi-enabled bodyweight scale**. A smartphone app allows users to sync data with the scale and **view their weight** anytime. They now press a button labeled "Origin of this data" in the app, whereupon the comic on the next page is displayed.

The comic is a combination of the *Input* and *Visualize* use cases. It shows a user stepping on a bodyweight scale. He then takes his smartphone, and the data is transmitted from the scale to the phone. An option to view the weight data in CSV format appears; the user taps that option, and a line chart is displayed on the phone.

The comic including the introduction can be seen in appendix Q. A list of expected findings is given in appendix W.

5.2.4 Comic #4

In the introductory text, test persons are asked to imagine that they wear a **fitness bracelet** everyday, which is able to track steps as well as heart rates. A smartphone app allows users to sync data with the bracelet and **view their steps and heart rates** anytime. They now press a button labeled “Origin of this data” in the app, whereupon the comic on the next page is displayed.

The comic is a combination of the *Request* (download), *Input*, *Aggregate*, and *Visualize* use cases. It shows a user downloading existing step data from FITBIT to his smartphone, since the data on his phone is out of date or missing. Later he also transfers pulse data from his bracelet to the phone. An option to combine steps and pulse data appears; the user taps that option, and a new set of data is created on the phone. An option to view that new data set appears; the user taps that option, and a line chart is displayed on the phone.

The comic including the introduction can be seen in appendix R. A list of expected findings is given in appendix X.

5.2.5 Comic #5

In the introductory text, test persons are asked to imagine that they regularly measure their **blood pressure** using a conventional blood pressure meter. A computer app allows them to **input and view their blood pressure values**. Moreover, they also have the mobile version of that app installed on their smartphone. They now press a button labeled “Origin of this data” in the smartphone app, whereupon the comic on the next page is displayed.

The comic is a combination of the *Input*, *Request* (upload), and *Request* (download) use cases. It shows a user measuring his blood pressure and then saving the values in a computer software. The data is then uploaded to the (fictional) organization MEDITEC, so in the end, both the user and the company have a copy of the data. Later the user switches to his smartphone and downloads his blood pressure data from MEDITEC, since the data on his phone is out of date or missing.

The comic including the introduction can be seen in appendix S. A list of expected findings is given in appendix Y.

5.3 Results

5.3.1 Assessment of re-narrations

As mentioned earlier, a list had been created for each of the test comics, containing possible findings that test readers might verbalize. Listening to the previously recorded re-narrations, tables were created to note which findings each test reader mentioned. Each finding was valued with a score of 1.
In some cases, findings were only hinted at, or readers used an unfortunate choice of words, but basically meant the right thing. For example, one user referred to the sleep tracker as

a movement tracker, which is what most sleep trackers really are. Findings like these were given a score of 0.5.
Additionally, every list contained a bonus finding: something that is not depicted in the comic, but might have been concluded by the readers. These bonus findings did not count towards the "findings score".
Based on these scores, the percentage of findings discovered by each test reader was calculated.

The findings tables are shown in Appendices U through Y. Another evaluative table containing comparisons between *male vs. female* as well as *older vs. younger* (i.e. below vs. above average age) participants can be seen in appendix Z.

5.3.2 Overall result

In general, it can be said that all five test comics were understood well by the test readers. In some cases, there were certain difficult parts that mostly stemmed from a lack of experience with Quantified-Self applications or web services in general. However, the essences of the overall stories were largely interpreted correctly.

No distinct correlations were found between age or technical and Quantified-Self experience and the number of findings expressed by the test persons. Interestingly, women performed better – sometimes by far – for all five comics (see appendix Z). However, the number of test subjects in this small study is too low to draw any general conclusions from that. It seemed that women always mentioned *any* details they could find in a comic, while men tended to skip findings that they deemed self-evident or repetitive.

Participants had no difficulties recognizing and interpreting the different icons for concrete elements, like persons, smartphones, computers, and bracelets/smartwatches. Even the more abstract notions did not pose many problems. As mentioned earlier in section 4.2.8, commonly used symbols were employed to depict notions like *"a collection of user data"* (represented by a document folder), *"transmitting data from one device to another"* (represented by a "radio waves" icon), or *"synchronizing data with a cloud"* (represented by a cloud icon with an arrow circle). The conventional use of similar icons in software and web applications over the past years or even decades has apparently integrated them into readers' "visual vocabulary" to such an extent that, for the most part, they are immediately understood without any confusion.

Readers also had no problem identifying themselves with the comic figure (human silhouette). Almost every re-narration was told from a first-person point of view, using sentences like "*I* was walking", "*I* was wearing a bracelet", "*I* clicked the button", etc.

Since the lists of findings were rather detailed, it was not expected that readers would mention 100 % of them (although they actually did in a few cases). Quite often, readers seemed to deem certain findings too obvious and self-evident to even mention them. For example, findings like "user takes/uses his smartphone" and "FITBIT has the user's data" were often omitted, but later implied by sentences like "I downloaded my data from FITBIT to my phone". Thus, even if only about 80-90 % of the findings were particularly verbalized, this was valued as a very good understanding of the story. The average percentage of findings over all five test comics was 77 % (women: 84 %; men: 73 %).

5.3.3 Comic #1

On average, 84 % of the findings in this comic were mentioned. Readers had no problems understanding that they were walking while wearing a step tracking bracelet. They also understood that they later transmitted the number of steps from the bracelet to their smartphone.

The act of uploading the data to an organization's server was clear to most, but not all readers. Some readers did not see the label on the building icon at first and thought that it was meant to be their home or some other random building. However, except for one, all participants discovered the label sooner or later and corrected their interpretations.

A more semantic problem with the label was that several readers did not know FITBIT as a provider of Quantified-Self software and gadgets. Considering the "fit" part of the name, they then supposed that the building symbolized a fitness center. However, the overall understanding of data being uploaded to a remote organization was not impaired by this.
Only one reader did not understand the "upload" panels at all, but instead thought that he was walking to or into a skyscraper to continue tracking his steps there, possibly by walking up and down the stairs.

One person mentioned the bonus finding that the purpose of uploading was the synchronization between different devices.

5.3.4 Comic #2

On average, 75 % of the findings in this comic were mentioned. Readers had no problems understanding that they were sleeping while wearing a sleep tracking bracelet. They also understood that they transmitted their sleep data from the bracelet to their smartphone on the next day.

The appearance of the export button, which had been problematic before (cf. section 4.3.4), did not cause any confusion now. The pressing of the button was also well understood.

Eight out of ten readers had no problems understanding that they exported their data to a file, and most of them assumed that this file was stored on the phone. However, one reader, who had never heard of "CSV" before, thought that the data was being sent somewhere.
Another reader, although having understood the export panels, noted that she had a different understanding of the term "export": She thought that exporting meant to send or save data *elsewhere*, rather than saving it on the phone again. For an export to a file on the phone, she would rather have expected a word like "save".

The fact that this comic used a different color for the human silhouette did not seem noteworthy to any reader. None of them mentioned it, and they identified with the person just as in the first comic.

Two persons mentioned the bonus finding that the purpose of exporting was the possibility to share data or load it into other applications. Another two readers noted this indirectly.

5.3.5 Comic #3

This comic seemed to be the clearest one with the fewest understanding problems. On average, 87 % of the findings in this comic were mentioned; three readers even mentioned 100 % of them. Several participants also stated in the informal conversation that they found this comic to be very clear and easily comprehensible.

All readers understood that their weight was measured and transmitted to the smartphone, and that they were able to view their weight history in a diagram.

One reader noted that the weight was apparently recorded in kilograms.

One person mentioned the bonus finding that the synchronization between the scale and the phone happens automatically without being induced by the user.

5.3.6 Comic #4

On average, 71 % of the findings in this comic were mentioned. All readers seemed to have understood that data was missing from the smartphone, even though not all of them explicitly mentioned it. Eight out of ten understood that the data was available at FITBIT and could be downloaded to the phone from there.

The reader who had already had difficulties with the upload in comic #1 had the same problems with the download depicted here. Again, he thought that the building was a skyscraper, or maybe a fitness center, that he was walking out of. Another reader thought it was a fitness center that she had to visit before starting her physical activities, in order to receive a workout plan or training schedule.

One reader did not recognize that the bracelet in this comic was a pulse tracker; she thought it was a step counter like in the first comic.

The word “combine”, which was chosen over “aggregate” as the button label, was understood well. Two readers (computer science students) referred to the depicted process as an “aggregation”. It seemed clear to all readers that steps and heart rates were merged into a new data set that gives an overview on the correlation of pulse and the number of steps taken.

One reader noted a logical problem in the story: According to the comic, steps have already been recorded earlier, but the pulse is being tracked later, not during walking. Thus it is not possible to aggregate them in a meaningful way. This was a conceptual error during the creation of this comic. However, it did not impact the general understanding of the depicted actions.

One person mentioned the bonus finding that the download of data from FITBIT to the phone happens automatically without being induced by the user.

5.3.7 Comic #5

On average, 71 % of the findings in this comic were mentioned by the readers. All readers understood that blood pressure was measured and a desktop computer was used afterwards. All readers except for one understood that the blood pressure values were manually entered into an application on the PC, then uploaded to an organization called MEDITEC, and finally downloaded from the organization to the smartphone. Moreover, half of the readers noted that the data was now available on both the PC and the phone, showing that they understood the actual purpose of up- and downloading, namely the synchronization of data between devices.

The reader who had already had problems with the up- and download pictures before again thought that he was walking up and down the stairs in a skyscraper. The reader who thought the building was supposed to be a fitness center in comic #4 still thought so, even though the label on the building was no longer "FITBIT", but "MEDITEC" instead. While she correctly described the upload of data to the organization, she thought that this "fitness center" evaluated her blood pressure data and then sent her an accordingly adapted fitness plan or training schedule.

One person mentioned the bonus finding that the download of data from MEDITEC to the phone happens automatically without being induced by the user.

5.3.8 Improvement measures

This section lists possible solutions to common problems that surfaced during the reading study, and also includes some of the comments, ideas, and suggestions that participants weighed in during the informal debriefing.

"Origin of this data" button

All participants where asked what they think the purpose of the comics might be. Eight out of ten thought that the comics were meant to be instruction manuals, providing hints on how to use the smartphone or computer app in combination with wearables or other measuring devices. When asked to take another look at the scenario that they were to imagine themselves in, including the "Origin of this data" button, most of them understood that the comics were about data provenance (although they did not use that term). It seemed as though they had not paid much attention to the button label before, and therefore did not have a provenance context in mind when reading the comics.
In any case, this brings forth a completely different scope of application for the comics created in this work, namely the automatic generation of instruction manuals from data documents.

One reader who *did* pay attention to the button label was confused about the number of actions depicted, like exports, aggregations, or visualizations. In his opinion, the "origin" of his data was only the moment when the data was created and input. The origin of weight data, for example, would only be the act of using the bodyweight scale and transmitting that data to the smartphone.

In summary, it seems advisable to reconsider the button label. Possibly, a text like "What happened to my data?" would more effectively establish the right context in the reader's mind before reading the comics.

Building icon

The bad recognizability of the organization labels in all of the comics led to the development of the "black or white" font color rule described in section 4.2.3. According to that rule, the labels would now have to be black. It is expected that they will not be overlooked anymore when using black font.

It should be made more clear that the building stands for a web service or cloud provided by some organization, while the actual name of that organization is not to be interpreted in any way (e.g. FITBIT → fitness center). One possibility would be to simply add the word "cloud" behind the company's name. This might shift readers' focus so that they understand that the actual name is only of secondary interest.

Weight icon

One reader explicitly said that weight data was apparently recorded in kilograms. Since this does not always have to be the case, the "kg" label on the weight icon should be reconsidered. The label was originally added to the icon to improve recognizability, because a trapezoid with a ring at the top might not have been clearly identifiable as a weight symbol.

There are several possible solutions to this:

a) Remove the label completely, and test whether the icon is still well understood.

b) Have the label show the correct unit dynamically (e.g. "kg", "lbs", ...). However, this would require the unit to be stated in the PROV document, which is currently not required by specification.

c) Discard the weight icon altogether and find a different, possibly better one instead.

Tracking device

In one situation, a reader did not recognize that the icon on a tracking bracelet had changed and heart rate was now being tracked instead of steps. Although this was not a general problem for all readers, it may be a good idea to magnify the tracking device even further in the zoomed-in view.

The term "export"

Only one reader had some minor difficulties with the term "export". However, this did not impact the overall comprehensibility of the export pictures. The expression "save as" might indicate more clearly that the *Export* activity creates a file on the same device. But since none of the other readers were confused about the term, and it did not seem to influence the general understanding, there is no urgent need to change the word.

"Missing data" icon

One user suggested that the grayed-out folder icon should be modified to indicate whether data is just outdated or missing completely. At the moment, it always seems as if no data was present at all. A solution might be keep the current symbol for the case that data is completely missing; however, when data is present, but outdated, a grayed-out version of the folder including the data type icon (e.g. footprints for steps) might be used. Additionally, there could be an icon on the folder, indicating that a synchronization is needed. A suggestion for such an icon is shown in Figure 5.1.

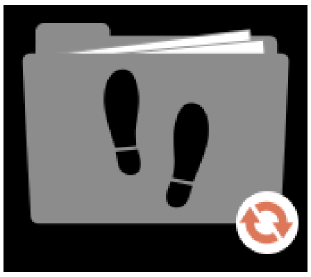


Figure 5.1: Suggestion for an "outdated data" icon

6 Conclusion and Future Work

6.1 Conclusion

The goal of this work was to develop a self-explaining, easy-to-understand visualization of data provenance that can be understood by non-expert end users of Quantified-Self apps.

A detailed concept has been created that defines a consistent visual language. Graphics for PROV elements like different agents and entities were designed, and sequences of comic panels to represent different activities were determined. Symbols, icons, and panel sequences were specified in an exact and uniform manner to enable the automatic generation of comics.

The visualization is compatible with documents compliant with the W3C PROV Recommendations, as required (see section 2.2.1). Also, the existing PROV data model was retained (see section 2.2.2), although some modifications to the model needed to be made.

As proof of concept, a prototypical website has been developed which is able to automatically generate comics from PROV documents compliant with the modified data model described in section 4.4.4. The documents are loaded from the PROVSTORE website, as required (see section 2.1).

A reading study involving ten test readers has shown that a non-expert audience is mostly able to understand the provenance of Quantified-Self data through PROV COMICS without any prior instruction or training. The overall percentage of 77 % for findings verbalized by participants is deemed a good result, given that the checklists were very detailed and contained findings that some readers probably omitted, because they seemed too obvious and self-evident to them.

6.2 Future work

6.2.1 Graphical improvements

Apart from the suggested improvement measures that resulted from the reading study (see section 5.3.8), a few other corrections may be made. Most of them are little inconsistencies that had not been noticed straight away:

- The “radio waves” icon uses a bright orange color, but, being an activity-related icon, should rather be white, as defined in section 4.2.3. However, this will bring forth a new problem: How should the white icon be painted on a white background? A border or similar boundary would be needed, which is not desired according to section 4.2.
- When transmitting data from a tracking device to a smartphone or other application device, a cloud icon with an arrow circle is displayed. Although readers had no problems interpreting this icon correctly, it still is a little inappropriate. A cloud icon is normally used when data is being stored at or retrieved from a remote location, which is not the case here. A possible alternative might be an icon that only consists of an arrow circle, without a cloud.
- In the depictions of a desktop computer, only one of the panels displays an application name (cf. comic #5 in Appendix S). However, this certain panel may not always be part of the comic. Thus, a title bar displaying the name of the application, similar to that in the smartphone graphic, should always be present.
- On the “view graph” button, the word “graph” might be replaced by “diagram”, since a graph is only one type of diagram. Other types include line charts, bar charts, pie charts, and scatter diagrams.
- To further reduce the use of textual annotations, a checkmark icon might be used instead of the words “complete” or “done” whenever an action performed by an application is finished.
- Aggregations are usually not induced by the user, but instead happen automatically when necessary. Take, for example, a user who wishes to view a “steps vs. weight” diagram: As soon as he selects that option, the aggregation is performed automatically in the background, before the actual diagram is displayed. For the sake of comprehensibility, aggregations have been depicted as if the user actively chose to perform them by pressing a “combine” button. It should be considered whether it is possible to create a version without the button press, while maintaining a good understandability at the same time.

6.2.2 Larger provenance graphs

The focus of this work was put on creating a visualization concept based on the five Quantified-Self use cases. Therefore, only single activities or rather short combinations of activities have been considered so far. However, with the help of PROV COMICS, it should be possible to visualize even longer chains of activities (i.e. larger provenance graphs) without overwhelming the reader’s mind.

For example, consider a graph with 10 activities, each involving several agents and entities. Such a graph may easily end up having 40 or 50 nodes, which is way more than can be handled by immediate memory (cf. section 1.2). Prov Comics solve the "7±2 problem" by moving the activities into the time domain and displaying each activity individually, one after another, in about 40 panels (three to five for each activity).

A question that arises is whether the comics will eventually become too long to still be useful. There might be a point at which viewing a conventional provenance graph is actually less tedious than reading a very long comic. The concept should be further tested and evaluated with large provenance graphs.

6.2.3 Domain agnosticism

A most useful improvement of the Prov Comics would be to make them application-generic to some extent, similar to the PROVglish architecture mentioned in section 3.2. Although PROVglish is about textual representations of provenance data, its domain agnosticism is of interest for graphical visualizations as well. However, there is a huge difficulty with that, namely the "media gap" between text an graphics. It is relatively easy to create a meaningful sequence of words (i.e. a sentence) out of a certain pool of words. But how is a program supposed to know what kind of graphic might be suitable to represent a certain word?

This work was only concerned with visualizations from a very specific domain (Quantified-Self applications), so it was possible to pre-assemble graphics for all elements that may occur in the provenance documents. For example, if an agent's `device` attribute is "`smartphone`", the correct corresponding picture can easily be picked and displayed, since a "smartphone" graphic has been created in advance for exactly this case.
But if the documents may come from *any* domain of application, then they may also contain *any* kinds of words. Also, it is not granted that they would be English words. In fact, they might not even be words at all; a provenance document might as well use fantasy words, abbreviations, encoded expressions, etc. No program can possibly know how to depict *any and all* possible descriptive words found in provenance documents.

It would generally be useful to examine more general and application-generic ways to create graphical representations of provenance. Bridging the "media gap", however, poses a big problem for developing such a domain-agnostic visualization.

References

[Anand et al. 2010] ANAND, Manish K. ; BOWERS, Shawn ; ALTINTAS, Ilkay ; LUDÄSCHER, Bertram: Approaches for Exploring and Querying Scientific Workflow Provenance Graphs. Version: 2010. http://dx.doi.org/10.1007/978-3-642-17819-1{_}3. In: HUTCHISON, David (Hrsg.) ; KANADE, Takeo (Hrsg.) ; KITTLER, Josef (Hrsg.) ; KLEINBERG, Jon M. (Hrsg.) ; MATTERN, Friedemann (Hrsg.) ; MITCHELL, John C. (Hrsg.) ; NAOR, Moni (Hrsg.) ; NIERSTRASZ, Oscar (Hrsg.) ; PANDU RANGAN, C. (Hrsg.) ; STEFFEN, Bernhard (Hrsg.) ; SUDAN, Madhu (Hrsg.) ; TERZOPOULOS, Demetri (Hrsg.) ; TYGAR, Doug (Hrsg.) ; VARDI, Moshe Y. (Hrsg.) ; WEIKUM, Gerhard (Hrsg.) ; MCGUINNESS, Deborah L. (Hrsg.) ; MICHAELIS, James R. (Hrsg.) ; MOREAU, Luc (Hrsg.): *Provenance and Annotation of Data and Processes* Bd. 6378. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. – DOI 10.1007/978–3–642–17819–1_3. – ISBN 978–3–642–17818–4, S. 17–26

[Bach et al. 2016] BACH, Benjamin ; KERRACHER, Natalie ; HALL, Kyle W. ; CARPENDALE, Sheelagh ; KENNEDY, Jessie ; HENRY RICHE, Nathalie: Telling Stories about Dynamic Networks with Graph Comics. In: KAYE, Jofish (Hrsg.) ; DRUIN, Allison (Hrsg.) ; LAMPE, Cliff (Hrsg.) ; MORRIS, Dan (Hrsg.) ; HOURCADE, Juan P. (Hrsg.): *The 2016 CHI Conference*, 2016, S. 3670–3682

[Biton et al. 2008] BITON, Olivier ; COHEN-BOULAKIA, Sarah ; DAVIDSON, Susan B. ; HARA, Carmem S.: Querying and Managing Provenance through User Views in Scientific Workflows. In: *2008 IEEE 24th International Conference on Data Engineering (ICDE 2008)*, 2008, S. 1072–1081

[Böhringer et al. 2008] BÖHRINGER, Joachim ; BÜHLER, Peter ; SCHLAICH, Patrick: *Kompendium der Mediengestaltung: Konzeption und Gestaltung für Digital- und Printmedien*. 4. Aufl. s.l. : Springer-Verlag, 2008 (X.media.press). http://gbv.eblib.com/patron/FullRecord.aspx?p=367414. – ISBN 978–3–540–78526–2

[Borkin et al. 2013] BORKIN, Michelle A. ; YEH, Chelsea S. ; BOYD, Madelaine ; MACKO, Peter ; GAJOS, Krzysztof Z. ; SELTZER, Margo ; PFISTER, Hanspeter: Evaluation of filesystem provenance visualization tools. In: *IEEE transactions on visualization and computer graphics* 19 (2013), Nr. 12, S. 2476–2485. http://dx.doi.org/10.1109/TVCG.2013.155. – DOI 10.1109/TVCG.2013.155. – ISSN 1077–2626

[Chen et al. 2012] CHEN, Peng ; PLALE, Beth ; CHEAH, You-Wei ; GHOSHAL, Devarshi ; JENSEN, Scott ; LUO, Yuan: Visualization of network data provenance. In: *2012 19th International Conference on High Performance Computing (HiPC)*, 2012, S. 1–9

[Cooper et al. 2016a] COOPER, Michael ; KIRKPATRICK, Andrew ; O CONNOR, Joshue: *Understanding WCAG 2.0: Contrast (Minimum)*. https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-contrast.html. Version: 2016, Last accessed on: 2017-01-08

[Cooper et al. 2016b] COOPER, Michael ; KIRKPATRICK, Andrew ; O CONNOR, Joshue: *Understanding WCAG 2.0: Use of Color*. https://www.w3.org/TR/UNDERSTANDING-WCAG20/visual-audio-contrast-without-color.html. Version: 2016, Last accessed on: 2017-01-09

[Freire et al. 2006] FREIRE, Juliana ; SILVA, Cláudio T. ; CALLAHAN, Steven P. ; SANTOS, Emanuele ; SCHEIDEGGER, Carlos E. ; VO, Huy T.: Managing Rapidly-Evolving Scientific Workflows. Version: 2006. `http://dx.doi.org/10.1007/11890850{_}2`. In: MOREAU, Luc (Hrsg.) ; FOSTER, Ian (Hrsg.): *Provenance and Annotation of Data: International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. – DOI 10.1007/11890850_2. – ISBN 978–3–540–46303–0, 10–18

[Gil et al. 2010] GIL, Yolanda ; CHENEY, James ; GROTH, Paul ; HARTIG, Olaf ; MILES, Simon ; MOREAU, Luc ; PINHEIRO DA SILVA, Paulo ; W3C (Hrsg.): *Provenance XG Final Report*. `https://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/`. Version: 2010, Last accessed on: 2016-10-13

[Groth and Moreau 2013] GROTH, Paul ; MOREAU, Luc ; W3C (Hrsg.): *PROV-Overview: An Overview of the PROV Family of Documents*. `https://www.w3.org/TR/prov-overview/`. Version: 2013, Last accessed on: 2016-12-28

[Hunter and Cheung 2007] HUNTER, Jane ; CHEUNG, Kwok: Provenance Explorer-a graphical interface for constructing scientific publication packages from provenance trails. In: *International Journal on Digital Libraries* 7 (2007), Nr. 1-2, S. 99–107. `http://dx.doi.org/10.1007/s00799-007-0018-5`. – DOI 10.1007/s00799–007–0018–5. – ISSN 1432–5012

[Huynh and Moreau 2015] HUYNH, Trung D. ; MOREAU, Luc: ProvStore: A Public Provenance Repository. Version: 2015. `http://dx.doi.org/10.1007/978-3-319-16462-5{_}32`. In: LUDÄSCHER, Bertram (Hrsg.) ; PLALE, Beth (Hrsg.): *Provenance and Annotation of Data and Processes* Bd. 8628. Cham : Springer International Publishing, 2015. – DOI 10.1007/978–3–319–16462–5_32. – ISBN 978–3–319–16461–8, 275–277

[Lebo et al. 2013] LEBO, Timothy ; SAHOO, Satya ; MCGUINNESS, Deborah L.: *PROV-O: The PROV Ontology*. `https://www.w3.org/TR/prov-o/`. Version: 2013, Last accessed on: 2017-01-11

[Macko 2011] MACKO, Peter: Provenance map orbiter: Interactive exploration of large provenance graphs. In: *Proceedings of the 3rd Workshop on the Theory and Practice of Provenance (TaPP)*. 2011

[McCloud 2011] MCCLOUD, Scott: *Understanding comics: The invisible art*. 39. [print.]. New York : HarperPerennial, 2011. – ISBN 0–06–097625–X

[Miller 1956] MILLER, George A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. In: *Psychological Review* 63 (1956), Nr. 2, S. 81–97. `http://dx.doi.org/10.1037/h0043158`. – DOI 10.1037/h0043158. – ISSN 0033–295X

[Moreau et al. 2008] MOREAU, Luc ; FREIRE, Juliana ; FUTRELLE, Joe ; MCGRATH, Robert E. ; MYERS, Jim ; PAULSON, Patrick: The Open Provenance Model: An Overview. Version: 2008. `http://dx.doi.org/10.1007/978-3-540-89965-5{_}31`. In: FREIRE, Juliana (Hrsg.) ; KOOP, David (Hrsg.) ; MOREAU, Luc (Hrsg.): *Provenance and Annotation of Data and Processes: Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected*

Papers. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – DOI 10.1007/978–3–540–89965–5_31. – ISBN 978–3–540–89965–5, 323–326

[Moreau and Missier 2013] MOREAU, Luc ; MISSIER, Paolo: *PROV-DM: The PROV Data Model*. `https://www.w3.org/TR/prov-dm/`. Version: 2013, Last accessed on: 2016-12-28

[Myers et al. 2003] MYERS, J. ; PANCERELLA, C. ; LANSING, C. ; SCHUCHARDT, K. ; DIDIER, B.: Multi-Scale Science, Supporting Emerging Practice with Semantically Derived Provenance. In: *ISWC workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, 2003

[Richardson and Moreau 2016] RICHARDSON, Darren P. ; MOREAU, Luc: Towards the Domain Agnostic Generation of Natural Language Explanations from Provenance Graphs for Casual Users. Version: 2016. `http://dx.doi.org/10.1007/978-3-319-40593-3{_}8`. In: MATTOSO, Marta (Hrsg.) ; GLAVIC, Boris (Hrsg.): *Provenance and Annotation of Data and Processes* Bd. 9672. Cham : Springer International Publishing, 2016. – DOI 10.1007/978–3–319–40593–3_8. – ISBN 978–3–319–40592–6, S. 95–106

[Richter and Flückiger 2013] RICHTER, Michael ; FLÜCKIGER, Markus D.: *Usability Engineering kompakt: Benutzbare Produkte gezielt entwickeln*. 3. Aufl. 2013. Berlin and Heidelberg : Springer, 2013 (IT kompakt). `http://dx.doi.org/10.1007/978-3-642-34832-7`. `http://dx.doi.org/10.1007/978-3-642-34832-7`. – ISBN 978–3–642–34832–7

[Rio and al 2007] RIO, Nicholas D. ; AL, et: *Identifying and Explaining Map Imperfections Through Knowledge Provenance Visualization*. 2007

[Schreiber 2016] SCHREIBER, Andreas: A Provenance Model for Quantified Self Data. Version: 2016. `http://dx.doi.org/10.1007/978-3-319-40250-5{_}37`. In: ANTONA, Margherita (Hrsg.) ; STEPHANIDIS, Constantine (Hrsg.): *Universal Access in Human-Computer Interaction. Methods, Techniques, and Best Practices* Bd. 9737. Cham : Springer International Publishing, 2016. – DOI 10.1007/978–3–319–40250–5_37. – ISBN 978–3–319–40249–9, S. 382–393

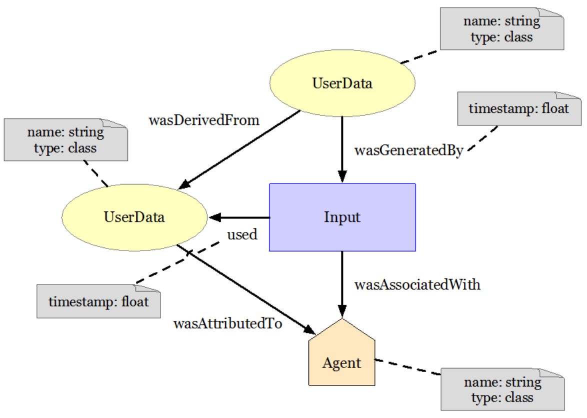
[Simmhan et al. 2005] SIMMHAN, Yogesh L. ; PLALE, Beth ; GANNON, Dennis: A survey of data provenance in e-science. In: *ACM SIGMOD Record* 34 (2005), Nr. 3, S. 31. `http://dx.doi.org/10.1145/1084805.1084812`. – DOI 10.1145/1084805.1084812. – ISSN 01635808

Appendices

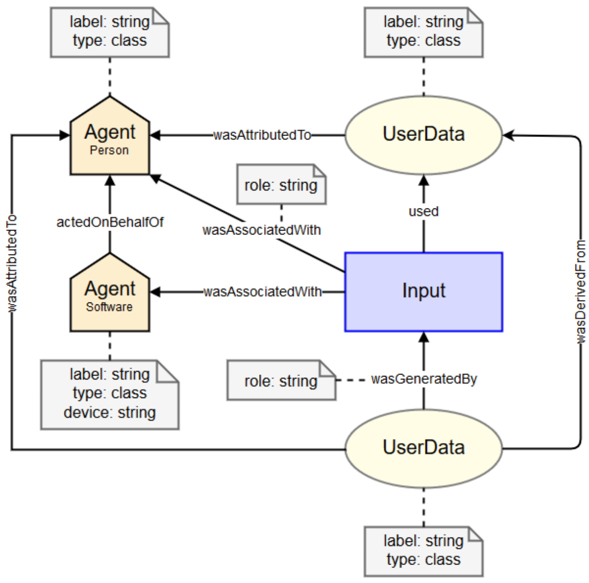
A An example provenance graph in XML notation

```xml
<?xml version='1.0' encoding='ASCII'?>
<prov:document xmlns:app="http://software.dlr.de/qs/app/" xmlns:device="http://software.dlr.de
/qs/device/" xmlns:method="http://www.java.com" xmlns:prov="http://www.w3.org/ns/prov#"
xmlns:qs="http://software.dlr.de/qs/" xmlns:user="http://software.dlr.de/qs/user/"
xmlns:userdata="http://software.dlr.de/qs/userdata/" xmlns:xsd="http://www.w3.org
/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <prov:wasAssociatedWith>
    <prov:activity prov:ref="method:input"/>
    <prov:agent prov:ref="qs:app/heartmonitor"/>
  </prov:wasAssociatedWith>
  <prov:wasAssociatedWith>
    <prov:activity prov:ref="method:input"/>
    <prov:agent prov:ref="qs:device/pulsetracker"/>
    <prov:role>providing input</prov:role>
  </prov:wasAssociatedWith>
  <prov:wasGeneratedBy>
    <prov:entity prov:ref="userdata:heartratesNow"/>
    <prov:activity prov:ref="method:input"/>
    <prov:time>2016-12-01T16:06:22+00:00</prov:time>
    <prov:role>updating</prov:role>
  </prov:wasGeneratedBy>
  <prov:activity prov:id="method:input">
    <prov:startTime>2016-12-01T16:06:21+00:00</prov:startTime>
    <prov:endTime>2016-12-01T16:06:22+00:00</prov:endTime>
  </prov:activity>
  <prov:entity prov:id="userdata:heartratesNow">
    <prov:label>Heart rates after input</prov:label>
    <prov:type xsi:type="xsd:string">heartrate</prov:type>
  </prov:entity>
  <prov:entity prov:id="userdata:heartratesThen">
    <prov:label>Heart rates before input</prov:label>
    <prov:type xsi:type="xsd:string">heartrate</prov:type>
  </prov:entity>
  <prov:agent prov:id="qs:app/heartmonitor">
    <prov:label>HeartMonitor</prov:label>
    <prov:type>prov:SoftwareAgent</prov:type>
    <qs:device>smartphone</qs:device>
  </prov:agent>
  <prov:agent prov:id="qs:device/pulsetracker">
    <prov:label>Pulse tracking bracelet</prov:label>
    <prov:type xsi:type="xsd:string">qs:device</prov:type>
    <qs:device>bracelet</qs:device>
  </prov:agent>
  <prov:agent prov:id="qs:user/regina@example.org">
    <prov:label>Regina Struminski</prov:label>
    <prov:type>prov:Person</prov:type>
  </prov:agent>
  <prov:wasAttributedTo>
    <prov:entity prov:ref="userdata:heartratesNow"/>
    <prov:agent prov:ref="qs:user/regina@example.org"/>
  </prov:wasAttributedTo>
  <prov:wasAttributedTo>
    <prov:entity prov:ref="userdata:heartratesThen"/>
    <prov:agent prov:ref="qs:user/regina@example.org"/>
  </prov:wasAttributedTo>
  <prov:actedOnBehalfOf>
    <prov:delegate prov:ref="qs:app/heartmonitor"/>
    <prov:responsible prov:ref="qs:user/regina@example.org"/>
  </prov:actedOnBehalfOf>
  <prov:actedOnBehalfOf>
    <prov:delegate prov:ref="qs:device/pulsetracker"/>
    <prov:responsible prov:ref="qs:user/regina@example.org"/>
  </prov:actedOnBehalfOf>
  <prov:used>
    <prov:activity prov:ref="method:input"/>
    <prov:entity prov:ref="userdata:heartratesThen"/>
    <prov:time>2016-12-01T16:06:21+00:00</prov:time>
  </prov:used>
  <prov:wasDerivedFrom>
    <prov:generatedEntity prov:ref="userdata:heartratesNow"/>
    <prov:usedEntity prov:ref="userdata:heartratesThen"/>
  </prov:wasDerivedFrom>
</prov:document>
```

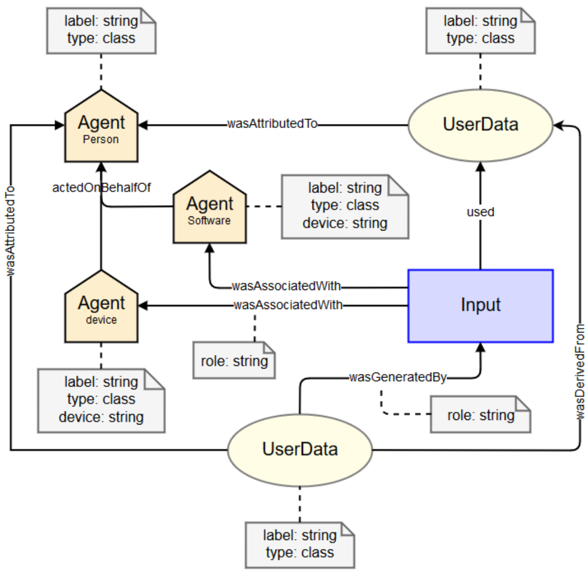
B Adjustments to the input sub-model



(a) Old input sub-model



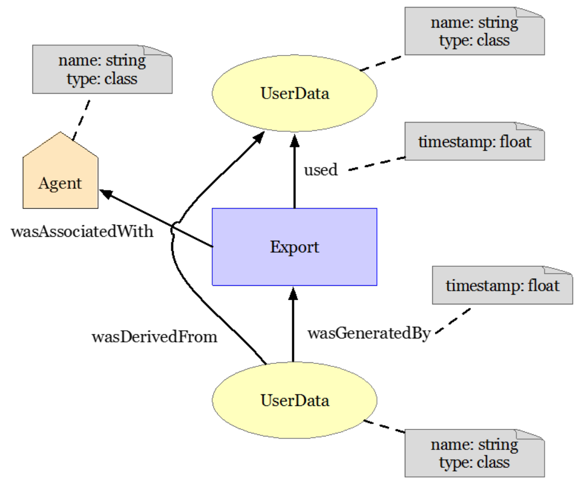
(b) New input sub-model with two agents



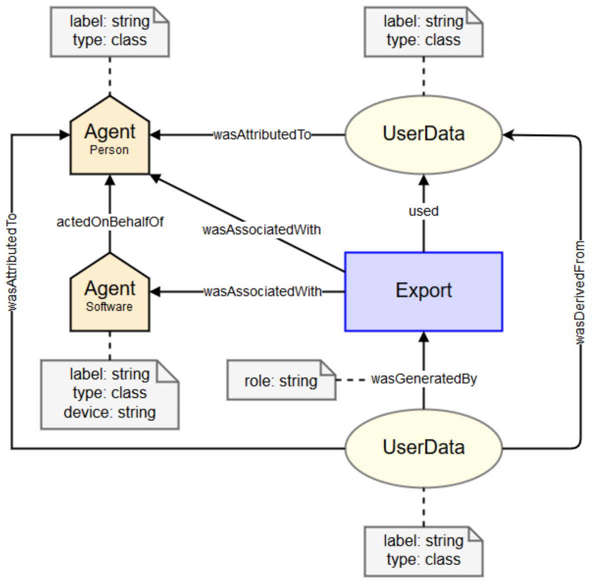
(c) New input sub-model with three agents

Figure B.1: Adjustments to the input sub-model

C Adjustments to the export sub-model



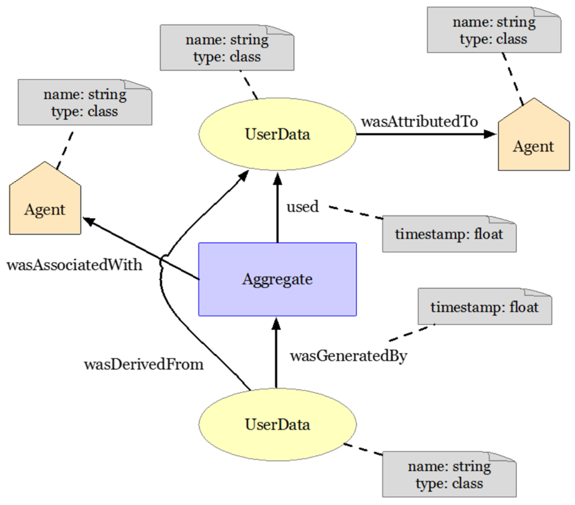
(a) Old export sub-model



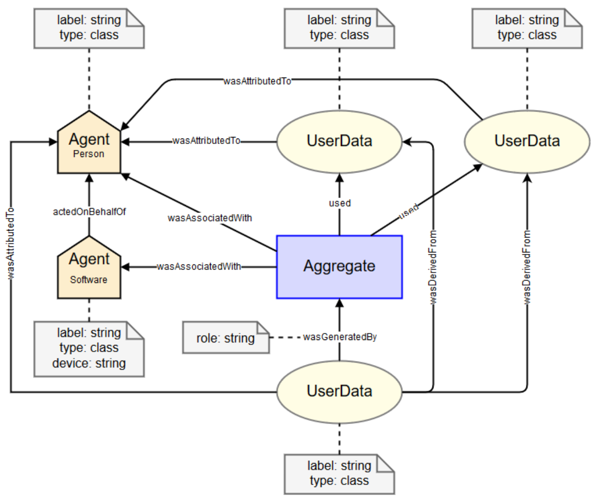
(b) New export sub-model

Figure C.1: Adjustments to the export sub-model

D Adjustments to the aggregate sub-model



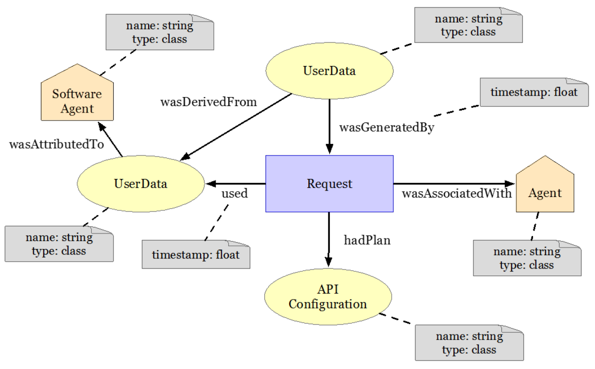
(a) Old aggregate sub-model



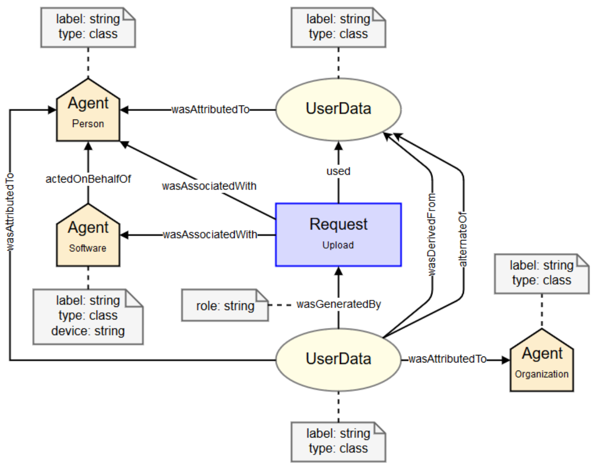
(b) New aggregate sub-model

Figure D.1: Adjustments to the aggregate sub-model

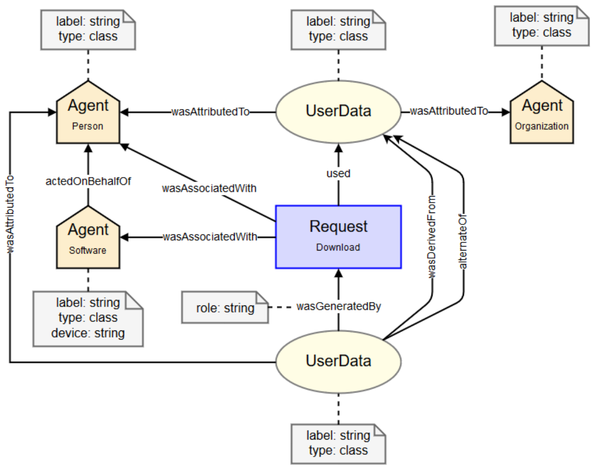
E Adjustments to the request sub-model



(a) Old request sub-model



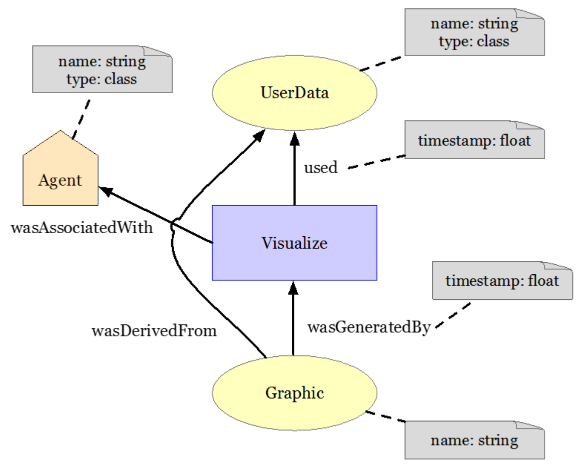
(b) New request model for uploading



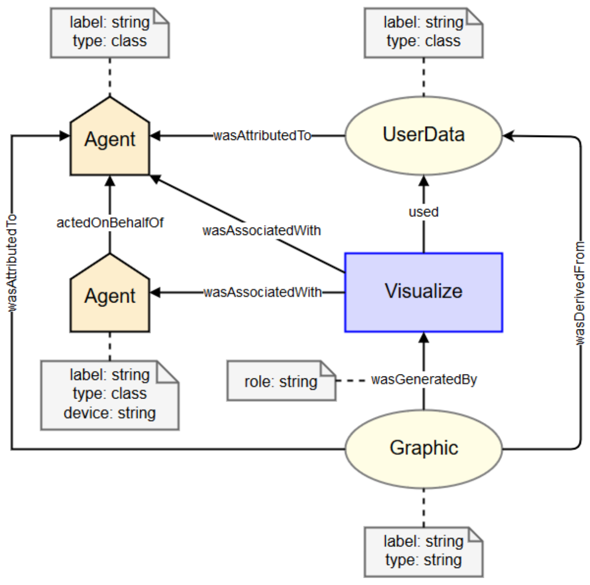
(c) New request model for downloading

Figure E.1: Adjustments to the request sub-model

F Adjustments to the visualize sub-model



(a) Old visualize sub-model



(b) New visualize sub-model

Figure F.1: Adjustments to the visualize sub-model

G Example provenance document: Input (manual)

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  wasGeneratedBy(userdata:heartratesNow, method:input, 2016-12-01T16:06:22+00:00, [prov:role="updating"])
  activity(method:input, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  entity(userdata:heartratesNow, [prov:type="heartrate", prov:label="Heart rates after input"])
  entity(userdata:heartratesThen, [prov:type="heartrate", prov:label="Heart rates before input"])
  agent(qs:app/heartmonitor, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="HeartMonitor"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  wasAttributedTo(userdata:heartratesNow, qs:user/regina@example.org)
  wasAttributedTo(userdata:heartratesThen, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/heartmonitor, qs:user/regina@example.org, -)
  used(method:input, userdata:heartratesThen, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:heartratesNow, userdata:heartratesThen, -, -, -)
  wasAssociatedWith(method:input, qs:app/heartmonitor, -)
  wasAssociatedWith(method:input, qs:user/regina@example.org, -, [prov:role="providing input"])
endDocument
```

H Example PROV-N document: Input (automatic)

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  wasAssociatedWith(method:input, qs:app/heartmonitor, -)
  wasAssociatedWith(method:input, qs:device/pulsetracker, -, [prov:role="providing input"])
  wasGeneratedBy(userdata:heartratesNow, method:input, 2016-12-01T16:06:22+00:00, [prov:role="updating"])
  activity(method:input, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  entity(userdata:heartratesNow, [prov:type="heartrate", prov:label="Heart rates after input"])
  entity(userdata:heartratesThen, [prov:type="heartrate", prov:label="Heart rates before input"])
  agent(qs:app/heartmonitor, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="HeartMonitor"])
  agent(qs:device/pulsetracker, [prov:type="qs:device", qs:device="bracelet", prov:label="Pulse tracking bracelet"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  wasAttributedTo(userdata:heartratesNow, qs:user/regina@example.org)
  wasAttributedTo(userdata:heartratesThen, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/heartmonitor, qs:user/regina@example.org, -)
  actedOnBehalfOf(qs:device/pulsetracker, qs:user/regina@example.org, -)
  used(method:input, userdata:heartratesThen, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:heartratesNow, userdata:heartratesThen, -, -, -)
endDocument
```

I Example PROV-N document: Export

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  wasGeneratedBy(userdata:steps.csv, method:export, 2016-12-01T16:06:22+00:00, [prov:role="exporting"])
  activity(method:export, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  alternateOf(userdata:steps.csv, userdata:steps)
  entity(userdata:steps, [prov:type="steps", prov:label="Steps database"])
  entity(userdata:steps.csv, [prov:type="csv", prov:label="Exported steps"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  agent(qs:app/stepcounter, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="StepCounter"])
  wasAttributedTo(userdata:steps.csv, qs:user/regina@example.org)
  wasAttributedTo(userdata:steps, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/stepcounter, qs:user/regina@example.org, -)
  used(method:export, userdata:steps, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:steps.csv, userdata:steps, -, -, -)
  wasAssociatedWith(method:export, qs:app/stepcounter, -)
endDocument
```

J Example PROV-N document: Aggregate

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  agent(qs:app/stepcounter, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="StepCounter"])
  wasAttributedTo(userdata:heartrate, qs:user/regina@example.org)
  wasAttributedTo(userdata:steps.heartrate, qs:user/regina@example.org)
  wasAttributedTo(userdata:steps, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/stepcounter, qs:user/regina@example.org, -)
  used(method:aggregate, userdata:heartrate, 2016-12-01T16:06:21+00:00)
  used(method:aggregate, userdata:steps, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:steps.heartrate, userdata:steps, -, -, -)
  wasDerivedFrom(userdata:steps.heartrate, userdata:heartrate, -, -, -)
  wasAssociatedWith(method:aggregate, qs:app/stepcounter, -)
  wasGeneratedBy(userdata:steps.heartrate, method:aggregate, 2016-12-01T16:06:22+00:00, [prov:role="exporting"])
  activity(method:aggregate, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  entity(userdata:steps, [prov:type="steps", prov:label="Steps database"])
  entity(userdata:heartrate, [prov:type="heartrate", prov:label="Heart rate database"])
  entity(userdata:steps.heartrate, [prov:type="qs:userdata", prov:label="Aggregation: steps and heart rate"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
endDocument
```

K Example PROV-N document: Request (upload)

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix service <http://software.dlr.de/qs/service/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com/>

  wasGeneratedBy(userdata:activities/steps, method:request, 2016-12-01T16:06:22+00:00, [prov:role="uploading"])
  activity(method:request, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  alternateOf(userdata:activities/steps, userdata:steps)
  entity(userdata:activities/steps, [prov:type="steps", prov:label="Steps database"])
  entity(userdata:steps, [prov:type="qs:userdata", prov:label="Steps database"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  agent(qs:app/stepcounter, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="StepCounter"])
  agent(qs:service/fitbit, [prov:type="prov:Organization", prov:label="Fitbit"])
  wasAttributedTo(userdata:steps, qs:user/regina@example.org)
  wasAttributedTo(userdata:activities/steps, qs:service/fitbit)
  wasAttributedTo(userdata:activities/steps, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/stepcounter, qs:user/regina@example.org, -)
  used(method:request, userdata:steps, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:activities/steps, userdata:steps, -, -, -)
  wasAssociatedWith(method:request, qs:app/stepcounter, -)
endDocument
```

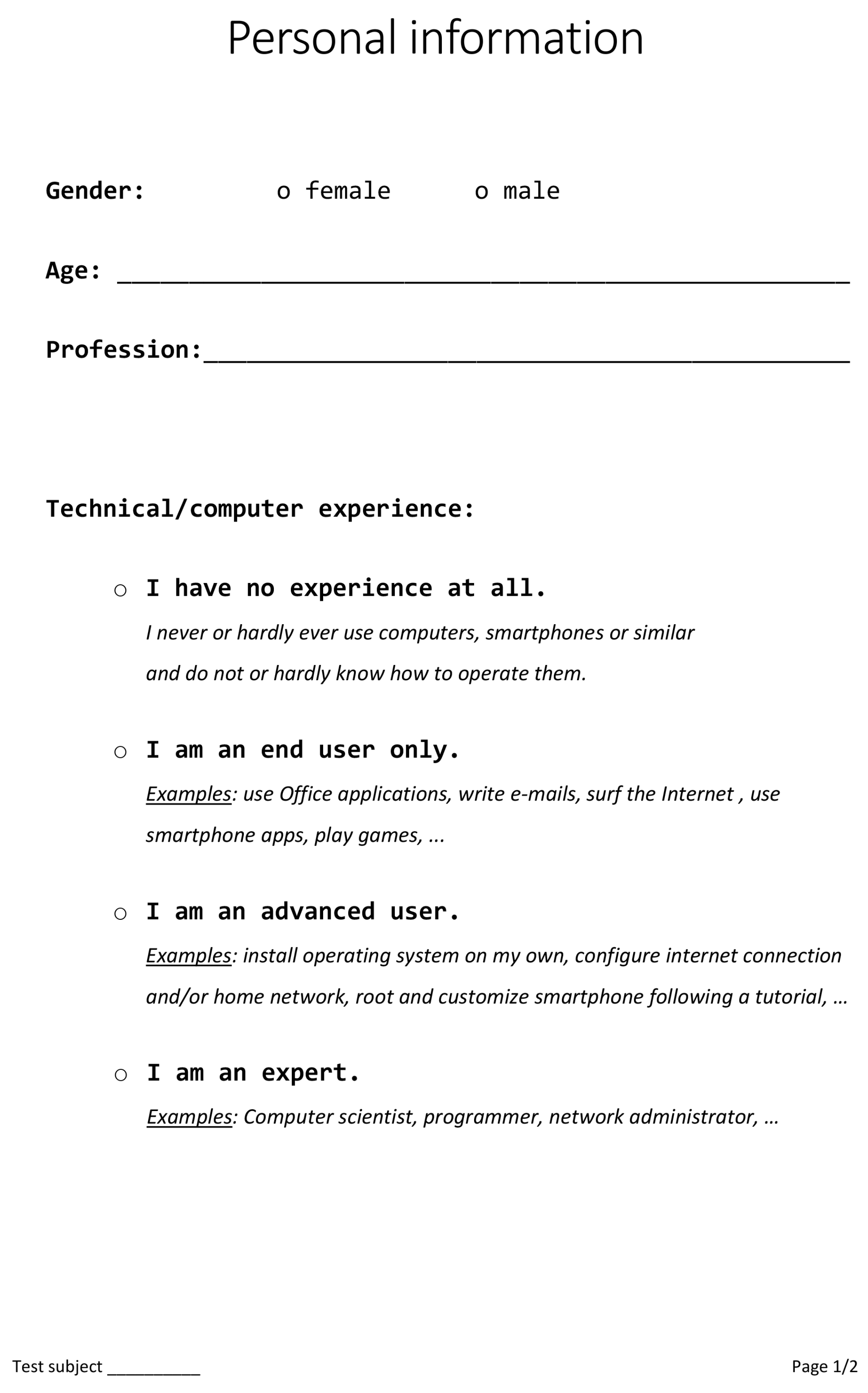
L Example PROV-N document: Request (download)

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix service <http://software.dlr.de/qs/service/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com/>

  actedOnBehalfOf(qs:app/stepcounter, qs:user/regina@example.org, -)
  used(method:request, userdata:activities/steps, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(userdata:steps, userdata:activities/steps, -, -, -)
  wasAssociatedWith(method:request, qs:app/stepcounter, -)
  wasGeneratedBy(userdata:steps, method:request, 2016-12-01T16:06:22+00:00, [prov:role="downloading"])
  activity(method:request, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  alternateOf(userdata:steps, userdata:activities/steps)
  entity(userdata:activities/steps, [prov:type="steps", prov:label="Steps database"])
  entity(userdata:steps, [prov:type="qs:userdata", prov:label="Steps database"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  agent(qs:app/stepcounter, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="StepCounter"])
  agent(qs:service/fitbit, [prov:type="prov:Organization", prov:label="Fitbit"])
  wasAttributedTo(userdata:activities/steps, qs:service/fitbit)
  wasAttributedTo(userdata:steps, qs:user/regina@example.org)
  wasAttributedTo(userdata:activities/steps, qs:user/regina@example.org)
endDocument
```

M Example PROV-N document: Visualize

```
document
  prefix userdata <http://software.dlr.de/qs/userdata/>
  prefix qs <http://software.dlr.de/qs/>
  prefix graphic <http://software.dlr.de/qs/graphic/>
  prefix app <http://software.dlr.de/qs/app/>
  prefix user <http://software.dlr.de/qs/user/>
  prefix device <http://software.dlr.de/qs/device/>
  prefix method <http://www.java.com>

  wasGeneratedBy(qs:graphic/diagram, method:visualize, 2016-12-01T16:06:22+00:00, [prov:role="displaying"])
  activity(method:visualize, 2016-12-01T16:06:21+00:00, 2016-12-01T16:06:22+00:00)
  entity(qs:graphic/diagram, [prov:type="linechart", prov:label="Line chart"])
  entity(userdata:steps, [prov:type="steps", prov:label="Steps database"])
  agent(qs:user/regina@example.org, [prov:type="prov:Person", prov:label="Regina Struminski"])
  agent(qs:app/stepcounter, [prov:type="prov:SoftwareAgent", qs:device="smartphone", prov:label="StepCounter"])
  wasAttributedTo(qs:graphic/diagram, qs:user/regina@example.org)
  wasAttributedTo(userdata:steps, qs:user/regina@example.org)
  actedOnBehalfOf(qs:app/stepcounter, qs:user/regina@example.org, -)
  used(method:visualize, userdata:steps, 2016-12-01T16:06:21+00:00)
  wasDerivedFrom(qs:graphic/diagram, userdata:steps, -, -, -)
  wasAssociatedWith(method:visualize, qs:app/stepcounter, -)
endDocument
```

N Reading study: Personal information



Personal information

Gender: o female o male

Age: ____

Profession:____

Technical/computer experience:

- ○ **I have no experience at all.**
  I never or hardly ever use computers, smartphones or similar
  and do not or hardly know how to operate them.

- ○ **I am an end user only.**
  Examples: use Office applications, write e-mails, surf the Internet , use
  smartphone apps, play games, ...

- ○ **I am an advanced user.**
  Examples: install operating system on my own, configure internet connection
  and/or home network, root and customize smartphone following a tutorial, …

- ○ **I am an expert.**
  Examples: Computer scientist, programmer, network administrator, …

Test subject __________ Page 1/2

I use (or used in the past) computer software, apps, websites, fitness trackers or similar to keep an eye on certain fitness values:

- ○ **Step counter**
  e.g. as a bracelet or other wearable device, or as a smartphone app

- ○ **Sleep tracker**
  e.g. as a bracelet or other wearable device, or as a smartphone app

- ○ **Pulse tracker**
  e.g. as a bracelet or using a smartphone app and the phone's camera

- ○ **Blood pressure**
  e.g. app, website, or Excel sheet, where blood pressure values can be entered

- ○ **Blood sugar**
  e.g. app, website, or Excel sheet, where blood sugar values can be entered

- ○ **Weight**
  e.g. app, website, or Excel sheet, where weight values can be entered
  or a radio-enabled weight scale that automatically transfers the weight

- ○ **Diet/Calories**
  e.g. app, website, or Excel sheet, where food intakes can be entered

- ○ **Sports**
  e.g. app, website, or Excel sheet, where type and duration
  of physical activities can be entered

- ○ **Others, namely:** ____

  ____

  ____

- ○ **Never used anything like that.**

Test subject __________ Page 2/2

O Reading study: Sheet 1

Sheet 1

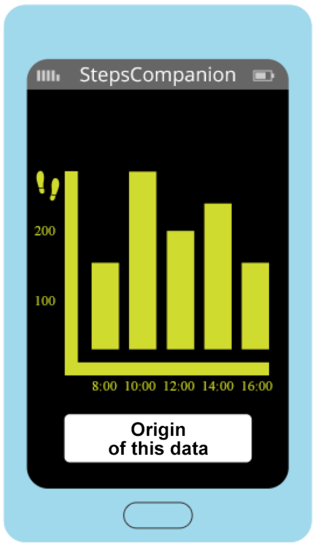
Please imagine the following scenario:

You wear a **fitness bracelet** every day,
counting the steps you take.

On your smartphone, there is an app called **„StepsCompanion“**,
which automatically syncs with your bracelet.

This way you can always view on your phone
how much you have already walked during the day.

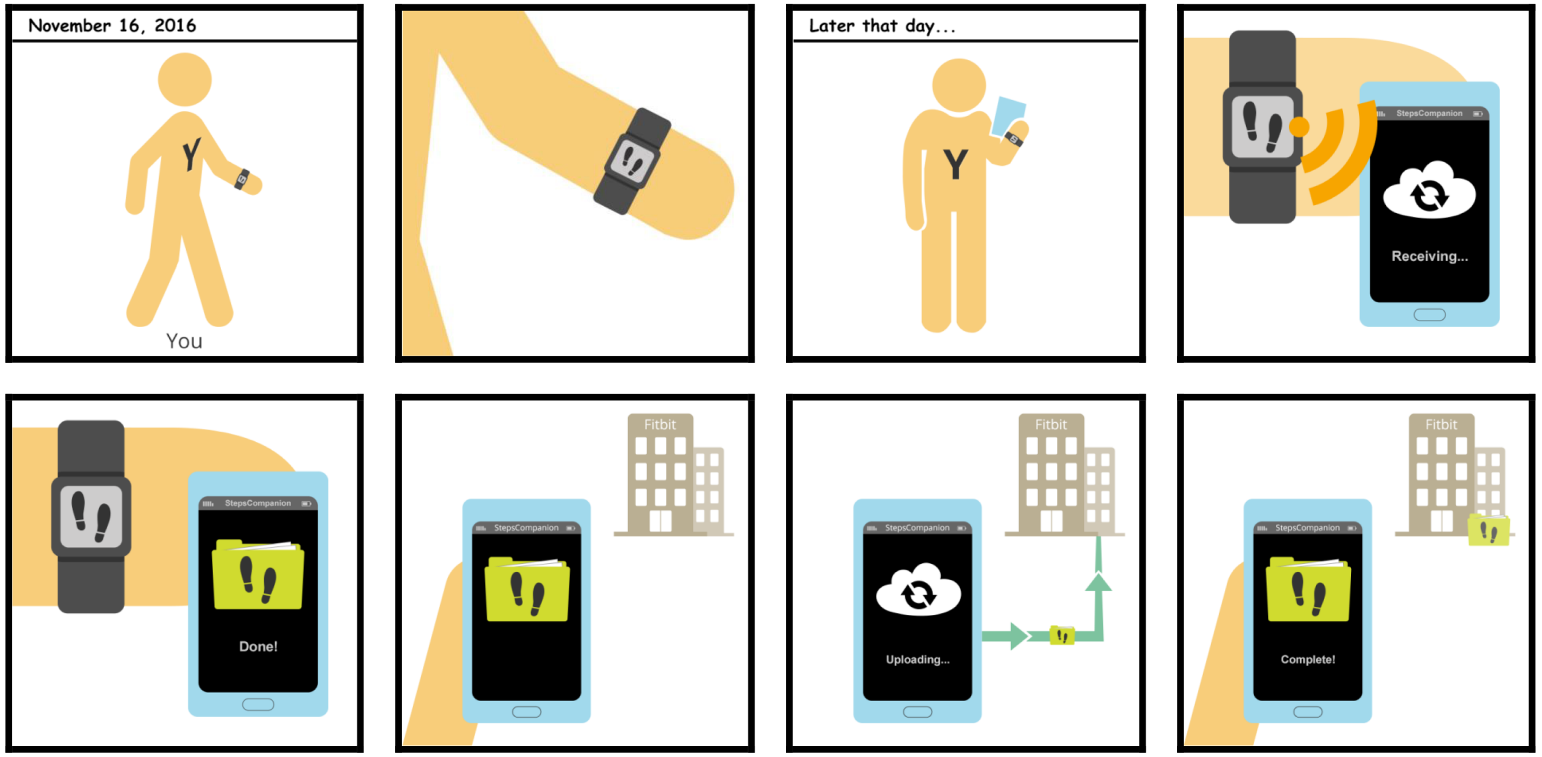
You are now viewing your steps from November 16, 2016 in the app:



The button *"Origin of this data"* makes you curious.

So you tap it and get to see the following:

Test subject __________ Page 1/2



Please take your time to inspect and interpret these pictures.

When you are ready, please tell the examiner what the pictures convey in your understanding.

Feel free to elaborate in great detail – mention anything that occurs to you or catches your attention.

Test subject __________ Page 2/2

P Reading study: Sheet 2

Sheet 2

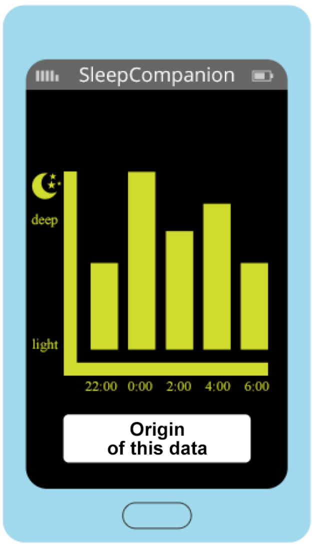
Please imagine the following scenario:

You wear a **fitness bracelet** every day,
tracking your sleep, i.e. your sleep cycles.

On your smartphone, there is an app called **„SleepCompanion“**,
which automatically syncs with your bracelet.

This way you can always view on your phone
how much you have slept,
and when your deep or light sleep phases were.

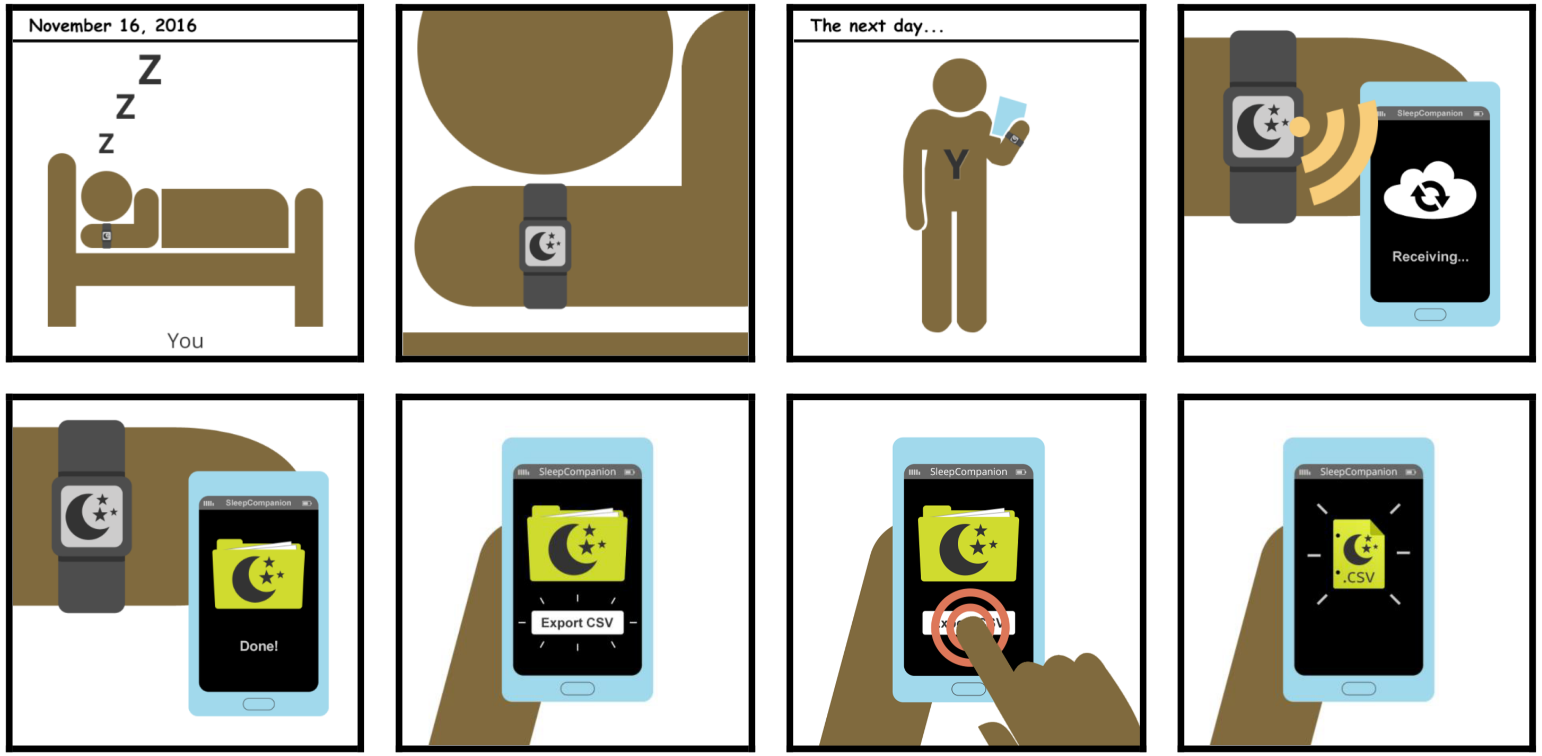
You are now viewing your sleep from November 16/17, 2016 in the app:



The button *"Origin of this data"* makes you curious.

So you tap it and get to see the following:

Test subject __________ Page 1/2



Please take your time to inspect and interpret these pictures.

When you are ready, please tell the examiner what the pictures convey in your understanding.

Feel free to elaborate in great detail – mention anything that occurs to you or catches your attention.

Test subject __________

Page 2/2

Q Reading study: Sheet 3

Sheet 3

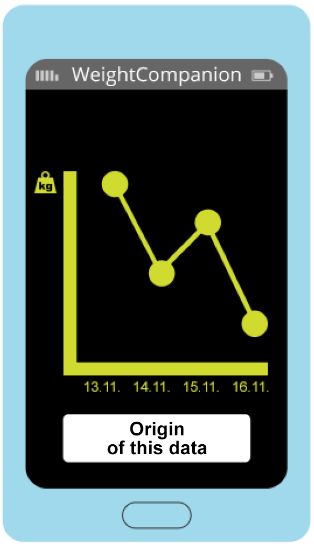
Please imagine the following scenario:

You own a **WiFi-enabled bodyweight scale**
that you use to **measure your weight every few days**.

On your smartphone, there is an app called **„WeightCompanion“**,
which can automatically receive data from your weight scale.

This way you can always view on your phone
what your weight was at different times.

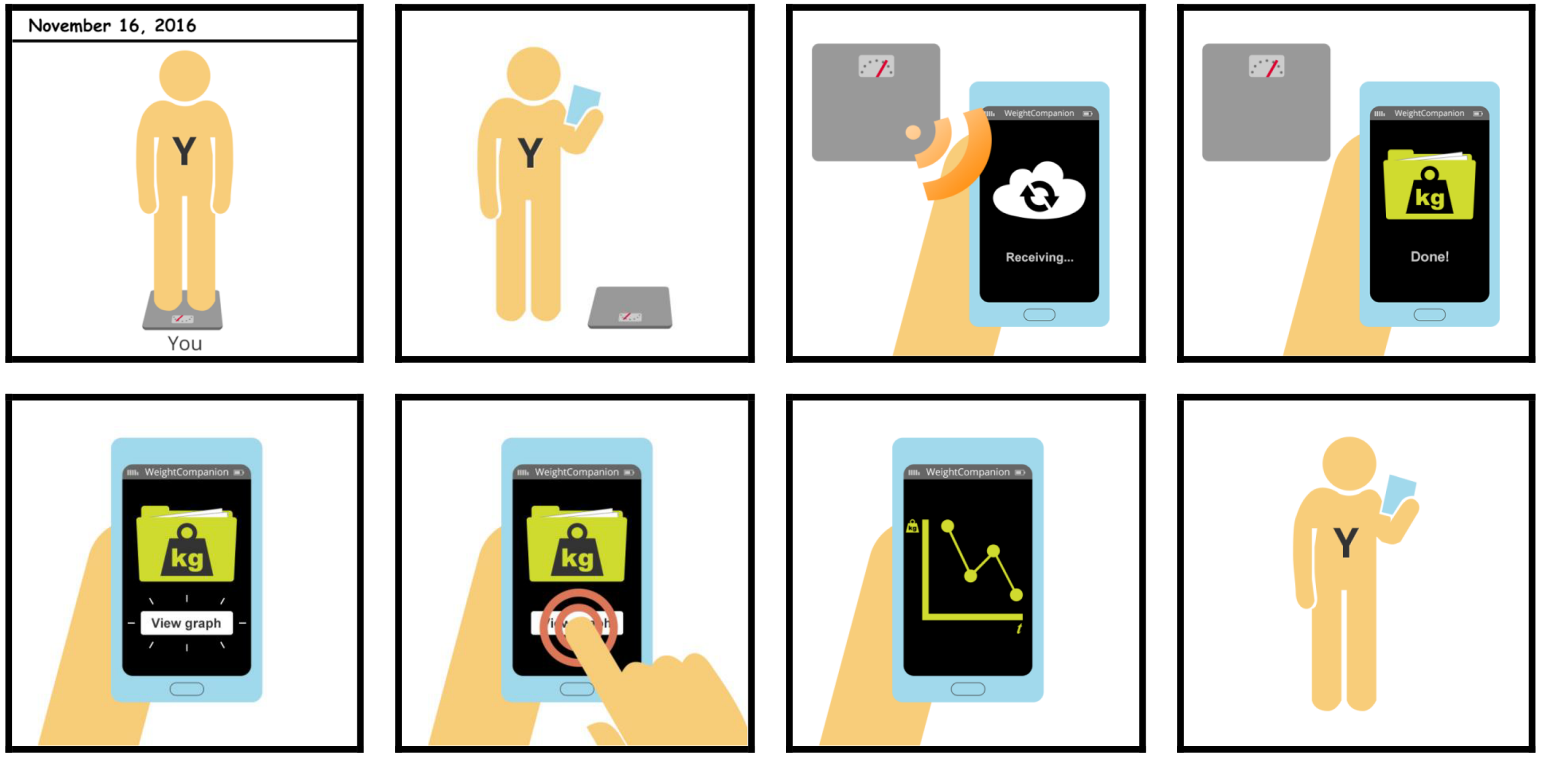
You are now viewing your weights up to November 16, 2016 in the app:



The button *"Origin of this data"* makes you curious.

So you tap it and get to see the following:

Test subject __________ Page 1/2



Please take your time to inspect and interpret these pictures.

When you are ready, please tell the examiner what the pictures convey in your understanding.

Feel free to elaborate in great detail – mention anything that occurs to you or catches your attention.

Test subject __________

Page 2/2

R Reading study: Sheet 4

Sheet 4

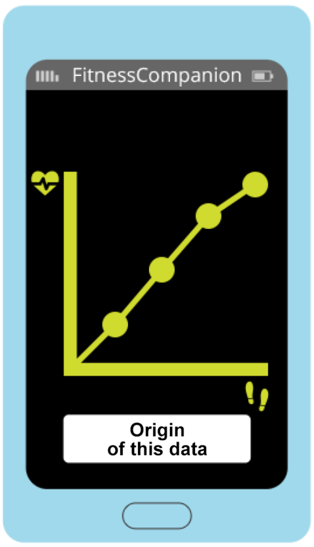
Please imagine the following scenario:

You have already logged your **steps** for a while using a fitness bracelet. This bracelet is also capable to **measure your pulse**.

On your smartphone, there is an app called **„FitnessCompanion“**, which is able to handle fitness data from various sources.

This way you can, for example, view on your phone **how much you walked at certain times, what your pulse was etc.**

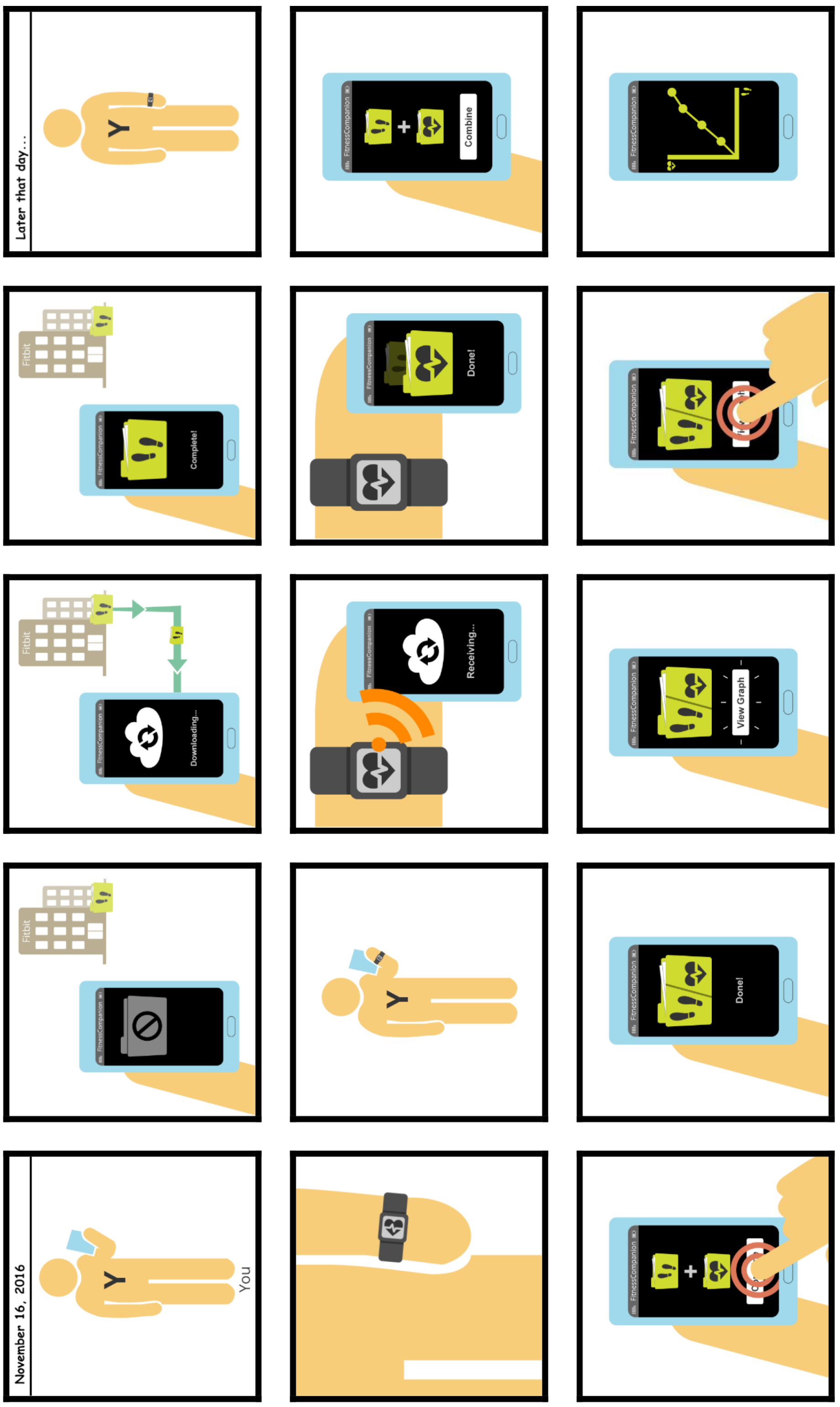
You are now viewing fitness data from November 16, 2016 in the app:



The button *"Origin of this data"* makes you curious.

So you tap it and get to see the following:

Test subject __________ Page 1/2



Page 2/2

Test subject ____________

S Reading study: Sheet 5

Sheet 5

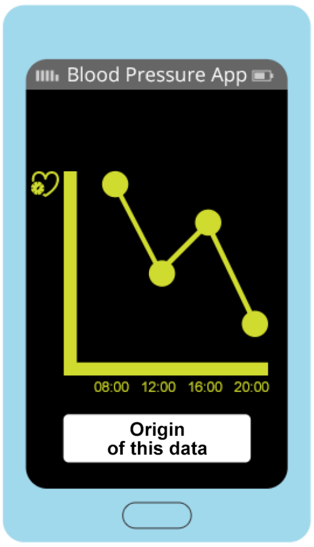
Please imagine the following scenario:

You measure your **blood pressure** regularly
using a conventional blood pressure meter.

Your computer has a **„Blood Pressure App“** installed,
which can be used to log the measurement results.

Moreover, there is also a **smartphone version** of this program.
You have this app installed on your phone.

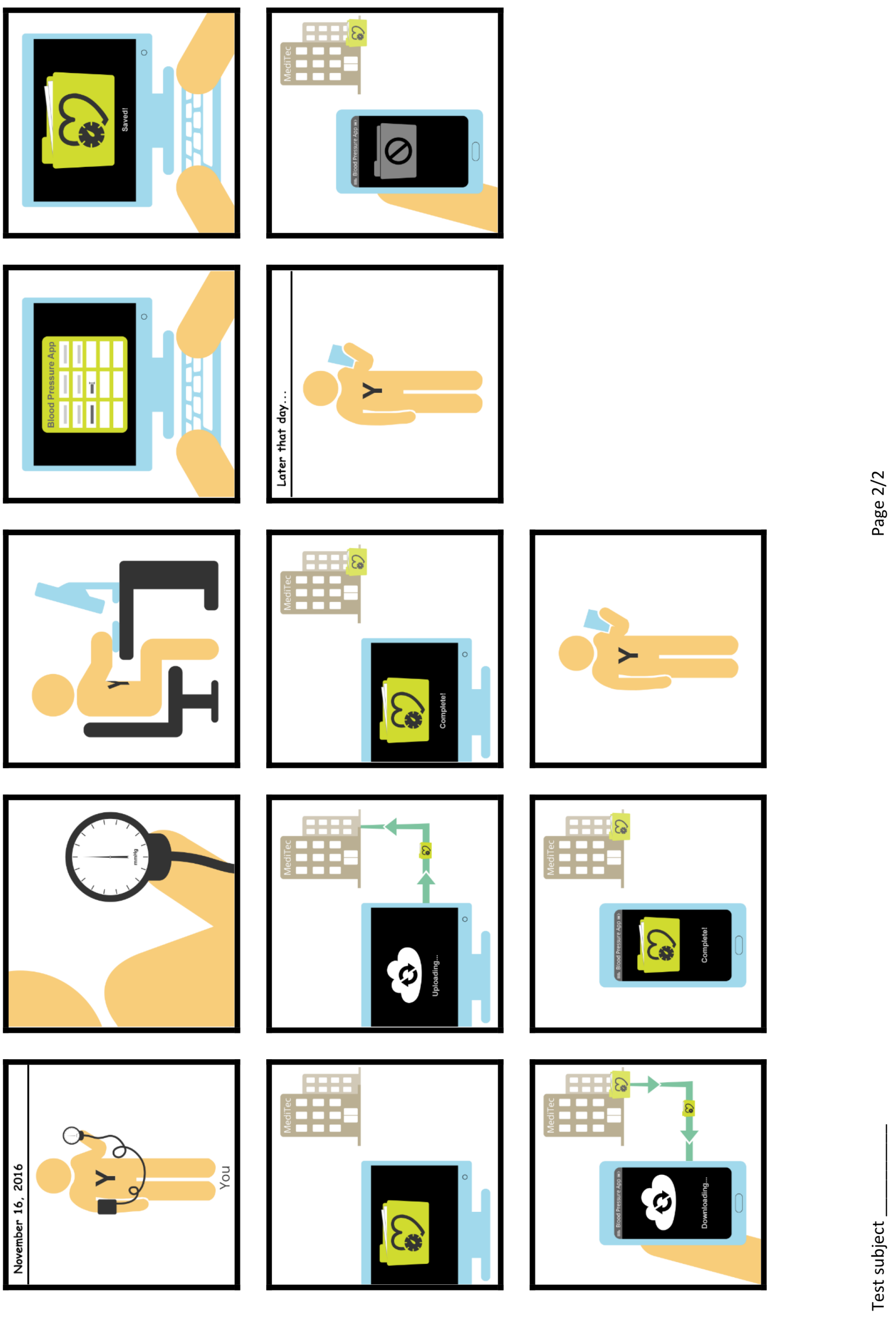
You are now viewing your blood pressure from
November 16, 2016 on the phone:



The button *"Origin of this data"* makes you curious.

So you tap it and get to see the following:

Test subject __________ Page 1/2


November 16, 2016
You
mmHg
Later that day…
Blood Pressure App
Saved!
MediTec
Uploading...
Complete!
Blood Pressure App
Downloading...
Page 2/2
Test subject ___________

T Reading study: Participants

Test subject	**Gender**	**Age**	**Technical expertise** (0 = none, 3 = expert)	**# QS applica-tions used**	**Profession**
on	f	28	2	4	Cook’s mate / waitress
er	f	63	1	4	Senior executive in aged care
mm	m	25	2	4	Student (computer science)
42	m	25	3	4	Student (computer science)
ab	m	26	3	4	Student (computer science)
nn	f	43	2	3	Primary school teacher
al	m	49	1	1	Commercial clerk
ud	f	40	2	1	Optometrist
te	m	49	2	0	Soldier
xe	m	29	2	1	Computer scientist / programmer
Average	n/a	37.7	2	2.6	n/a
Median	n/a	34.5	2	3.5	n/a

U List of findings for comic #1

Input and Request (upload)

		Test subjects (pseudonyms)									
Panel	**Finding**	on	er	mm	42	ab	nn	al	ud	te	xe
1	silhouette is "me", the user	x	x		x	x	x	x	x	x	x
1	user is walking	x	x		x	x	x		~	x	x
1	user is wearing a (fitness) bracelet / smartwatch	x	x	x	x	x	x	x	x	x	x
2	bracelet is a pedometer (step counter)	x	x	x	x	x	x	x	x	x	x
3	user takes/uses his smartphone	x	x	x	x	x	x	x	x	x	x
4	phone syncs with / gets data from bracelet	x	x	x	x	x	x	x	x	x	x
5	sync is complete, data is now on the phone	x		x	x	x	x	x	x	x	
6	there is a company or organization (called "Fitbit")	x	x	x	x	x	x		~	x	x
7	phone uploads the data to their server / "the cloud"	x	x	x	x	x	x	~	~	x	x
8	upload is complete	x			x	x	x			x	
8	data is now on the phone and the organization's server	x	x	x						x	x
8	…so it can be viewed from other devices, too	x									
	Percent found	**100**	**82**	**73**	**91**	**91**	**91**	**59**	**68**	**100**	**82**

Legend

Symbol	**Meaning**	**Score**
x	mentioned by the test person	1
~	mentioned partially	0.5
🟧	bonus finding, not included in result percentages	0

V List of findings for comic #2

Input and Export

Panel	Finding	on	er	mm	42	ab	nn	al	ud	te	xe
		Test subjects (pseudonyms)									
1	silhouette is "me", the user	x	x		x	x	x	x	x	x	x
1	user is sleeping	x	x	x	x		x	x	x	x	x
1	user is wearing a (fitness) bracelet / smartwatch	x	x		x	x	x	x	x	x	x
2	bracelet is a sleep tracker	~	x	x	x	x	x		x	x	x
3	the next morning; user wakes up / is awake		x		x		x	x	x	x	x
3	user takes/uses his smartphone	x	x	x	x			x	x	x	
4	phone syncs with / gets data from bracelet	x	x	x	x	x	x	x	x	x	x
5	sync is complete, data is now on the phone	x	x	x	x	x	x	x		x	
6	a possibility to export the data (to CSV) appears	x	~		x	x	x	~	x	x	x
7	user clicks/taps/uses the export function	x	x		x		x	x			x
8	data has been exported to a CSV file	x		x	x	x	x	~		x	x
8	the file is on the phone	x	x		x	x	x				x
8	the file is a copy/backup of the data		x	x						x	
8	…so it can be shared, used with other applications etc.	x		~					~	x	
	Percent found	81	88	54	92	62	85	69	62	85	77

Legend

Symbol	Meaning	Score
x	mentioned by the test person	1
~	mentioned partially	0.5
🟨	bonus finding, not included in result percentages	0

W List of findings for comic #3

Input and Visualize		Test subjects (pseudonyms)									
Panel	**Finding**	on	er	mm	42	ab	nn	al	ud	te	xe
1	silhouette is “me”, the user	x	x	x	x	x	x	x	x	x	x
1	user is measuring his weight	x	x		x	x	x	x	x	x	x
2	user takes his smartphone (directly after weighing)	x	x	x	x	x	x	x	x	x	
3	phone syncs with / gets data from weight scale	x	x	x	x	x	x	x	x	x	x
3	sync is done automatically, without user interaction	x									
4	sync is complete, data is now on the phone	x	x			x	x	x	x	x	x
5	a possibility to view the data as a graph appears	x	x	x	x	x	x	x	x	x	x
6	user clicks/taps/uses the view function	x	x			x	x	x		x	x
7	app displays a weight graph	x	x	x		x	x	x	x		x
7	the graph shows the weight history/progress	x	x		x	x	~	~	x	x	
	Percent found	100	100	56	67	100	94	94	89	89	78

Legend

Symbol	**Meaning**	**Score**
x	mentioned by the test person	1
~	mentioned partially	0.5
🟨	bonus finding, not included in result percentages	0

X List of findings for comic #4

Request (download), Input, Aggregate, and Visualize		Test subjects (pseudonyms)									
Panel	**Finding**	on	er	mm	42	ab	nn	al	ud	te	xe
1	silhouette is "me", the user	x	x	x	x	x	x	x	x	x	
1	user is using/looking at his smartphone	x	x	x		x	x	x			
2	data is missing from the phone/app	x		x		x	x			x	x
2	there is an organization (Fitbit)	x	x		x	x	x	~	x	x	x
2	Fitbit has the user's data	x	x		x	x	~				
3	phone downloads user's data from Fitbit	x	x	x	x	x	x	x		x	x
3	sync is done automatically, without user interaction		x								
4	sync is complete, data is now on the phone	x	x	x		x	x	x		x	x
5/6	it is some time later on the same day now	x	x		x		x	~		x	
5/6	user is now wearing a pulse tracking bracelet	x	~	x	x	x	x		x	x	x
7	user takes/uses his smartphone		x	x			x	x	x		
8	phone syncs with / gets data from bracelet	x	x	x	x	x	x	x	x	x	x
9	sync is complete, data is now on the phone	x	x	x		x	x	x		x	
10	both steps and pulse data are now on the phone	x		x			x		x		x
10	there is a possibility to "combine" (aggregate) them	x	x	x	x	x	x	x	~	x	x
11	user clicks/taps/uses the "combine" function	x	x				x	x		x	
12	"combination" (aggregation) is finished	x	x		x			x		x	
12	there is now a file/folder with steps *and* pulse data				x	x	x	x			~
13	a possibility to view the data as a graph appears	x	x	x	x	x	x	x	x	x	x
14	user clicks/taps/uses the view function	x	x				x	x		x	x
15	app displays a graph	x	x	x	x		x	x	x	x	
15	the graph shows the steps and pulse correlations	x	x	x	x	x	x		x	x	x
	Percent found	90	83	67	62	67	93	71	45	76	55

Legend

Symbol	**Meaning**	**Score**
x	mentioned by the test person	1
~	mentioned partially	0.5
🟧	bonus finding, not included in result percentages	0

Y List of findings for comic #5

Input, Request (upload), and Request (download)		**Test subjects (pseudonyms)**									
Panel	**Finding**	on	er	mm	42	ab	nn	al	ud	te	xe
1/2	silhouette is "me", the user	x	x	x	x	x	x	x	x	x	
1/2	user is measuring his blood pressure	x	x	x	x	x	x	x	x	x	x
3	user is at the desktop computer	x	x	x	x	x	x	x	x	x	x
4	there is a blood pressure app/website/tool on the PC	x	x		x	x			x	x	x
4	user enters the values he just measured	x	x	x	x	x	x		x	x	x
5	data is now saved on the PC	x	x	x	x	x	x		x	x	x
6	there is a company or organization (called "MediTec")	x	x	x	x	x	x		x	x	x
7	computer uploads the data to their server / "the cloud"	x	x	x	x	x	x		x	x	x
8	upload is complete	x	x			x		x			
8	data is now on the PC and the organization's server			x					x		
9	it is some time later on the same day now	x	x	x			x	x		x	
9	user takes/uses his smartphone	x	x		x	x	x	x	x		x
10	data is missing from the phone	x				x	x			x	
10	Meditec has the user's data						x		~		
11	phone downloads user's data from Meditec	x	x	x	x	x	x		x	x	x
11	sync is done automatically, without user interaction						x				
12	sync is complete, data is now on the phone	x	x	x	x	x	x			x	x
12	… and on the PC	x		x	x		x		x		~
	Percent found	**88**	**76**	**71**	**71**	**76**	**82**	**35**	**74**	**71**	**62**

Legend

Symbol	**Meaning**	**Score**
x	mentioned by the test person	1
~	mentioned partially	0.5
🟧	bonus finding, not included in result percentages	0

Z Evaluation table

	on	er	mm	42	ab	nn	al	ud	te	xe	ø	Med.	ø f	ø m	Med. f	Med. m	ø age<37.7	ø age>37.7	Med. age<37.7	Med. age>37.7
Gender	f	f	m	m	m	f	m	f	m	m										
Age	28	63	25	25	26	43	49	40	49	29	37,7	34,5	43,5	33,8	41,5	27,5	26,6	48,8	26,0	48,8
Tech. Experience	2	1	2	3	3	2	1	2	2	2	2,0	2,0	1,8	2,2	2,0	2,0	2,4	1,6	2,0	1,6
QS Experience	4	4	4	4	4	3	1	1	0	1	2,6	3,5	3,0	2,3	3,5	2,5	3,4	1,8	4,0	1,8
Σ Experience	6	5	6	7	7	5	2	3	2	3	4,6	5,0	4,8	4,5	5,0	4,5	5,8	3,4	6,0	3,4
Comic #1																				
findings	11	9	8	10	10	10	6,5	7,5	11	9										
out of	11																			
Percent	**100**	**82**	**73**	**91**	**91**	**91**	**59**	**68**	**100**	**82**	**84**	**86**	**85**	**83**	**86**	**86**	**87**	**80**	**91**	**80**
Comic #2																				
findings	10,5	11,5	7	12	8	11	9	8	11	10										
out of	13																			
Percent	**81**	**88**	**54**	**92**	**62**	**85**	**69**	**62**	**85**	**77**	**75**	**79**	**79**	**73**	**83**	**73**	**73**	**78**	**77**	**78**
Comic #3																				
findings	9	9	5	6	9	8,5	8,5	8	8	7										
out of	9																			
Percent	**100**	**100**	**56**	**67**	**100**	**94**	**94**	**89**	**89**	**78**	**87**	**92**	**96**	**81**	**97**	**83**	**80**	**93**	**78**	**93**
Comic #4																				
findings	19	17,5	14	13	14	19,5	15	9,5	16	11,5										
out of	21																			
Percent	**90**	**83**	**67**	**62**	**67**	**93**	**71**	**45**	**76**	**55**	**71**	**69**	**78**	**66**	**87**	**67**	**68**	**74**	**67**	**74**
Comic #5																				
findings	15	13	12	12	13	14	6	12,5	12	10,5										
out of	17																			
Percent	**88**	**76**	**71**	**71**	**76**	**82**	**35**	**74**	**71**	**62**	**71**	**72**	**80**	**64**	**79**	**71**	**74**	**68**	**71**	**68**
Overall %	**92**	**86**	**64**	**76**	**79**	**89**	**66**	**67**	**84**	**71**	**77**		**84**	**73**			**76**	**78**		