Universität des Saarlandes
Fakultät für Mathematik und Informatik

Fachrichtung Informatik

Masterarbeit

Domain-Independent and Customizable User Interface Builder for RDF Data

vorgelegt von
Mohsen Mansouryar
am 16.8.2022

Begutachtet von:
Supervisor: Prof. Dr.-Ing. Philipp Slusallek
Second Examiner: Dr. Hilko Hoffmann
Co-Supervisor: Andre Antakli

Declarations

Declaration of Authorship

I, Mohsen Mansouryar, hereby confirm that I have written this thesis, titled 'Domain-Independent and Customizable User Interface Builder for RDF Data', on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Declaration of Consent

I, Mohsen Mansouryar, agree to make both my thesis document and the developed software's source code, titled 'Domain-Independent and Customizable User Interface Builder for RDF Data' and the accompanying software Graph-ical v0.9, (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

16.8.2022

Mohsen Mansouryar

Abstract

This thesis is aimed to provide a generic solution to visualizing data which is modelled as RDF triples. Our main goal is not just visualization but doing so in a user-friendly, domain-independent, and interactive manner. Moreover, we try to infer context via the information available in RDF vocabularies used in representing given data and custom domain-specific adaptations applied by an expert when designing a user interface.

As real-world objects can have multiple representations, we provide different sets of visualizations that capture partial views of an object. To this end we provide an entity-centric approach with different visualization contexts such as lists, tables, networks, charts, etc.

To prove the effectiveness of this framework we have applied it to two completely separate domains with different visualization needs covering both interactivity and exploration as well as flexibility in terms of UI setup. We discuss how the effectiveness of our tool in addressing key requirements in each domain and how the rest of requirements can be addressed in the same framework in future work.

Contents

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Goals . . . 4
  - 1.2.1 Research Questions . . . 4
  - 1.2.2 Approach . . . 5

2 Related Work **6**
- 2.1 Resource Description Framework . . . 6
  - 2.1.1 From RDF to RDFS and OWL . . . 6
  - 2.1.2 SPARQL . . . 7
  - 2.1.3 RDF Ontologies and Vocabularies . . . 8
  - 2.1.4 Visualizing RDF Data . . . 9
- 2.2 Graph-based Visualizations . . . 9
- 2.3 Cognitive Modelling . . . 9

3 Applications **10**
- 3.1 Pxio . . . 10
  - 3.1.1 Overview . . . 10
  - 3.1.2 Goals . . . 12
  - 3.1.3 Personas . . . 12
  - 3.1.4 Usecases . . . 13
- 3.2 Summary . . . 15

4 Methodology **15**
- 4.1 Metrics . . . 16
- 4.2 Keystroke-Level Model . . . 16
- 4.3 Requirement Gathering . . . 16
- 4.4 Prototypes . . . 16
- 4.5 Implementation . . . 16
- 4.6 Technical Evaluation . . . 18

5 Graph-ical Workflow **18**

6 Results **26**

7 Conclusion and Future Work **36**
- 7.1 Context-Awareness . . . 36

8 Appendix A **37**
- 8.1 Pxio Requirement Interview . . . 37
  - 8.1.1 Goal . . . 37
  - 8.1.2 Participants . . . 37
  - 8.1.3 Initial Discussion (30 minutes) . . . 37
  - 8.1.4 User Context . . . 37
  - 8.1.5 Pxio . . . 38

8.1.6 User Interface 38
8.1.7 Concepts 40
8.1.8 Prototype Reveal (3-5 minutes) 40
8.1.9 Tasks (roughly 30 minutes (3 each task)) 40
8.1.10 Final Discussion (5 to 10 minutes) 41

1 Introduction

1.1 Motivation

The user interface (UI) is an essential component of every software system. With the rapid growth in the volume of data and development of software solutions to both model and analyze data, providing a user-friendly interface to explore data has become increasingly important. The need for representing and modelling complicated data has led to development of data description standards such as resource description framework (RDF), which aim to facilitate description of complex relationships between different entities. Although such standards make it easier to describe data, the verbose syntax makes the data less human-readable. It is more desirable for human visual system to comprehend visualizations rather than text data. Software systems that enable users to explore data, usually provide tools such as graphs and tables to visualize data. While it is important to add more features to software systems to analyze and visualize data, it is also crucial to pay attention to the complexity of the UI. A user-friendly interface needs to give users as much control over data as possible, but it should maintain a high level of comfort and should not increase cognitive load. The other problem with existing visualization tools is that often such tools are either too focused on the application and the domain of data, or too simple that the UI lacks the necessary interactivity and often do not provide the user enough flexibility to explore the data.

While it may seem trivial to provide more control to user to explore data by supplying software systems with additional UI tools, a complicated UI usually cause more distractions which can lead to a poor user experience. It is essential to build a user-friendly interface that gives users as much control as possible over the ways they can explore data, accompanied by ease of use and reduction of cognitive load. The other challenge is to make visualization tools domain-independent without losing interactivity and flexibility to explore the data. Such problems highlight the importance of measuring the quality of UI to prevent a UI to become increasingly difficult to work with or to lose the essential interactivity. There is a variety of tools that can measure the quality of UI either by measuring direct metrics on the UI or by surveying the user experience. The latter requires surveying a sufficiently large and diverse population to prevent biased statistical results. For the scope of this thesis, we focus on metrics that we can measure directly by studying the UI.

Designing domain-independent visualization tools requires going beyond one or a few applications and working directly with abstract data semantics. Focusing too much on the domain-dependent properties of data prevents a data visualization tool to flexibly work with different applications. On the other hand, a domain-independent visualization tool needs to facilitate sufficient interactivity that enables users to easily explore and understand the data. The main challenge is to find the right balance between the two properties of domain-independence and interactivity, yet achieving a high score with regards to UI quality metrics.

Apart from the size of information in Semantic Web, real-world objects can have multiple representations. These representations can only capture a view of the object in a specific context. These issues of representation and context in data create diversity. This diversity creates challenges while designing a generic User Interface.

The main contribution of this thesis is in the development of a domain-independent visualization framework for a variety of users. Starting from the domain analysis, requirement gathering phase to implementation of a User Interface design tool, multiple stakeholders were included in the design and development cycle. The next section presents the motivation for the research in more detail and sets the stage for the thesis.

1.2 Goals

Domain independent visualization is a challenging task mainly due to the variety of requirements it needs to cover to be intuitive and useful to a different set of user personas. When underlying data consists of multiple types and properties visual cohesion and complexity also become challenging to address. In this work we aim to make use of the modelling power of RDF data to alleviate some of the uncertainties in visualizing data coming from its Semantics.

Flexibility of RDF modelling also allows to design a visualization in the same format as the data being visualized. This requires understanding the basics of Semantic Web evolution from RDF to RDFS and OWL. In the spirit of "AAA" principle from the Semantic Web community which states Anyone can say Anything about Any topic, we propose an extension where Anyone can further visualize Any data from Any domain.

Our goal is essentially to build a framework upon which domain experts that can either model their data as RDF or already have it modelled as RDF triples can build user interfaces that can be useful and intuitive for a wide range of end-users. This requires being able to create task-specific UIs that focus on singular user stories defined during a requirement gathering phase and a UI design process.

1.2.1 Research Questions

In order to achieve our goals, The following research questions (RQ) serve as the focal point of our work, and they represent the important components of our work as a whole.

- RQ1: "What kinds of visualization strategies do users desire in a domain independent tool built for visualizing and interacting with data modelled in RDF format?"
  Given semantics of the data and the requirements of end-users, we need to investigate which features should be present in a tool that provides a convenient workflow for building domain-independent UIs.

- RQ2: "Is using a node graph for displaying RDF information intuitive and friendly enough for interactive interfaces?"
  Due to nature of RDF triples being easily representable using a node graph, we want to investigate whether extending such visualization and adding customizations can make up for the lack of other ways of visualizing the same data and still be intuitive enough for end-users.

- RQ3: "How can end-users and Semantic Web experts use the same tool to both design a UI and use it for their domain of choice?"
  Since our tool and the underlying data are extensively based on RDF and Semantic Web

knowledge, we want to further investigate how ca can decouple this knowledge from the UI workflow such that the same tool can be used by both experts and novice users.

- RQ4: "Is the resulting software practical and an improvement in terms of user experience and metrics compared to existing solutions?"
  Since our goal is to develop a proof of concept tool that can match the requirements of two different domains Naturally, a key question is whether the resulting software is of practical use.

1.2.2 Approach

We partially follow the guidelines of user-centered design to achieve the aforementioned aims and to discover solutions to our research goals. Furtheremore using a cognitive modelling approach we use a measurable metric of execution time for evaluation in absence of usability testing. The following diagram summarizes our approach

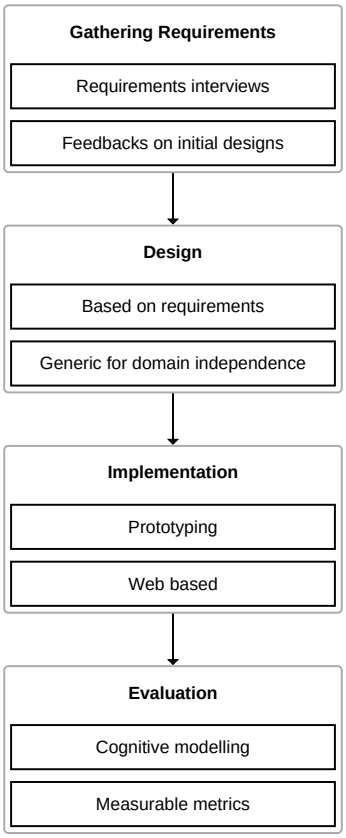


Figure 1: Overview of our Approach

- Gathering Requirements
  We initially start by gathering requirement from domain experts regarding their specific needs and the existing tools they use. This helps up also get a feedback over initial design concept while making sure the end result meets user requirements regardless of their domains.
- Design
  We design our software based on the gathered set of requirements and certain domain-independent requirements depending on each visualization type as we require to keep a certain our tool generic.
- Implementation
  A proof of concept implementation is carried out based on the results of the previous stages. Here we limit our focus to a web-based interface that can run smoothly on a desktop computer.
- Evaluation
  We evaluate our framework based on selective key use-cases for each user persona identified during the requirement gathering process. here we use a measurable metric to compare the estimated execution time of each usecase comparing to the existing tools in each domain and discuss the results.

2 Related Work

2.1 Resource Description Framework

The Resource Description Framework (RDF) is the official W3C Recommendation for describing data about resources on the web and serves as the data model for Linked Data. Because RDF is graph-based, it offers the desired interconnectedness that the Semantic Web demands. Its main goal is to explain the information for web resources, including things like author, title, copyright, and so forth. RDF is designed to be processed quickly by computers, not for human readers to easily understand. It makes it possible for entities to exchange information without any meaning being lost. RDF's structure is determined by the data itself, as opposed to relational databases like SQL, where it is determined by things like tables [7, 9]. Incorporating technologies like RDF and SPARQL, both of which will be covered in the upcoming sections, Linked Data is the Semantic Web's main building block. Large-scale integration and inference on Semantic Web data are made possible by Linked Data. But even more crucially, it is about how the many data bits relate to one another. It is not just about the data itself. [2]

2.1.1 From RDF to RDFS and OWL

In Semantic Web, Anyone can say Anything about Any topic. this is the so called AAA principle [13]. statements are described in languages with different levels of expressivity, with RDF being the most basic layer (least expressive). Resource Description Framework (RDF) is thus the basic framework for allowing anyone to make a basic statement about anything via (Subject, Predicate, Object) triples. RDF Schema (RDFS) adds more expressivity so we are able to describe basic notions of commonality and variability (classes, sub classes, properties). Furthermore Web Ontology Language (OWL) brings expressivity of Logic, allowing models to express detailed constraints

between classes, instances, and properties.

2.1.2 SPARQL

A method of retrieving and altering the data is also needed because simply storing information as RDF data is not particularly useful. As its name suggests, the query language used with RDF is called SPARQL Protocol and RDF Query Language (SPARQL). We take into account SPARQL 1.1, which is the most recent revision. Similar to RDF, SPARQL queries are constructed using triple patterns, which are then compared to the RDF triples contained in a particular repository. The primary distinction is that triple terms in a SPARQL query might be variables rather than resources. Additionally, SPARQL has a variety of filters, functions, and operators, as is customary for query languages. Following, we'll provide a brief summary of key SPARQL characteristics. In the SPARQL specification, a complete list of constructs, functions, and other items is provided [8].

Although the WHERE clause in SPARQL can occasionally be removed, graph patterns are normally matched inside of it. The matching RDF graph is then altered to produce the final solution, depending on the query type. There are four different query forms available. Any number of particular variables or all variables bound in the pattern match are returned by SELECT. CONSTRUCT creates a set union based on the discovered matches, building an RDF graph using the template specified in the WHERE clause. The answer excludes triples with unbound variables. If and only if a pattern match has been discovered, ASK returns a boolean that is true. DESCRIBE provides an RDF graph containing all the details about each resource in the pattern match that were discovered. The SPARQL processor chooses the precise details of the description that is contained. It should be noted that, like RDF, namespaces can be condensed using PREFIX declarations [8].

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX pxio: <http://www.pxio.de/rdf#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?s ?p ?o WHERE { {
    ?s ?p ?o
    FILTER EXISTS{
      VALUES ?type1 { pxio:User pxio:UserGroup }
      ?s rdf:type ?type1 .
    }
    FILTER(
      isLiteral(?o) ||
      ?p IN (rdf:type)
    )
  } UNION {
    ?s ?p ?o
    FILTER EXISTS{
      VALUES ?type1 { pxio:User pxio:UserGroup }
      VALUES ?type2 { pxio:User pxio:UserGroup }
      ?s rdf:type ?type1 .
      ?o rdf:type ?type2 .
    }
    FILTER(?p = foaf:member)
  } UNION {
    ?s ?p ?o
    FILTER (?s IN (pxio:User, pxio:UserGroup))
  }
}
```

Query 1: Example SPARQL query

An example SPARQL query is shown in Query 1, which consists of PREFIX definitions on the top and a SELECT statement which utilizes three UNIONs to apply FILTERs on RDF triples to query RDF data.

2.1.3 RDF Ontologies and Vocabularies

Ontologies and vocabularies are required to make sense of all the data on the Semantic Web and to allow reasoning based on that data. They explain the connections between various data swathes. The distinction between "ontology" and "vocabulary" for the Semantic Web is not well defined, despite the fact that ontologies often describe more complicated, formal collections while vocabulary is utilized more broadly. We shall interchangeably use both terms throughout this study [11].

A variety of terms are available thanks to vocabularies. Ontologies can reduce ambiguity by choosing synonyms or by sticking to a particular lexicon. These definitions provide a trustworthy description of inference rules. A variety of methods, including OWL [4] and RDF [7] and RDF Schema (RDFS) [6], can be used to define vocabularies. These have unique languages for defining relationships, classes, and other concepts. Because a vocabulary's complexity can vary greatly, one

description style may be better suited for a certain context than another [11].

2.1.4 Visualizing RDF Data

As the evolution of modelling languages in RDF goes in the direction of enriching the level of expressivity of data, we can make use of this to identify context of data during visualization. For instance by knowing an instance to be a of a basic XML data type, we can render it accordingly in UI, reflecting the properties of its type (e.g. checkbox for Boolean data, input field for a string, ...). Or by knowing the domain and range information of a specific predicate, we can render the corresponding classes and their instances when working with such relation anywhere in the UI. There are a number of research studies dedicated to visualizing RDF data, but we can categorize them into non-graph based and graph-based visualizations [25].

Non-graph based methods of visualizing RDF, such as Haystack [21] and mSpace [24], usually focus on presenting data in a logical sequence that comes naturally from the data and utilize facets, categories, and subject description. Such methods have a better performance than primitive graph-based visualization tools, such as RDF-gravity [5] and IsaViz [1], as the graphs become larger and less easily explorable with the increase in the size of RDF data [23]. Graph-based visualization tools have been developed independently that can partially address the size problem. However, non-graph based RDF visualization methods may still be used in domain specific applications to represent the data in a more meaningful way than graph-based visualization tools.

2.2 Graph-based Visualizations

Graph-based visualization tools often use different shapes, colors, and schemes to make the data more visually appealing, but by paying the price of losing details and having more redundant visuals repeated on every node and edge. Tools such as Visual RDF [10], WebVOWL [12], and LodLive [3], have the advantage of being able to easily visualize RDF data using graphs, but their visualization gets complicated when the size of the graph gets larger, the graphs becomes less readable because of overlapping edges and redundant information, and they fail to represent details of each node, hence not suitable for domain-independent large-scale data visualization.

2.3 Cognitive Modelling

Rather than observing and measuring actual users completing tasks, another approach for estimating productivity is cognitive modeling. Cognitive modeling is an analytic technique (as opposed to the empirical technique of usability testing). It estimates the task completion time from generalized estimates of the low-level motor operations. Breaking up the task that a user performs into millisecond level operations permits the estimation of task completion times for experienced users completing error-free trials.

The most familiar of these cognitive modeling techniques is GOMS (Goals, Operators, Methods and Selection Rules), first described in the 1970s in research conducted at Xerox Parc and Carnegie-Mellon and documented in the still highly referenced text The Psychology of Human Computer Interaction, by Card, Moran and Newell (1983) [16]. GOMS itself represents a family of techniques, the most familiar of which is Keystroke Level Modeling (KLM).

A usability analyst can use KLM to estimate user behaviors in its most basic version by utilizing just a few operators (pointing, clicking, typing, and thinking); for a brief introduction, see [22] p.72. KLM has been used the most by practitioners, presumably because of its simplicity. It has been

demonstrated that this method can accurately predict job completion durations within 10% to 30% of real timeframes. These projections may be based on prototypes or actual functional goods. Maps, PDAs, and database applications are just a few of the numerous applications and domains on which it has been tested [20, 18, 19].

The time-consuming nature of millisecond-level time estimation is one of KLM's biggest drawbacks. Even simple operations that take a user two to three minutes to complete have hundreds of operators. The standard approach to use KLM for measuring execution time for user interface involves the following:

1. Select one or more scenarios for typical tasks.
2. Have the design detailed enough so that the keystroke-level actions for the various task scenarios may be stated.
3. Determine the optimal method to complete each task scenario, or the method you anticipate users would use.
4. List the physical operators that match to the keystroke-level operations needed to complete the task.
5. Incorporate operators for times when the user must wait for the system to reply, if needed.
6. Insert mental operators for times when the user needs to pause and reflect.
7. Find out how long each operator typically takes to execute.
8. Sum the operators' execution times.
9. Calculate the estimated time to complete the task by summing the operator times.

3 Applications

In this section we study two applications from different domains with their unique requirements and end-user needs that have shaped features of current iteration of our software. In each of the following two sections we briefly introduce these domains, their requirements, and how they influenced the feature set of our tool.

3.1 Pxio

Pxio Gmbh is a growing IT startup originating from DFKI Saarbruecken providing a software solution for display setups. The company offers a cloud based infrastructure and client softwares that allow users to share their multimedia content such as videos and live screen captures from their devices onto phyical displays available on the same network such as that of a conference room.

3.1.1 Overview

Pxio Cloud v2.0 was used as an application for this project. In this iteration of Pxio multiple user interfaces are built into a Single Page Application (SPA) built using VueJs framework (v2). These interfaces provide ways to both browse and interact with data that is living on a cloud micro-service

infrastructure developed using the Microsoft .Net Framework. In the following figure you can see an overview of the main data types used across Pxio services and UIs and their relations

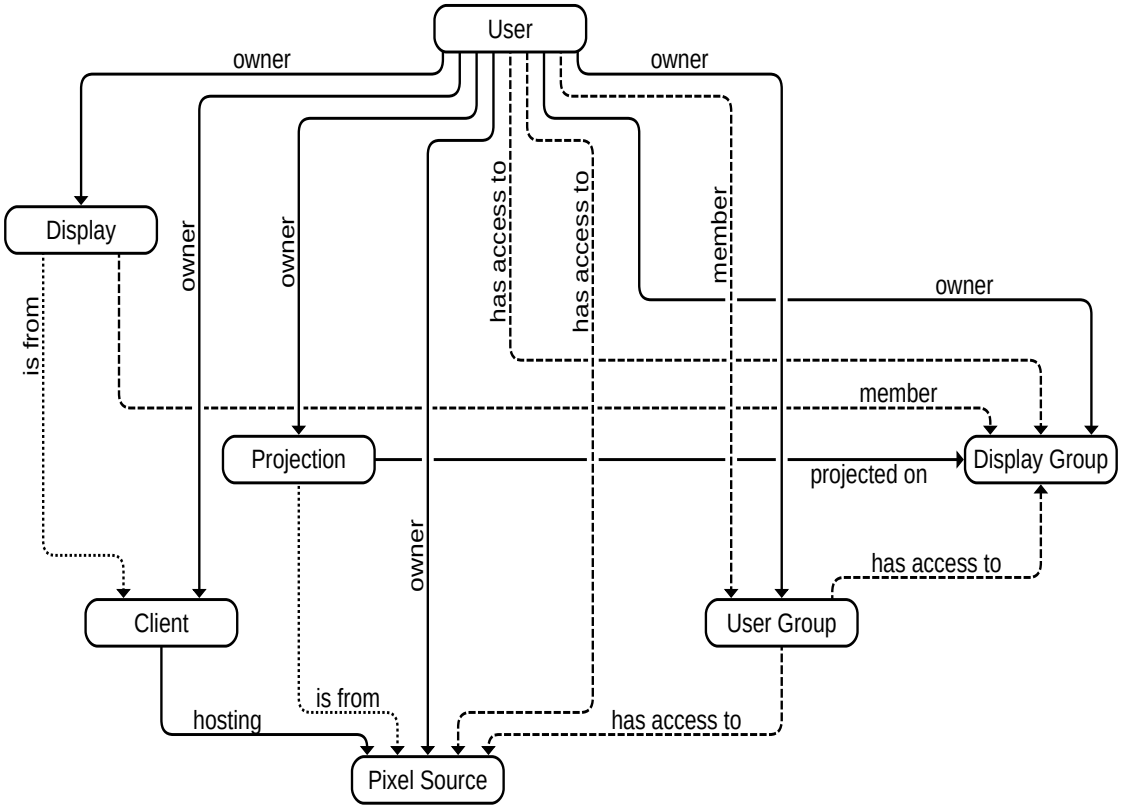


Figure 2: Overview of Pxio Data

The inter-relatedness of entities on Pxio Cloud as can be seen in Figure 2 is why it was mainly interested for us to investigate the feasibility of applying such a tool in this domain. Of course this requires a modelling of this data using RDF vocabularies prior to using our tool. However the flexibility of how this can be achieved makes it beyond the scope of this document.

Following is a glimpse into the existing UIs that live under Pxio Portal, the SPA mentioned earlier. For the purposes of this study and based on the Requirement Gathering we do not discuss all functionality of Pxio Portal and rather narrow down our analysis on the interfaces related to one use-case from each user persona.

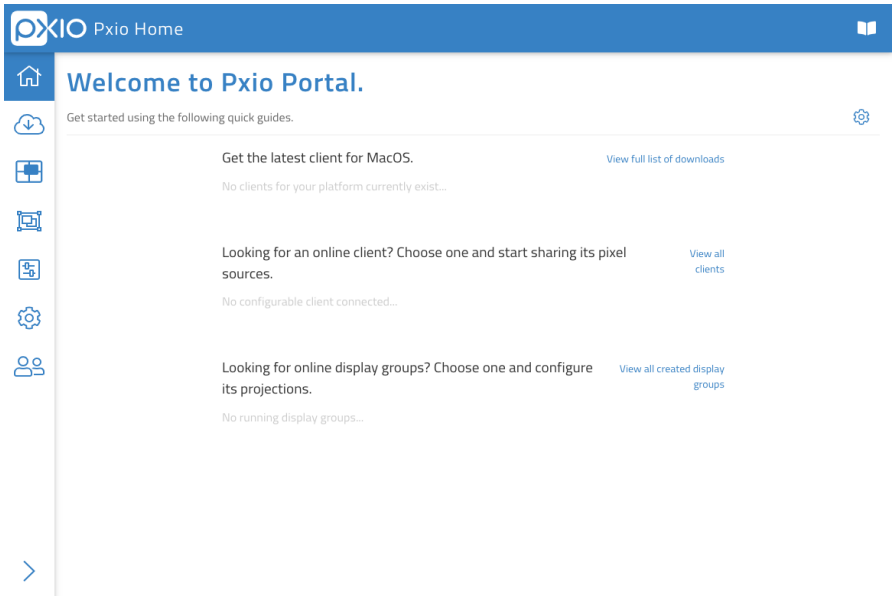


Figure 3: A view of home page of Pxio Portal v2.0

3.1.2 Goals

Existing user interfaces provide a local view over a data that is tightly connected as evident in Figure 2. It also lacks a way for a user to see the overall state and changes in the data. Our goals for this application is to both provide a rough Semantic Web modelling of this data to include as much context as a visualization might require to provide a simple and configurable User Interface that can suit multiple personas.

3.1.3 Personas

During an initial Requirement Gathering we identified three Persona of end-users that may interact with Pxio data

- **Admin**
  This Persona matches any user with full access to all UIs within Pxio Portal. typically such a user is in charge of administrative tasks such as managing users and groups, creating Display Groups, etc.

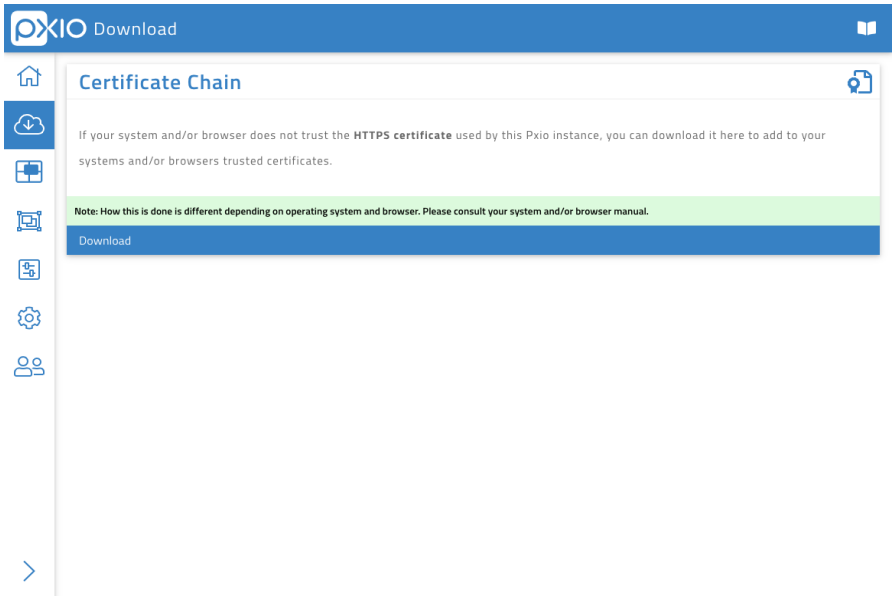


Figure 4: Static UI Specific for Downloads

- **Guest**
  This Persona matches any user with the most limited access to Pxio Portal. Typically such a user can only present their Pixel Sources either in form of video stream or screen capture stream and is able to share them with other users and group or project them on running Display Groups.
- **Standard User**
  Or User for short is any user that is registered on Pxio Cloud and may even have multiple clients connected to it. They are limited only with respect to administrative tasks.

3.1.4 Usecases

For the purpose of this study, we have selected one usecase per persona identified during the requirement gathering phase and investigated the workflow of Pxio Portal and a similar prototype for the same goal designed using our tool. based on the Personas discussed previously these usecases are as follows.

For the Admin Persona, we study the flow of adding a User to a Group. actions in this usecase are listed in the following table

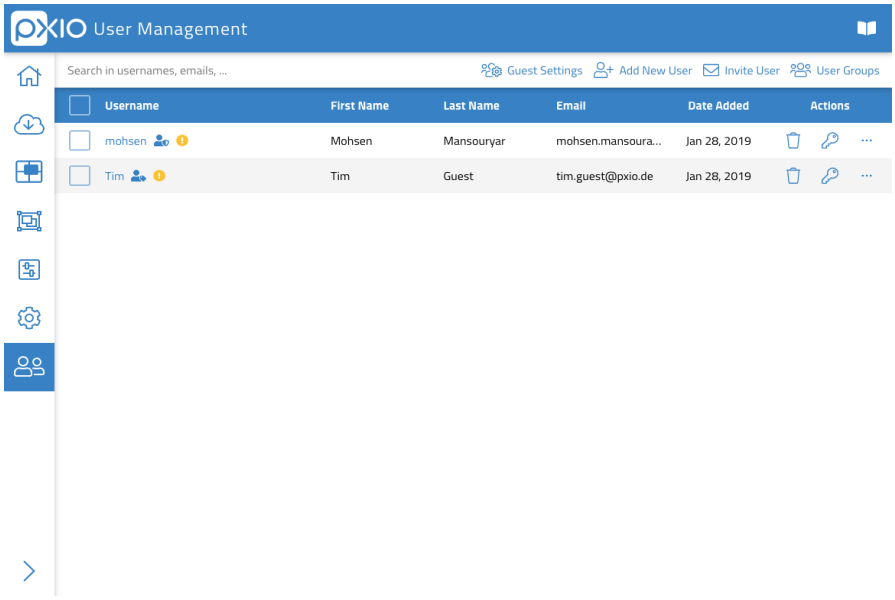


Figure 5: UI Specific for all User and Group Management usecases

Table 1: Admin Usecase: Adding a User to a Group

Pxio Design Actions	**Graph-ical Design Actions**
navigating to the User Management view	navigating to the User Management view
selecting Group Management modal	selecting shortcut to add User to Group
selecting the desired Group	finding the desired User to add
finding the desired user to add	finding desired Group to add User to
toggling user's group membership	dragging the desired User to the Group

For the User Persona, we study the flow of creating a 2-Display Display Group. actions in this usecase are listed in the following table

Table 2: Standard User usecase: Creating a Display Group from 2 Displays

Pxio Design Actions	**Graph-ical Design Actions**
navigating to the Display Configuration view	navigating to the Mapping view
creating an empty Display Configuration	selecting shortcut to add new display group
Finding displays to add	creating an empty Display group
dragging displays onto the center	Finding displays to add
	dragging displays onto the the group (page center)

For the Guest Persona, we study the flow of projecting a Pixel Source on a running Display Group. actions in this usecase are listed in the following table

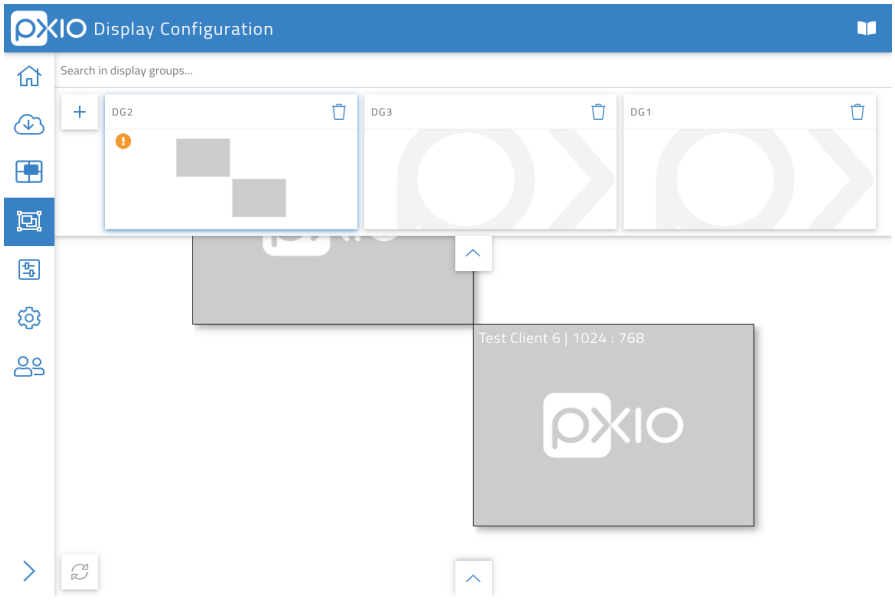


Figure 6: Display Configuration UI

Table 3: Guest usecase: Projecting a Pixel Source

Pxio Design Actions	**Graph-ical Design Actions**
navigating to the Projection Mapping view	navigating to the Mapping view
selecting the desired display group	selecting shortcut to project a pixel source
finding the desired pixel source	selecting the desired display group
dragging the source onto display group	finding the desired pixel source
	dragging the source onto display group

3.2 Summary

4 Methodology

In this section, I present the methodology, which has been used to answer the research questions. The answers can be obtained by conducting experiments. The methodology has the following steps

1. Requirement Gathering: At this step we interview domain experts to get an understanding of their context, existing User Interfaces they are using and what can be interesting and useful to their respective personas in an alternative UI. Appendix A gives a detailed description of how the interview is conducted for the Pxio domain.

2. Usecase Analysis: At this step we identify the key use-cases in each domain and present our approach to realize those in a User Interface designed using Graph-ical. We do not cover all usecases but give further intruction how the tools developed in Graph-ical can be utilized to achive the remaining ones.

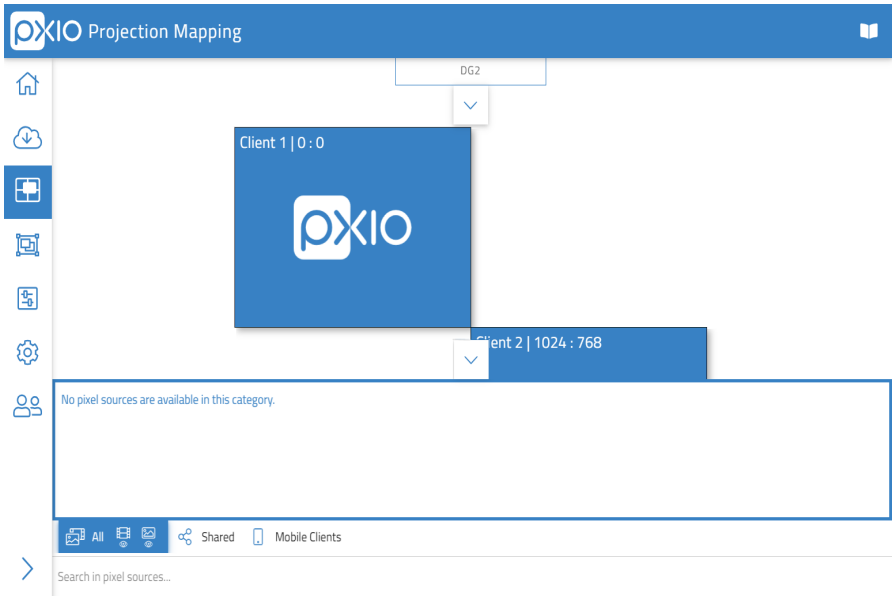


Figure 7: Projection Mapping UI

3. Interface Evaluation-Metric model: At this step we evaluate both presented UIs designed using Graph-ical and their existing counterparts in each respective domain using a number of GUI complexity metrics. These metrics include TODO

4.1 Metrics

4.2 Keystroke-Level Model

A family of predictive models known as Goals, Operators, Methods, and Selection (GOMS) were developed to compare and evaluate goals, methods, and selection rules of skilled, error-free user performances [17]. GOMS techniques model goal hierarchies of defined unit tasks rendered as a composition of action and cognitive operators. The simplest member of the GOMS family is the keystroke-level model (KLM), which predicts the execution time of specific tasks in a desktop environment using a mouse and keyboard. The KLM has been widely utilised to evaluate expert performances of various desktop interfaces, and its aptitude and usefulness have been well demonstrated.

4.3 Requirement Gathering

4.4 Prototypes

4.5 Implementation

Due to the nature of the widget based design of UIs created with Graph-ical plus the possibility of same data or related data being visualized in the same tab, we have decided to choose a reactive

Operator	Time (in seconds)
Classic KLM Operators	
K: keystroke	.08 (135 wpm: best typist) to 1.20 (worst typist)
P: point to an element on screen	1.1
H: homing the hand(s) on the keyboard or other device	0.4
M: mental preperation	1.35
R: system response time	system dependent
B: press or release mouse button	0.1
S: scan for content	2.29
Combined Operators	
Click a Link/ Button	3.73
Pull-Down List (No Page Load)	3.04
Pull-Down List (Page Load)	3.96
Date-Picker	6.81
Cut & Paste (Keyboard)	4.51
Typing Text in a Text Field	2.32
Scrolling	3.96

based approach for the underlying event system. we use ReactiveX (TODO: ref), more specifically rxjs which is its port to Javascript.

The ability to customize Document Object Model (DOM) while creating a UI also required to use a direct approach for visualization in a browser that allows DOM manipulation. to this end we chose d3 [14] library which is a standard and well-known solution to vector based 2D visualization on the web. since d3 allows full control over DOM and our application deals with RDF data, it is natural to adopt RDFa in templating. this ensures our generated UIs and the data they represent are easily extractable via RDF scrapers and similar tools and search engine friendly. TODO: show an example use https://rdfa.info/play/ copy generated html from Graph-ical into this tool and show the extracted data do NOT implement anything, just modify the html so that is uses some RDFa

A Semantic Web Application typically consists of the following components: RDF Parser/Serializer, RDF Store (triple store), RDF Query Engine, and finally an Application component [13]. so far we discussed technologies used in the Application component. For RDF storage in principal any triple-store could work. we chose RDF4J - the successor of the OpenRDF Sesame project [15] - which is an open-source framework for storing, querying, and analyzing RDF data that also provides a SPARQL based query engine. In order to communicate between the application and triplestore we use N3js as the RDF Parser/Serialiyer. N3.js is an implementation of the RDF.js low-level specification that allows working with RDF formatted data in JavaScript quite fast and easy, more importantly supporting all major RDF serialization formats.

Last but not least, UIs designed with Graph-ical are stored in Terse RDF Triple Language (Turtle) format, which is mostly due to its similar syntax to SPARQL.

4.6 Technical Evaluation

1. it evaluates key requirements coming from both applications 2. shows scenarios either in a demo setting or illustrated how the requirements are addressed 3. can be compared to existing state of the art tools that can be applied to meet the same requirement in question 4. a number of quantitative metrics can be measured to show effectiveness of the proposed approaches and to compare against existing tools

TODO: Discus GUI metrics Usability metrics not really an option since they depend on subjective preferences. we instead aim to use complexity metrics such as response time, search time and speed. Also we may use visuali cohesion metrics such as semantic relatedness.

5 Graph-ical Workflow

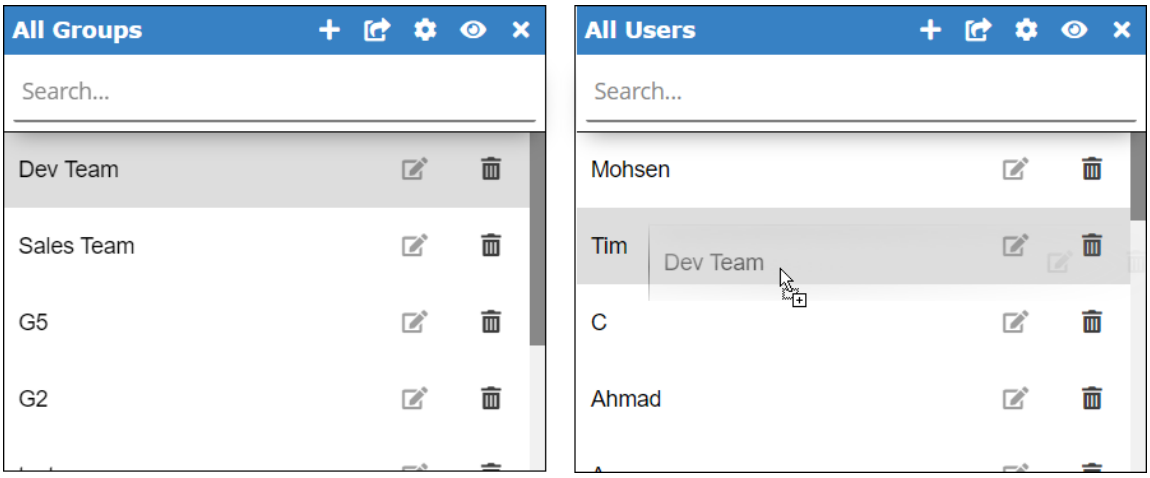


Figure 8: Creating drag behaviour at UI design time by dragging items between two Listview Widgets

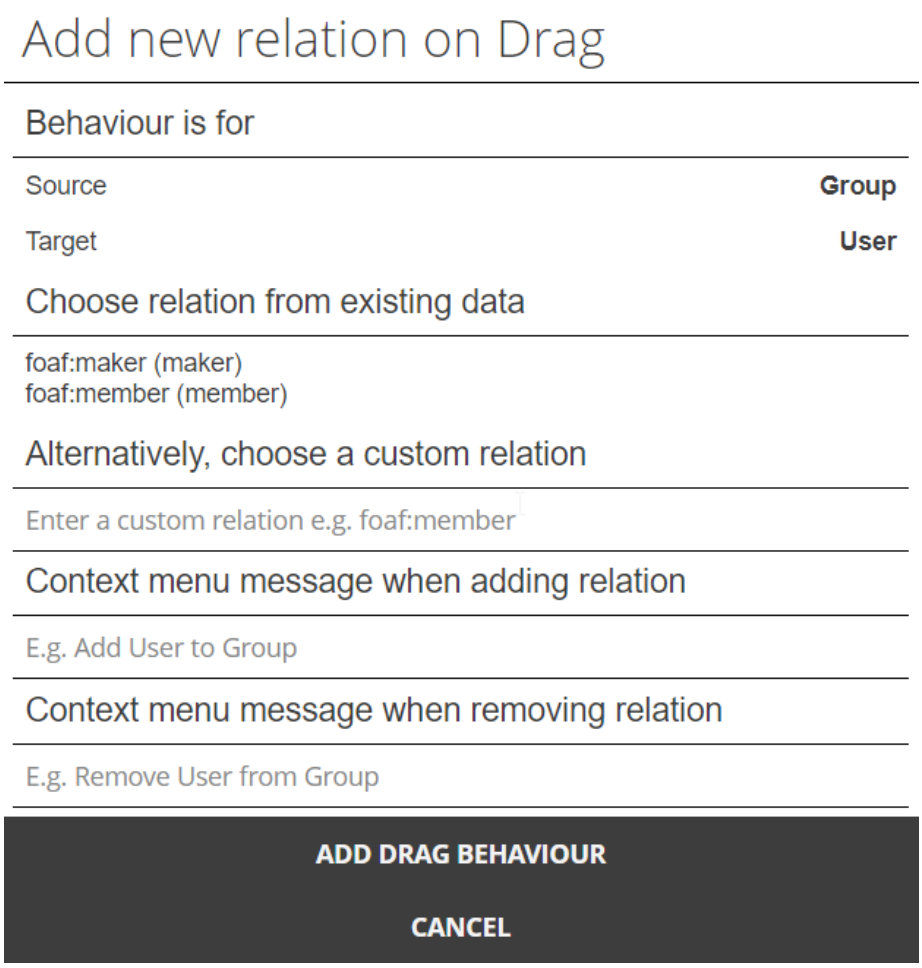
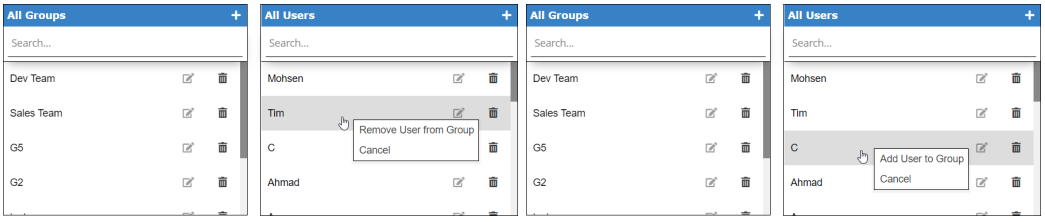


Figure 9: Setting up drag context menus and RDF data pattern



(a) Context menu when RDF triple exists (b) Context menu when RDF triple does not exist

Figure 10: Drag UI at runtime

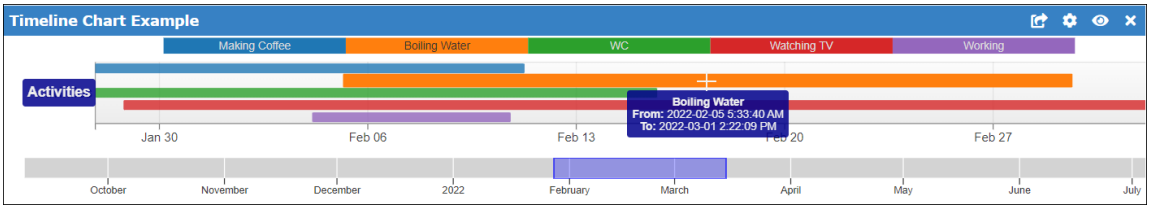
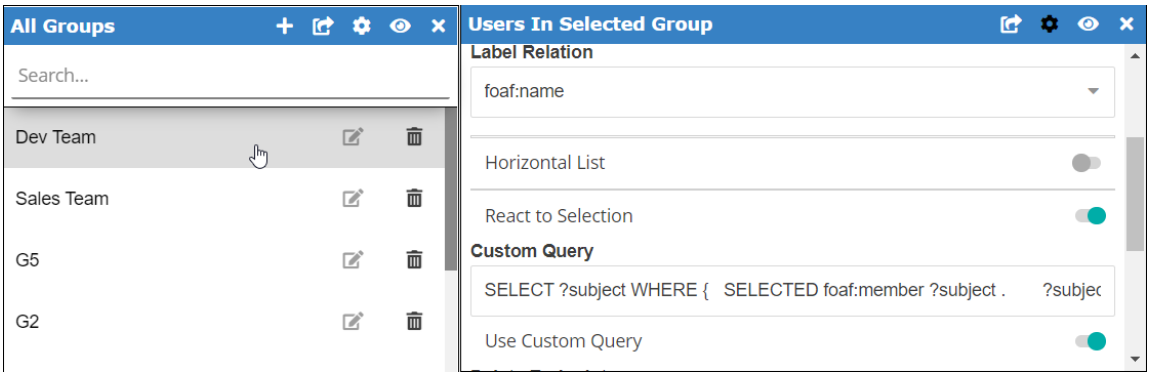
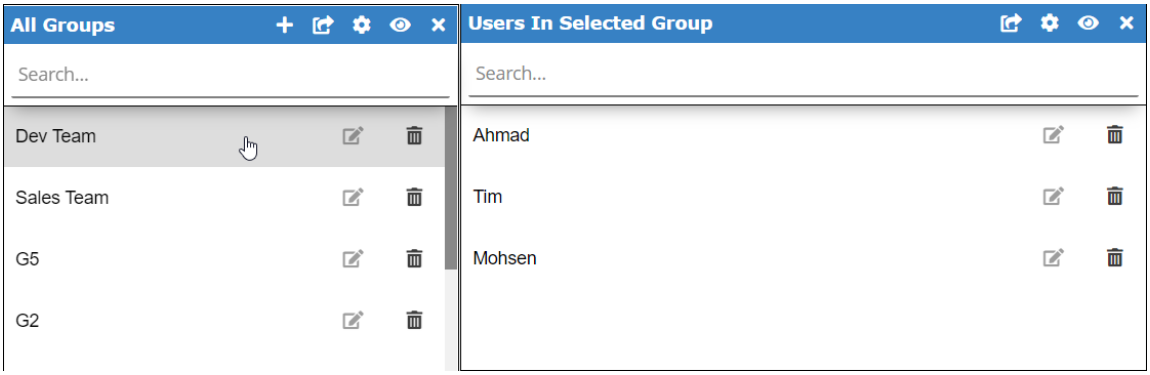


Figure 11: Example Gan Chart for displaying SENSE Activity timelines (using mock data)

listview_react_to_selection/

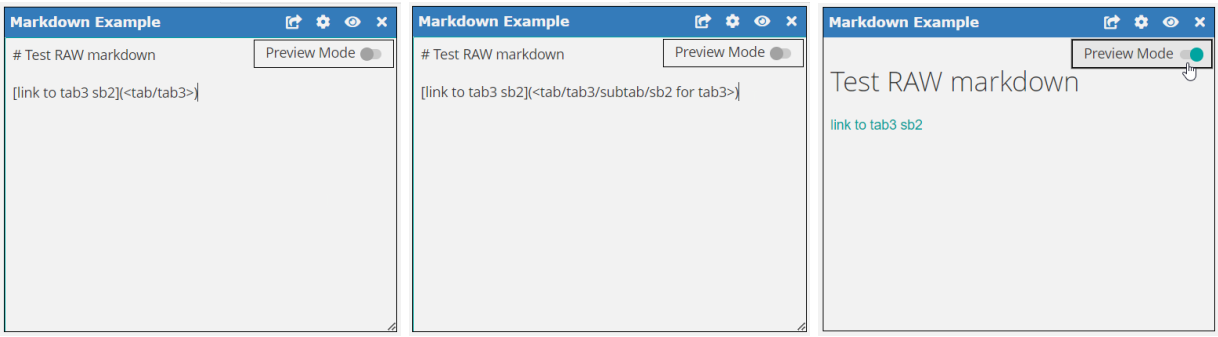


(a) Enabling 'React to Selection' and providing a SPARQL query with 'SELECTED' keyword to filter instances that are displayed



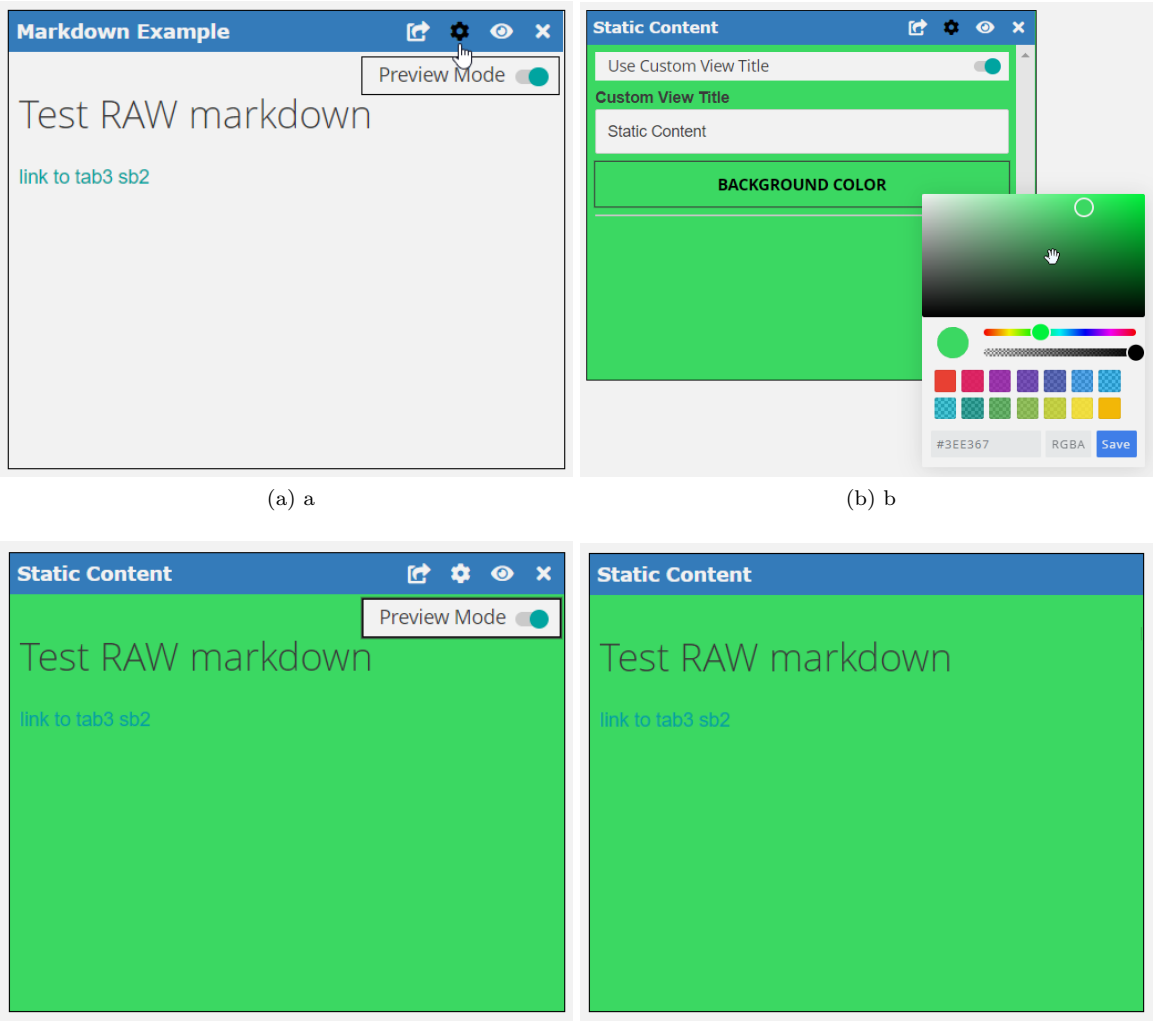
(b) Result when an instance is selected that matches the SPARQL query

Figure 12: How a Listview widget can react to Selection



(a) a (b) b (c) c

Figure 13: How to Link to Tabs and Subtabs with a Markdown Widget



(a) a

(b) b

(c) c

(d) d

Figure 14: How to modify basic widget settings (Title and Background color)

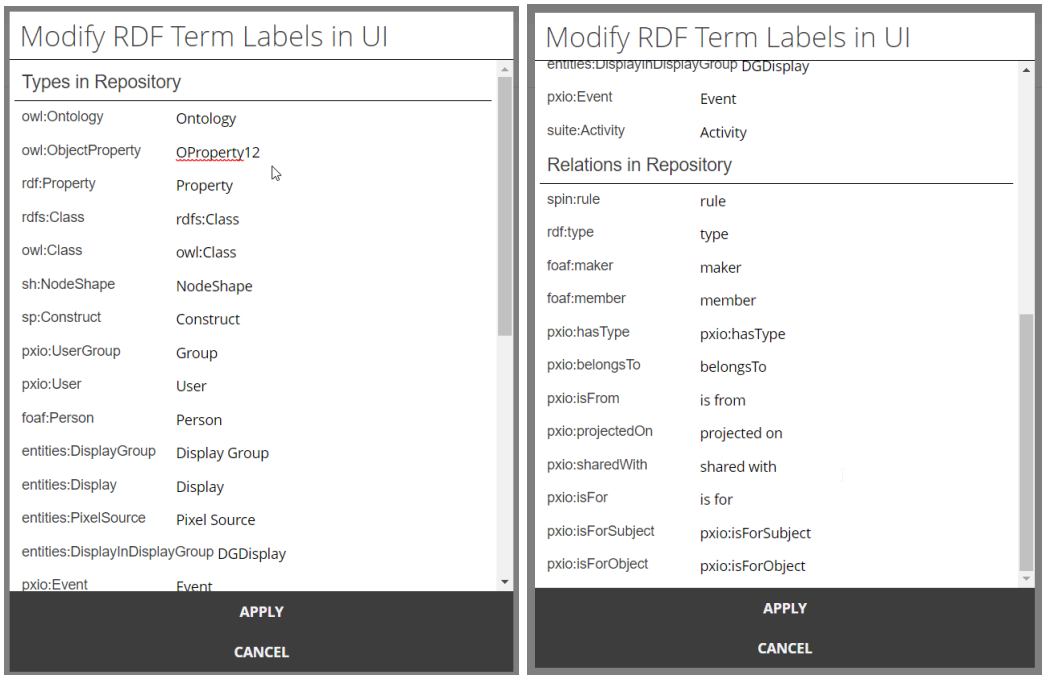
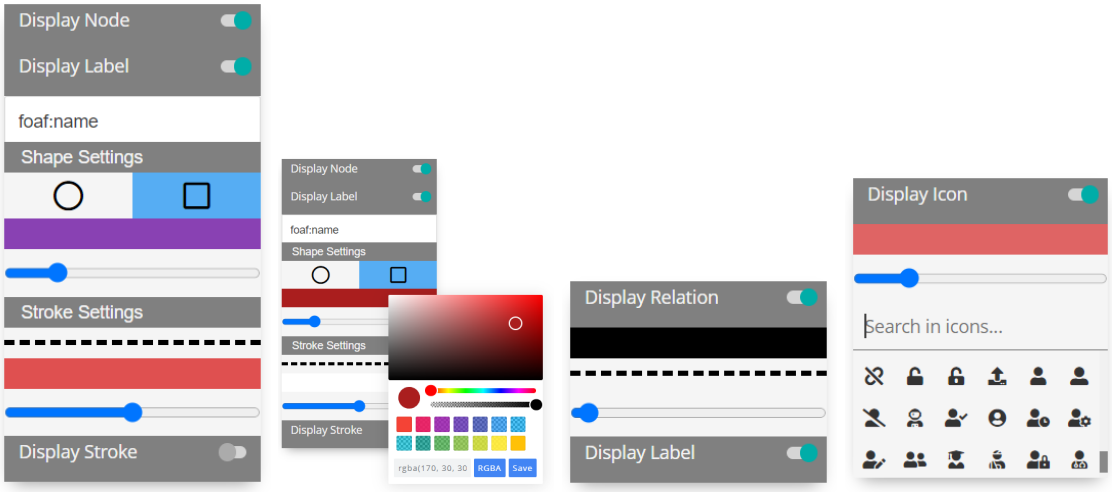


Figure 15: Modifying RDF term labels Globally (Classes and Relations)



(a) a (b) b (c) c (d) d

Figure 16: Context menu controls when Modifying network node and link styles

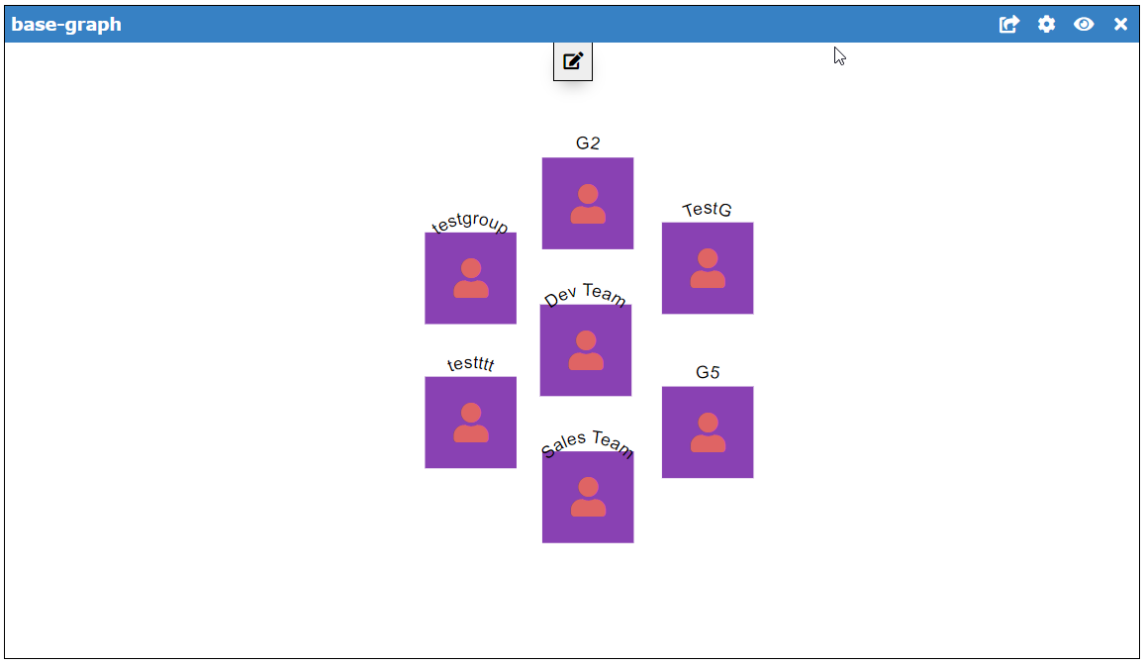


Figure 17: Simple Network with custom node/link styling

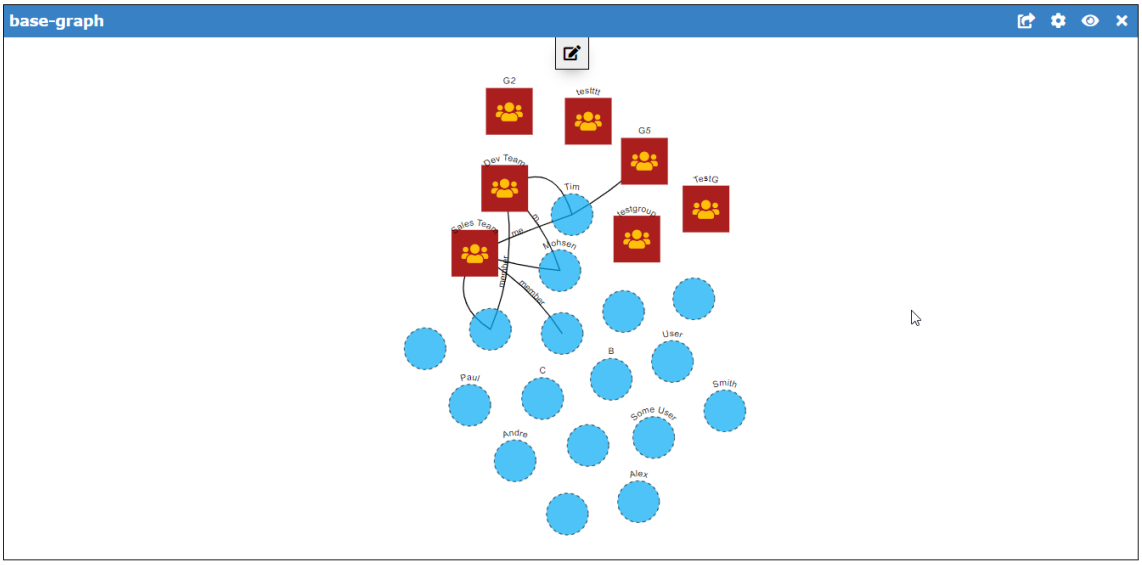
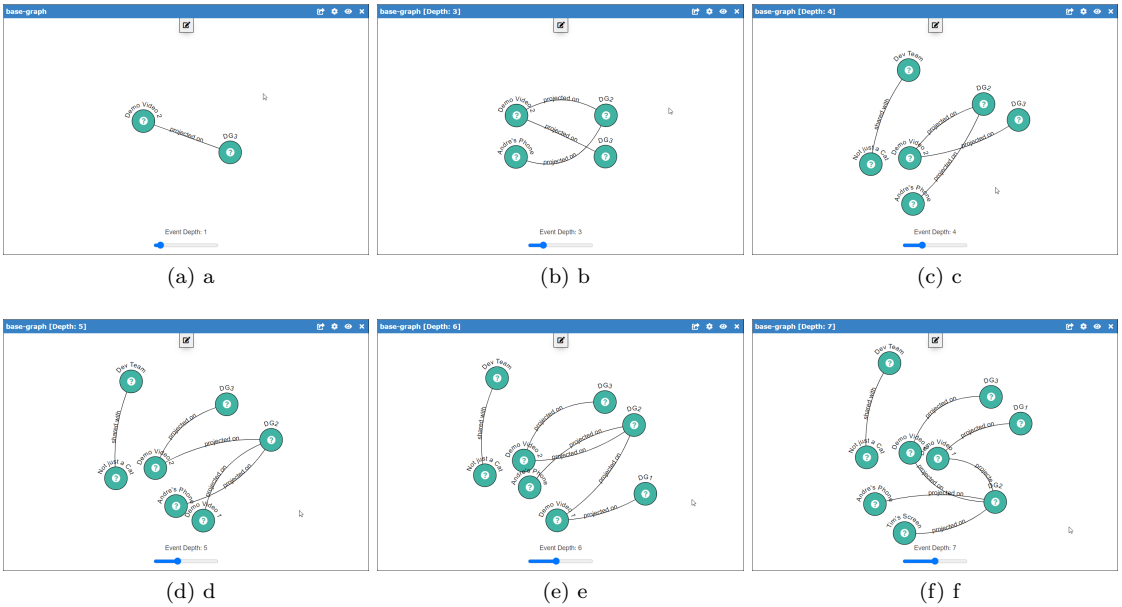


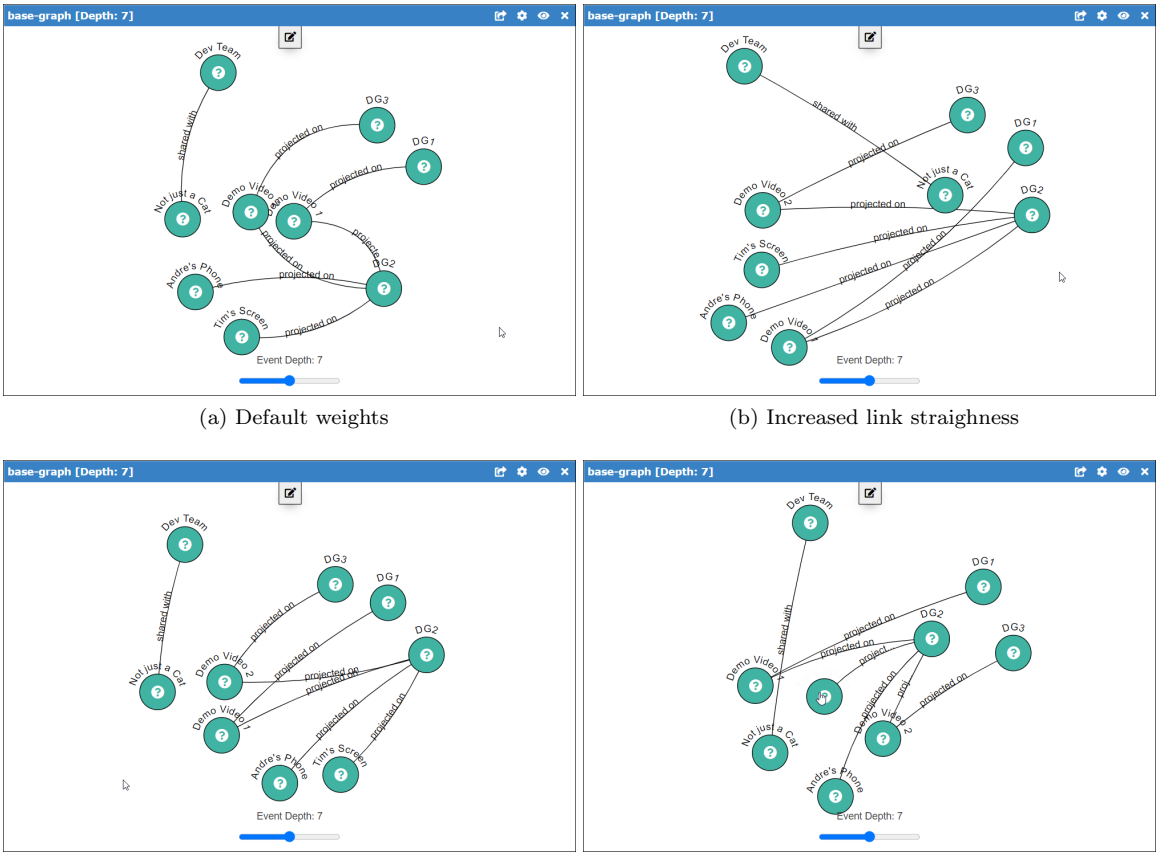
Figure 18: Slightly more complex Network with custom node/link styling



(a) a (b) b (c) c

(d) d (e) e (f) f

Figure 19: Increasing Event Depth when Displaying Network in Event-based Data mode



(a) Default weights

(b) Increased link straighness

(c) Increased link straighness and shortness weights

(d) Increased edge label size weight

Figure 20: Custom forces affecting layouting and readability of a Network

6 Results

We study one usecase per persona using KLM metric to analyze Pxio and SENSE and to compare them with Graph-ical. We calculate the total time for each persona to make both applications comparable.

Table 4: Admin Persona (can manage users and groups)

Pxio Design Operator Sequence	**Graph-ical Design Operator Sequence**
Point to User Management button (P)	Point to User Management tab (P)
Click mouse button (BB)	Click mouse button (BB)
Point to group management button (P)	Point to shortcut to add User to Group (P)
Click mouse button (BB)	Click mouse button (BB)
Point to group list (P)	Point to user list (P)
Scroll to desired group (Scrolling)	Scroll to desired user (Scrolling)
Point to desired group's edit button (P)	Point to group list (P)
Click mouse button (BB)	Scroll to desired group (Scrolling)
Point to user list (P)	Press and hold mouse button (B)
Scroll to the desired user (Scrolling)	Drag user to group item (P)
Point to desired user's membership checkbox (P)	Release mouse button (B)
Click mouse button (BB)	Asynchronously update membership (R)
Asynchronously update membership (R)	

Total time	
6P + 8B + 2(Scrolling) + R = 16.32 sec	5P + 6B + 2(Scrolling) + R = 15.02 sec

Table 5: User Persona (can configure display groups). Usecase: Creating a 2-Display Display Group

Pxio Design Operator Sequence	**Graph-ical Design Operator Sequence**
Point to Display Configuration button (P)	Point to Mapping tab (P)
Click mouse button (BB)	Click mouse button (BB)
Point to display group top bar (P)	Point to shortcut to add new display group (P)
Pull down display group bar (Pull-Down List (No Page Load))	Click mouse button (BB)
Point to add new display configuration button (P)	Point to add button on display group list (P)
Click mouse button (BB)	Click mouse button (BB)
Point to display group name box (P)	Point to display group name box (P)
Type the name for display group (Typing Text in a Text Field)	Type the name for display group (Typing Text in a Text Field)
Point to add button (P)	Point to add button (P)
Click mouse button (BB)	Click mouse button (BB)
Asynchronously add empty display group (R)	Asynchronously add empty display group (R)
Point to display bottom bar (P)	Point to display list (P)
Pull up display bar (Pull-Down List (No Page Load))	Scroll to first display to add (Scrolling)
Point to display list (P)	Point to first display to add (P)
Scroll to first display to add (Scrolling)	Press and hold mouse button (B)
Point to first display to add (P)	Drag display to center (P)
Press and hold mouse button (B)	Release mouse button (B)
Drag display to center (P)	Asynchronously add display to group (R)
Release mouse button (B)	Point to display list (P)
Asynchronously add display to group (R)	Scroll to second display to add (Scrolling)
Point to display list (P)	Point to desired spot display to add (P)
Scroll to second display to add (Scrolling)	Press and hold mouse button (B)
Point to desired spot display to add (P)	Drag display to desired spot (P)
Press and hold mouse button (B)	Release mouse button (B)
Drag display to desired spot (P)	Asynchronously add display to group (R)
Release mouse button (B)	
Asynchronously add display to group (R)	

Total time	
12P + 10B + 2(Pull-Down List (No Page Load)) + 2(Scrolling) + 3R = 31.2 sec	11P + 12B + 2(Scrolling) + 3R = 24.22 sec

Table 6: Guest Persona (can use running sources and display groups) Usecase: Projecting a Pixel Source on a Running Display Group

Pxio Design Operator Sequence	**Graph-ical Design Operator Sequence**
Point to Projection Mapping button (P)	Point to Mapping tab (P)
Click mouse button (BB)	Click mouse button (BB)
Point to display group top bar (P)	Point to shortcut to add new projection (P)
Pull down display group bar (Pull-Down List (No Page Load))	Click mouse button (BB)
Point to display group list (P)	Point to display group list (P)
Scroll to desired display group (Scrolling)	Scroll to desired display group (Scrolling)
Point to desired display group (P)	Point to desired display group (P)
Click mouse button (B)	Click mouse button (B)
Point to pixel source bottom bar (P)	Point to pixel source list (P)
Pull up pixel source bar (Pull-Down List (No Page Load))	Scroll to desired pixel source (Scrolling)
Point to pixel source list (P)	Point to desired pixel source (P)
Scroll to desired pixel source (Scrolling)	Press and hold mouse button (B)
Point to desired pixel source (P)	Drag pixel source to display group (P)
Press and hold mouse button (B)	Release mouse button (B)
Drag pixel source to display group (P)	Asynchronously add new projection (R)
Release mouse button (B)	
Asynchronously add new projection (R)	

Total time	
8P + 5B + 2(Pull-Down List (No Page Load)) + 2(Scrolling) + R = 24.3 sec	7P + 7B + 2(Scrolling) + R = 17.32 sec

Table 7: SENSE usecase: widget configuration for developing UIs (timeseries data chart plus statistics)

Garafana Design Actions	**Graph-ical Design Actions**
creating a new dashboard	adding a time series visualisation
adding a time series visualisation	adding time series data to widget
configuring time series with desired data	configuring widget for end-user
adding a stat panel visualisation	adding a stat display for the data
configuring the stat panel for desired data	configuring stat display by writing a sparql query
	configuring stat display for end-user
	adding RDF terms translations for developer
	exporting UI template as ttl file

Table 8: SENSE usecase for developer: has access to SENSE data, wants to build applications and UIs (task specific) for end-users

Garafana Design Actions	**Graph-ical Design Actions**
Accessing a dashboard designed by an expert	importing UI template setup from expert
Creating two new persona specific dashboards	split UI into a tab with two subtabs
Copy timeseries and stat panels into separate dashboards	adding markdown information in each subtab with shortcuts from the default tab
Adding text panels in each dashboard with descriptions for respective persona	

Garafana Design Operator Sequence	**Graph-ical Design Operator Sequence**
Point to dashboards button (P)	Point to add visualisation button (P)
Point to new dashboard button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to time series visualisation (P)
Point to add a new panel button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to visualisation settings button (P)
Point to visualisation type dropdown (P)	Click mouse button (BB)
Click mouse button (BB)	Scroll to object type input (Scrolling)
Scroll to time series visualization (Scrolling)	Point to object type input (P)
Point to time series visualization (P)	Click mouse button (BB)
Click mouse button (BB)	Scroll to desired data type (Scrolling)
Point to data source dropdown (P)	Point to desired data type (P)
Click mouse button (BB)	Click mouse button (BB)
Point to desired source (P)	Point to Done button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to query type dropdown (P)	Point to Zoom toggle (P)
Click mouse button (BB)	Click mouse button (BB)
Point to 'Series' (P)	Point to visualisation settings button (P)
Click mouse button (BB)	Click mouse button (BB)
Press Esc button (K)	Point to add visualisation button (P)
Point to Add panel button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to markdown visualisation (P)
Point to add a new panel button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to widget text input (P)
Point to visualisation type dropdown (P)	Prepare sparql query to compute desired stat (M)
Click mouse button (BB)	Type in sparql query to compute desired stat (Typing Text in a Text Field)
Scroll to stat visualisation (Scrolling)	Point to Translate RDF Terms button (P)
Point to stat visualisation (P)	Click mouse button (BB)
Click mouse button (BB)	Choose a term to modify translation (M)
Point to data source dropdown (P)	Point to desired term for translation (P)
Click mouse button (BB)	Type in label for desired term (Typing Text in a Text Field)
Point to desired source (P)	Point to apply button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to query type dropdown (P)	Point to export template button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to 'Series' (P)	Point to template name field (P)
Click mouse button (BB)	Type in template name (Typing Text in a Text Field)
Press Esc button (K)	Point to Ok button (P)
Point to save dashboard button (P)	Click mouse button (BB)
Click mouse button (BB)	
Point to dashboard name field (P)	
Type in dashboard name (Typing Text in a Text Field)	
Point to Save button (P)	
Click mouse button (BB)	

Total time	
20P + 36B + 2(Scrolling) + 2K + 1(Typing Text in a Text Field) = 36.1 sec	17P + 28B + 2(Scrolling) + 2M + 3(Typing Text in a Text Field) = 39.1 sec

Table 9

Garafana Design Operator Sequence	**Graph-ical Design Operator Sequence**
Point to dashboards button (P)	Point to import template button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to desired dashboard (P)	Point to template file for import (P)
Click mouse button (BB)	Click mouse button (BB)
Point to first widget's title (P)	Point to tab option button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to More... button (P)	Point to Add subtab option (P)
Point to Copy button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to subtab name field (P)
Point to dashboards button (P)	Type in first subtab name (Typing Text in a Text Field)
Point to new dashboard button (P)	Point to ok button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to Paste panel from Clipboard (P)	Point to Add subtab option (P)
Click mouse button (BB)	Click mouse button (BB)
Point to add a new panel button (P)	Point to subtab name field (P)
Click mouse button (BB)	Type in second subtab name (Typing Text in a Text Field)
Point to visualisation type dropdown (P)	Point to ok button (P)
Click mouse button (BB)	Click mouse button (BB)
Scroll to text visualisation (Scrolling)	Point to subtab dropdown (P)
Point to text visualisation (P)	Click mouse button (BB)
Click mouse button (BB)	Point to default subtab (P)
Scroll to text content field (Scrolling)	Click mouse button (BB)
Point to text content field (P)	Point to first widget's move button (P)
Click mouse button (BB)	Click mouse button (BB)
Prepare information text for subtab (M)	Point to desired subtab to move to (P)
Type in desired text for current view (Typing Text in a Text Field)	Click mouse button (BB)
Press Esc button (K)	Point to add visualisation button (P)
Point to dashboards button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to markdown visualisation (P)
Point to initial dashboard (P)	Click mouse button (BB)
Click mouse button (BB)	Point to widget text input (P)
Point to second widget's title (P)	Prepare information text for subtab (M)
Click mouse button (BB)	Type in desired text for current view (Typing Text in a Text Field)
Point to More... button (P)	Point to subtab dropdown (P)
Point to Copy button (P)	Click mouse button (BB)
Click mouse button (BB)	Point to default subtab (P)
Point to dashboards button (P)	Click mouse button (BB)
Point to new dashboard button (P)	Point to second widget's move button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to Paste panel from Clipboard (P)	Point to desired subtab to move to (P)
Click mouse button (BB)	Click mouse button (BB)
Point to add a new panel button (P)	Point to add visualisation button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to visualisation type dropdown (P)	Point to markdown visualisation (P)
Click mouse button (BB)	Click mouse button (BB)
Scroll to text visualisation (Scrolling)	Point to widget text input (P)
Point to text visualisation (P)	Prepare information text for subtab (M)
Click mouse button (BB)	Type in desired text for current view (Typing Text in a Text Field)
Scroll to text content field (Scrolling)	
Point to text content field (P)	
Click mouse button (BB)	
Prepare information text for subtab (M)	
Type in desired text for current view (Typing Text in a Text Field)	

Total time

24P + 40B + 4(Scrolling) + 2M + K + 2(Typing Text in a Text Field) = 53.7 sec	23P + 38B + 2M + 4(Typing Text in a Text Field) = 47.64 sec

Garafana Design Operator Sequence	**Graph-ical Design Operator Sequence**
Point to dashboards button (P)	Point to import template button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to desired dashboard (P)	Point to template file for import (P)
Click mouse button (BB)	Click mouse button (BB)
Scan linechart information (S)	Point to shortcut for power consumption stats (P)
Identify desired interval to look into (M)	Click mouse button (BB)
Hover rendered linechart (P)	Scan linechart information (S)
Press and hold mouse button (B)	Identify desired interval to look into (M)
Drag over desired interval (P)	Hover rendered linechart (P)
Release mouse button (B)	Zoom in on desired interval (Scrolling)
Total time	
4P + 6B + S + M = 8.64 sec	4P + 6B + S + M + (Scrolling) = 12.6 sec

Table 10: SENSE usecase for technician: Can modify designed UIs, more access over all visualizations, can interpret more abstract visualizations checks state of sensors and data. Usecase: monitor power consumption linechart over a specific period of time, identify peaks and drops

Garafana Design Actions	**Graph-ical Design Actions**
Accessing a dashboard designed by a developer	import a UI designed by developer
Scan time series information from a line chart	locate shortcut to power consumption stats
Selecting a particular time interval for inspection	scan information displayed on a linechart
	zoom in chart for a particular time interval

Table 11: SENSE usecase for Inhabitant: zero RDF knowledge, no way of modifying UI, should be able to interpret visualizations. Usecase: finding a particular sensor from a map of the house, locating it in a table and checking its status (on/off)

Garafana Design Actions	**Graph-ical Design Actions**
Accessing a dashboard designed by a developer	import a UI designed by developer
Locating desired sensor on a table	locate shortcut to sensor information
Checking sensor state	locate sensor on a map and clicking on it
	checking sensor state from table (on/off)

Garafana Design Operator Sequence	Graph-ical Design Operator Sequence
Point to dashboards button (P)	Point to import template button (P)
Click mouse button (BB)	Click mouse button (BB)
Point to desired dashboard (P)	Point to template file for import (P)
Click mouse button (BB)	Click mouse button (BB)
Identify desired sensor to look into (M)	Point to shortcut for sensor information (P)
Point to desired sensor row (P)	Click mouse button (BB)
Identify sensor state column (S)	Scan map to find desired sensor (S)
	Point to sensor on the map (P)
	Click mouse button (BB)
	Point to sensor row in status table (P)
	Identify sensor state column (S)

Total time	
3P + 4B + M + S = 7.34 sec	5P + 8B + 2S = 10.88 sec

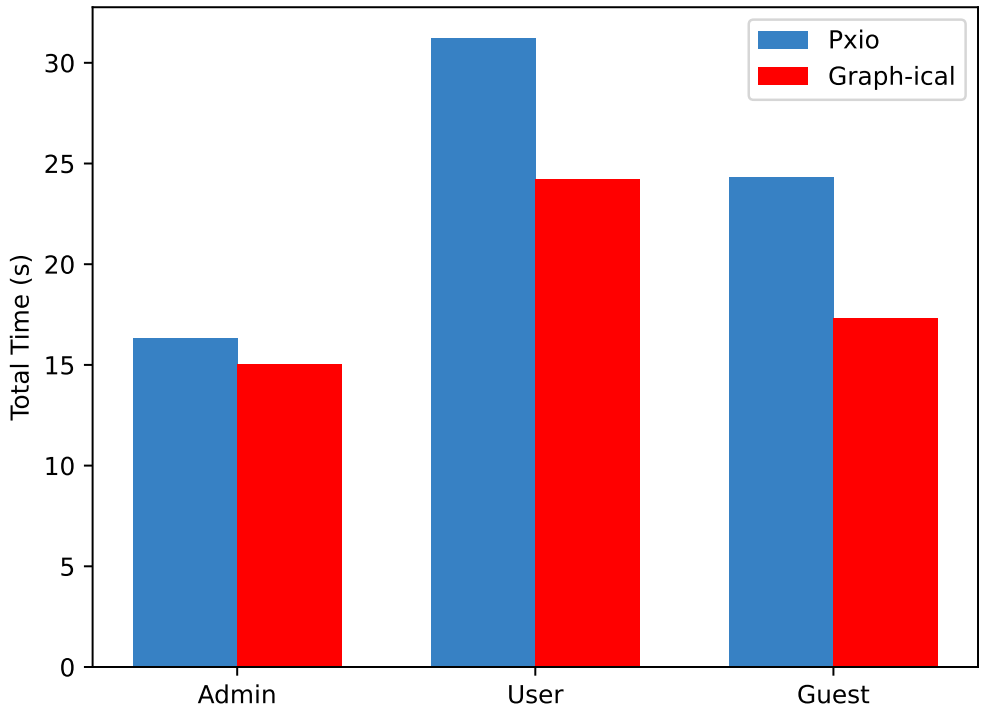


Figure 21: Comparison between Pxio and Graph-ical

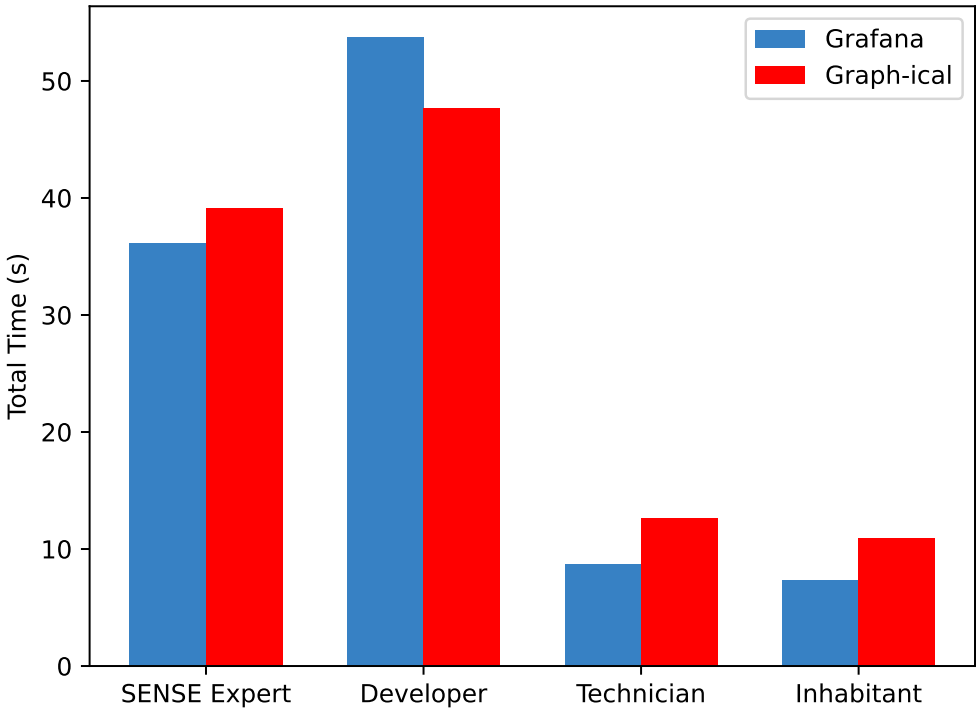


Figure 22: Comparison between Grafana and Graph-ical

To summarize and compare results from the tables above, we present bar plots comparing the total time for each persona in both Pxio and SENSE applications. We can see from the comparison in Fig. 21 and Fig. 22 that Graph-ical achieves similar results to Pxio and Grafana (SENSE) and in some cases we can say Graph-ical outperforms Pxio and Grafana (SENSE). To elaborate on why Graph-ical performs similar (or better in some cases), we need to note that Graph-ical provides task-specific visualizations that helps the user focus on what is their current intent to use such visualizations, rather than seeing all visualizations in one place, which is the case for Pxio and SENSE visualizations. The task-specific focus of Graph-ical helps users find relevant components faster and to perform less number of actions to achieve what they can achive in Pxio and Grafana (SENSE).

7 Conclusion and Future Work

7.1 Context-Awareness

While it is possible to use any RDF repository without any changes in Graph-ical, we recommend a data extension process during which we can increase the level of expressivity of data by adding new triples that help UI understand it better and makes it easier to configure. Essentially what we are proposing here is to use the tools from RDFS and mainly OWL to increase the expressivity of our data model such that certain statements that are useful in its visualization can be inferred from it. these steps will both remove the need to have domain-specific code in the application component and make the UI configuration faster and simpler.

1. RDFS allows giving a human readable name to any resource via ***rdf:label*** property. in case that we have data with other properties for their display name in a UI, we can use property union to state this. For instance assume we have the following triples

```
:Person1 :personName "Anna" .
:Movie1 :movieTitle "K-Pax" .
```

If we now add the following two statements, stating that these properties are indeed intended to be labels

```
:personName rdfs:subPropertyOf rdfs:label .
:movieTitle rdfs:subPropertyOf rdfs:label .
```

Then a presentation engine that queries for ***rdf:label*** against a triplestore with an inference engine will find these labels by RDFS inferencing rules.

2. In Semantic Web applications it is not uncommon to have data coming from multiple sources. this could lead to a scenario where multiple types point essentially to the same class of data or the same relation. In order to disambiguate this and make sure widgets that visualize either types react to changes in one another we need to make such relations explicit. using the expressivity of OWL we can declare equivalence between such classes via the ***owl:equivalentClass*** property (and similarly for properties via ***owl:equivalentProperty***).

3. Specifying a property to be of the type ***owl:InverseFunctionalProperty*** will help UI visualize it differently to indicate uniqueness for its objects.

4. Being able to distinguish between properties that link to a resource vs those that link to xml data types can also help editing tools display UI accordingly. this can be achieved in OWL via assigning properties to either ***owl:ObjectProperty*** or ***owl:DatatypeProperty*** respectively.

8 Appendix A

8.1 Pxio Requirement Interview

This Interview is conducted as the Requirement Gathering phase for the Pxio domain. It is aimed to understand the context within this domain and identify requirements that need to be met for an alternative UI using Graph-ical. At the time of this interview an existing interface with key elements that could potentially be used in such a UI was presented and illustrated to the participants to get their feedback and evaluate the ideas before adaptations based on the information gathered in this phase. The rest of the sections in this chapter contains information that is shared with participants regarding the interview process and questions that are asked without any changes.

8.1.1 Goal

Goal of this interview is to meet requirements for the master's thesis "Domain-Independent and Customizable User Interface Builder for RDF Data" We discuss if such an interface is relevant for an End-User and what properties and features it must have.

8.1.2 Participants

Attendees are limited to experts who are familiar with existing Pxio user interfaces and the overall product and entities. They are also familiar with End-User needs and requirements.

This interview is conducted in a remote semi-structured manner where participants are asked a scripted set of questions over a video call. There are both open and closed questions and an interactive demo which is used to answer some questions.

8.1.3 Initial Discussion (30 minutes)

In the beginning section we ask participants about data and how they would like to see it visualized. We also ask what kind of interactions they are interested in, and generally what information from data they want to see. We ask them what is the most important aspect of Pxio data for a visualization. This part is divided into the following sections:

8.1.4 User Context

1. What is the topic of your study/profession?
2. What are your areas of expertise in Pxio?
3. Have you ever worked with Linked data? If yes please elaborate on the tools used and experience.
4. Have you ever experienced visualizing Linked data (or relational data)? If yes, which tools?
   (a) What did you like about these tools?
   (b) What you didn't like about these tools?

(c) What is your knowledge and experience regarding Graphs and Graph-based interfaces?

8.1.5 Pxio

1. What existing tools and interfaces do you use to interact with Pxio data (such as users/display groups/...)?
   (a) Are these tools web-based? Desktop? Or mobile?
   (b) What are essential features of these tools that you need?
   (c) What features about these tools do you like?
   (d) What features you don't like?
   (e) What other features you may need which these tools don't provide?
2. Do you feel like existing tools are suitable enough for End-Users?
   (a) Are they intuitive and user friendly enough?
   (b) Do they provide all necessary information for users to know about their data?
   (c) Do they provide all interactions with data that is required?
   (d) Would end-users be interested in a new tool at all?
3. How would you visualize Pxio data if you were to use a different tool?
   (a) What kind of visualizations would you like to see? (how would you like to see pxio data) (ask participant to visit https://datavizproject.com/ to get some ideas)
   (b) What kind of user interactions would you like to have?
   (c) How would you want to access this tool? As a web-based app, desktop, or mobile?
4. Are there any existing tools in other domains that you would like to use for Pxio data?
   (a) If not, are there any tools that are closest to what you think Pxio end-users need?
   (b) If yes, please elaborate.
5. If you could allow end-users to customize their interface, what knowledge would you expect them to have?
6. What do you think is the knowledge level of End-Users regarding RDF?

8.1.6 User Interface

1. Considering what End-Users need to see and be able to interact with, how would you prioritize the following in display data?
   (a) User
   (b) Group
   (c) Pixel Source

(d) Display
(e) Display Group
(f) Projection

2. For each entity in 1, please prioritize required interaction with this data
   (a) Add
   (b) Remove
   (c) Update
3. For each entity in 1, how would you visualize them using their properties? (e.g. User can be visualize by their name)
   (a) What properties would you like to visualize and how?
   (b) What properties would you not visualize? Why?
4. For each entity in 1
   (a) Would you like to be able to search through all instances of them in data?
   (b) Would you like to visualize all or just a subset of their instances at a time?
5. Would you like to see all entities mentioned in 1 (current state) in a single UI?
   (a) How would you visualize that?
   (b) What information would be emphasized?
   (c) What information would you skip?
6. If you were to split User Interface into parts that enable interaction and view on subsets of the above entities:
   (a) How many splits would you like to have? why?
   (b) how would you assign entities to splits? (e.g. User and Group can be in the same split)
7. How would you prioritize the following changes in data? (split them if necessary)
   (a) A User/Group/DisplayGroup is created/deleted
   (b) A Pixel Source/Display is discovered/removed
   (c) A User is ¡added to¿/¡remove from¿ a group
   (d) A Pixel Source is ¡shared with¿/¡unshared with¿ a User/Group
   (e) A Pixel Source is ¡projected on¿/¡removed from¿ a Display Group
   (f) A Display Group is ¡shared with¿/¡unshared with¿ a User/Group
8. Would you be interested in seeing how user interactions change data as a function of time? (Events that remove/add relations and instances)
   (a) If so, how would you like to visualize that?
   (b) Would you like to have such visualizations in each UI split or just in one place? (assuming each UI split only shows changes regarding data in the split)
9. Is there any other thing that end-users might find useful/interesting that we've missed?

8.1.7 Concepts

1. In terms of viewing entity instances and their relations from User Interface.1, How would you interpret the following?
2. Given a list of Users and a list of Groups such as in the following,
   (a) How would you add a user to a group?
   (b) How would you expect to see more details about user "Paul"?
   (c) How would you add a new group?
3. In terms of instance properties,how would you interpret the following?
   (a) Would you visualize instance details for an entity from User Interface.1 differently? If so, how?
4. Considering Displays, Display Groups and Projections in the following illustration
   (a) How do you interpret this illustration?
   (b) How would you add/remove a display to/from the selected Display Group?
   (c) How would you add/remove a display to/from a different (not selected) Display Group?
   (d) How would you modify a display group?
   (e) How would you add/remove a projection to/from a Display Group?
   (f) How would you add/remove a projection to/from the selected Display Group?
   (g) How would you want to see all projections in a Display Group?
   (h) How would you want to see all Display Groups with a projection from a given source?
5. How do you interpret the following series of illustrations? (left to right, top to bottom) (arrows are for guidance only)

8.1.8 Prototype Reveal (3-5 minutes)

Observe users as they interact with the prototype. Answer their questions (if any) and take note of their feedback/struggles.

8.1.9 Tasks (roughly 30 minutes (3 each task))

1. Share a Pixel Source with a Group
   (a) Look for Sharing UI
   (b) Locate "Not just a Cat" source
   (c) Share with group "Dev Team"
2. Add a new Group
   (a) Locate User Management UI

(b) Add a group named “Test Group”

3. Add members to a Group
   (a) Add two users to “Test Group”
4. Find members of a Group
   (a) Find all members of “Sales Team”
5. Delete a Group
   (a) Delete “Test Group”
6. Create a Display Group
   (a) Add a display group named “DG_test”
   (b) Add two displays to DG_test
7. Configure a Display Group
   (a) Position two displays in DG2 in a “Horizontal” layout
8. Project a Pixel Source
   (a) Project “Demo Video 1” on DG_test
   (b) Remove the new projection
9. Find details about pixel source “Not just a cat”
10. Query for a particular Event
    (a) Locate Event Timeline UI
    (b) Search if user “Alex” has appeared in any recent interactions?

8.1.10 Final Discussion (5 to 10 minutes)

Let users discuss their experience. If they missed something during tasks (other ways to perform the task) tell them. Ask them about their understanding of different visualizations used and network templates. Also ask:

1. In your opinion, would an end-user like to be able to customize this UI (prototype)?
   (a) If yes, how? (e.g. being able to change background colors, use icons, . . .)
2. How would you integrate this UI into your product? (iframe, direct, standalone app, . . .)

References

[1] IsaViz: A Visual Authoring Tool for RDF. https://www.w3.org/2001/11/IsaViz/.

[2] Linked Data W3C Standard. https://www.w3.org/standards/semanticweb/data.

[3] LodLive. http://en.lodlive.it/.

[4] OWL 2 Web Ontology Language Document Overview. https://www.w3.org/TR/owl2-overview/.

[5] RDF Gravity. http://semweb.salzburgresearch.at/apps/rdf-gravity/.

[6] RDF Schema (RDFS) 1.1 W3C Recommendation. https://www.w3.org/TR/rdf- schema/.

[7] Resource Description Framework (RDF). https://www.w3.org/RDF/.

[8] SPARQL 1.1 Query Language Specification. https://www.w3.org/TR/sparql11-query/.

[9] SQL. ISO/IEC 9075-1:2016 Information technology – Database languages – SQL – Part 1: Framework (SQL/Framework). https://www.iso.org/standard/63555.html.

[10] Visual RDF. https://graves.cl/visualRDF/.

[11] Vocabulary/Ontology W3C Standard. https://www.w3.org/standards/semanticweb/ontology.

[12] WebVOWL: Web-based Visualization of Ontologies. http://vowl.visualdataweb.org/webvowl.html.

[13] Dean Allemang and James Hendler. *Semantic web for the working ontologist: effective modeling in RDFS and OWL*. Elsevier, 2011.

[14] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[15] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen Sesame. A generic architecture for storing and querying rdf and rdf schema. In *First International Semantic Web Conference (ISWC 2002)*, 2001.

[16] Stuart K Card, Thomas P Moran, and Allen Newell. The psychology of human-computer interaction. hillsdale, nj.: Lawrence rrlbaum associates. *Inc.*, 1983.

[17] Stuart K Card, Thomas P Moran, and Allen Newell. *The psychology of human-computer interaction*. Crc Press, 2018.

[18] Richard Gong and David Kieras. A validation of the goms model methodology in the development of a specialized, commercial software application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 351–357, 1994.

[19] Bonnie E John, Konstantine Prevas, Dario D Salvucci, and Ken Koedinger. Predictive human performance modeling made easy. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 455–462, 2004.

[20] Judith Reitman Olson and Gary M Olson. The growth of cognitive modeling in human-computer interaction since goms. In *Readings in Human–Computer Interaction*, pages 603–625. Elsevier, 1995.

[21] Dennis Quan, David Huynh, and David R Karger. Haystack: A platform for authoring end user semantic web applications. In *International semantic web conference*, pages 738–753. Springer, 2003.

[22] Jef Raskin. *The humane interface: new directions for designing interactive systems*. Addison-Wesley Professional, 2000.

[23] M Schraefel and David Karger. The pathetic fallacy of rdf. In *International workshop on the semantic web and user interaction (SWUI)*, volume 2006, 2006.

[24] MC Schraefel, Daniel A Smith, Alisdair Owens, Alistair Russell, Craig Harris, and Max Wilson. The evolving mspace platform: leveraging the semantic web on the trail of the memex. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 174–183, 2005.

[25] Seema Sundara, Medha Atre, Vladimir Kolovski, Souripriya Das, Zhe Wu, Eugene Inseok Chong, and Jagannathan Srinivasan. Visualizing large-scale rdf data using subsets, summaries, and sampling in oracle. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 1048–1059. IEEE, 2010.