Tampere University

Esa Lempiäinen

VISUALIZATION ANALYSIS AND DESIGN FOR A GEOGRAPHIC INFORMATION VISUALIZATION PRODUCT

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
October 2021

ABSTRACT

Esa Lempiäinen:
Visualization analysis and design for a geographic information visualization product
M.Sc. Thesis
Tampere University
Master's Degree Programme in Human-Technology Interaction
October 2021

Developing information visualization software is a relatively new activity that involves theory of many disciplines. Recent decades have seen big development in theory and practices for software development, as well as design practices of information visualization. However, developing interactive information visualization software has its own range of considerations for design, implementation and evaluation that are not addressed by any other discipline individually. As such the development work of visualizations can benefit from a theoretical base that defines common concepts specifically for developing information visualization software. Such established concepts would enhance the designers' and researchers' ability to think and discuss and thus improve their designs and practices. In her book Visualization Analysis and Design Tamara Munzner has presented a framework that defines a body of over-arching concepts for design and analysis of information visualization tools that spans across the design and implementation process. In this thesis I evaluate an interactive information visualization software product and its development process considering Munzner's framework, while also evaluating Munzner's framework considering how well it lends itself to the task.

Key words and terms: M.Sc. thesis, information visualization, design, analysis, requirement definition, validation, software development.

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

Contents

1. Introduction

Creating effective information visualizations (vis) requires understanding of a wide range of theory on human visual perception. Likewise, creating computer software that performs interactive information visualization involves many more concepts from the fields of computer science and software engineering. Both realms, information visualization and software engineering, have seen rapid development in theoretic foundations and practices in the last decades. On software engineering, Erdogmus et al. [2018] note how the field is yet to even reach the equilibrium of a mature discipline, which would require widely agreed-on ways to solve common problems, since in software engineering the solutions tend to be largely influenced by each practitioner's own experience. On information visualization, Friendly [2006] points out that "most of the innovations in data visualization arose from concrete, often practical goals". Given the ever-present practical goals in both domains driving their development, it is fair to assume that new solutions and theory will keep on coming in both.

The field of information visualization has its roots in statistical graphics, earliest of which were employed by William Playfair in late 1700's [1786]. As such the practices of creating static visualizations of information has a history of some two centuries. Last decades have seen researchers and authors such as Edward Tufte [1985, 1990, 1997], Colin Ware [2013] and Card et al. [1999] establish a theoretical foundation for these practices, first as static graphic representations, and eventually as interactive, computer-mediated visualizations.

The field of software engineering has seen the rise (and decline) of many concepts that aim to serve the software development process. They range from over-arching concepts and process models, such as waterfall and agile development, to technical paradigms for programming and software testing and verification. Developing these concepts has been strongly driven by practical commercial interests of software industry, which aims to produce maximum economic value with minimum resource investment. These concepts concentrate on software engineering at a general level and don't delve in the use cases of the software.

The field of usability and interaction design has introduced concepts like user-centered design that mingle with software development concepts to focus on the fact that usefulness of software is ultimately realized in the interface between the user and the software. These concepts that aim to support development of software with graphical user interfaces (GUIs) do lend themselves to a great degree for creating information visualization software, as interactive information visualizations are graphical user interfaces.

However, developing interactive information visualizations requires special attention to requirements set by the nature of the visualized data and the user's needs for gaining insights from it. The complexity of user needs and domains and the sheer number of possible designs means that general usability principles offer only limited support for vis design. On the other hand, the main body of vis theory focuses largely on the designs alone. That is, it describes how to solve problems, but not how to define those problems, let alone how to evaluate whether the problems were defined and solved adequately. As such, conventional vis theory offers limited support for vis development work. Tamara Munzner [2015] has introduced perhaps the first framework that is tailored for practical vis design work in her book *Visualization Analysis and Design*, bridging the gap between design concepts and vis creation process.

This thesis seeks to build understanding on vis software development by applying Munzner's framework to the development process of a geographic vis software product. The results shed light both on how the vis and its development process can be improved, as well as how Munzner's framework and its associated concepts could be extended to better support dealing with certain characteristics exhibited by the vis software at hand. The analyzed vis software is introduced in chapter 2 and Munzner's framework is introduced in chapter 3. The evaluation of the two is described in chapter 4, followed by discussion in chapter 5 and conclusions in chapter 6.

Due to commercial nature of the vis product, the exact nature of the data and the product's business and technology context are confidential. Because of this it is not possible to describe the product in its actual domain language in this thesis. Also, all identifying information has been redacted from images that show the vis product's UI.

2. The information visualization software product

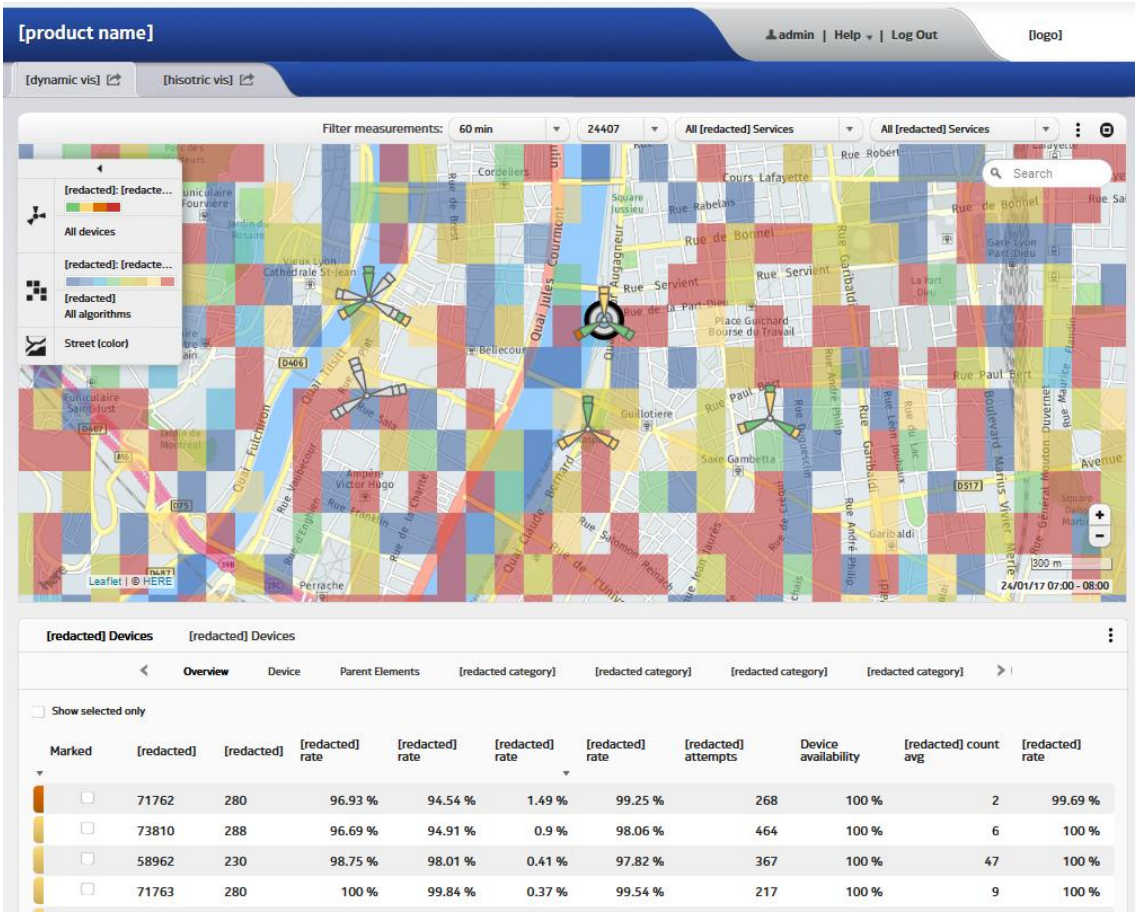

Figure 1. The real-time data vis UI of the vis product. Information that reveals the technological domain and product identity has been redacted.

This thesis investigates one user interface (UI) of an information visualization software product that is used for monitoring and troubleshooting the behavior of a system that continuously processes large volumes of geographic data. This user interface displays the data in a geographic visualization (see Figure 1). The product contains also other user interfaces that display the data in different ways, but these GUIs are not in the scope of this thesis.

The product is made specifically for a certain technological domain and is deeply tied to the data of that domain. The monitored system processes data continuously, and the product gathers and aggregates this data in effective real time, allowing real time monitoring of the system. The product also stores the data in a database, allowing querying historical data with the resolution down to individual raw data items. The users of the product are engineers responsible for planning, management and optimization of the monitored system. The product's real-time monitoring allows users to continuously verify the desired operation of the system, and to locate any anomalies in the system behavior as soon as they appear.

The geographic vis UI that this thesis focuses on provides capability to monitor and troubleshoot the system's behavior based on geographic information. Following the two-fold data of the existing product (aggregated real-time and raw historical), there are two kinds of geographic visualizations: aggregated real-time visualization and visualization of raw historic data per data item.

The vis UI is implemented as a single page application for web browsers with web technologies (JavaScript, HTML, CSS). This is worth noting as certain design decisions leverage web browser functionality.

In this chapter, aspects of the vis product are introduced in general terms to provide an overview of the product. This provides context for the analysis in later chapters.

2.1 Real-time vis

In the real-time visualization (see Figure 1) the data items gathered by the product are aggregated into geographic brackets denoted by *geohashes* [Niemeyer, 2008]. These are called *geotiles*. Each geotile has data attributes assigned to it, such as numbers of different types of data items, ratios between certain data item types and statistical figures like averages and medians of data item measurements. These attributes are encoded with colors so that as user defines one geotile attribute to be displayed, each geotile receives a color that denotes its value for that attribute. The result is a colormap where geographic map is overlaid with colored geotiles.

Another graphical element on the map are markers for device items associated with the data. Each of these devices has associated data that is a collection of similar attributes as geotiles. These attributes describe aggregated data associated with the devices, analogous to how geotile attributes describe aggregated data associated with geographical areas. As such, the device items are also colored to indicate their attribute values.

The coloring of geotiles and device items is segmented. That is, all values that fall within certain value range receive the same color. These value ranges can be edited by user. The functionality is described in detail in chapter 4 where the coloring is evaluated.

The data can be filtered in several ways. The period from which data is shown can be chosen between one minute and one week, and data can be chosen from different data sets and filtered by certain attributes. These selections and filters affect all data. Additionally, geotile data and devices can be filtered separately.

The aggregation uses the product's capability of continuously aggregating large amounts of data in a very performant way. This has implications to the design, which are explained in chapter 4.

2.2 Historic vis

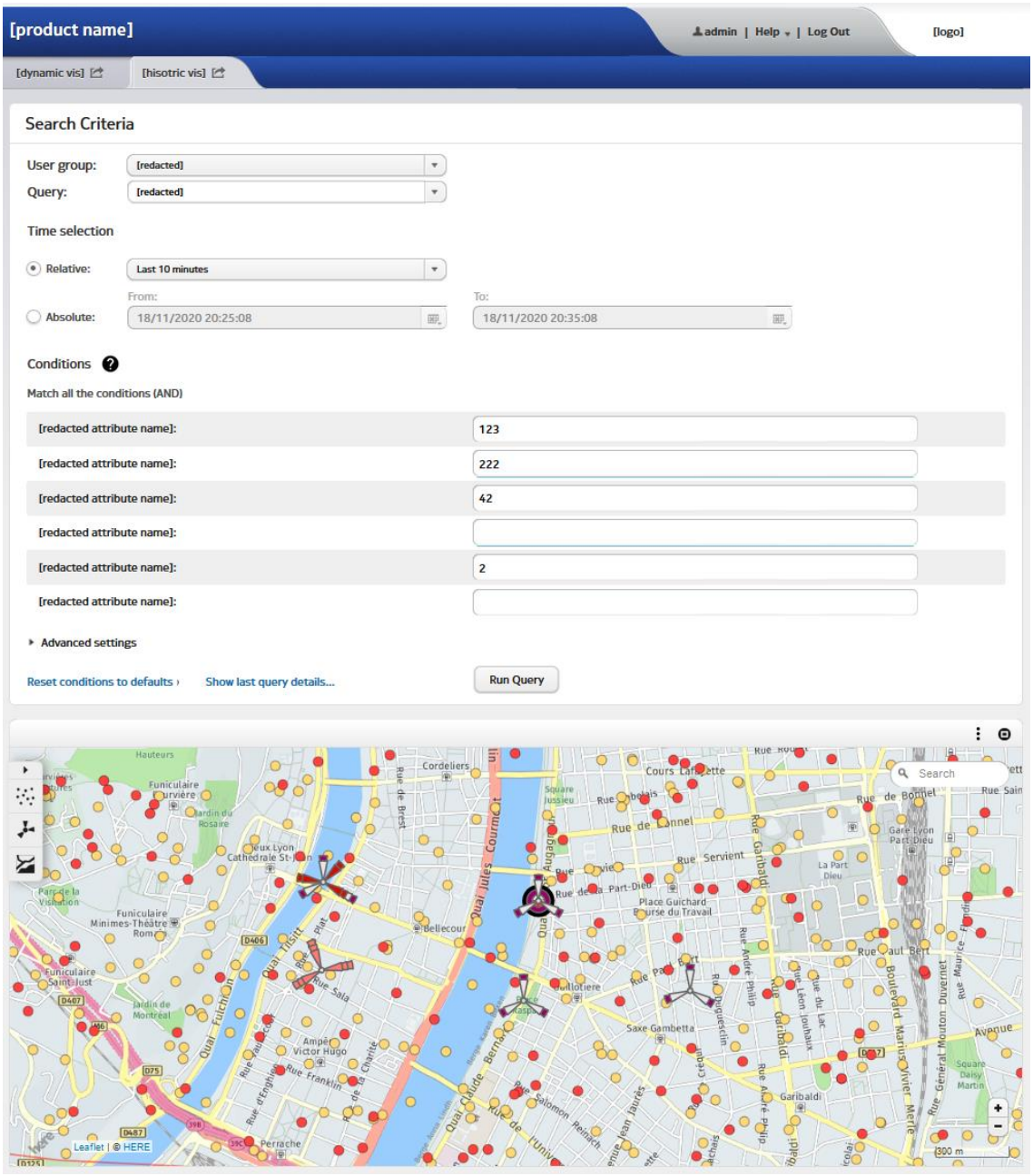

Figure 2. The historic data vis UI of the vis product. Information that reveals the technological domain and product identity has been redacted.

The vis for historic data (see Figure 2) provides functionality for querying data from database and visualizing query results on map. As such it differs from real-time vis in many ways, but also has many similarities.

The data in this case is not aggregated. Instead, it is individual data items, each with its own geographic location. These are visualized on map as circular markers. The markers can be colored by data item attributes in similar fashion as geotiles in the real-time vis.

This vis also has the same markers for device items on map as the real-time vis. The functionality about them is the same as in real-time vis, save for coloring by attributes. Since in this historic vis all data is individual data items on map, there is no aggregate data for devices either, and thus the device items cannot be colored by such data. Instead, here the device items can be colored with individual colors that signify their relation to the data items. That is, the data items have attributes that refer to the devices, so when results for a query are received, each device for which there is a reference in the data is assigned a unique color, and the data points are given the same color. This way the user can see what devices are related to the data on the map.

2.3 Development process

The vis product was developed for a corporate product portfolio. The path from requirements to design and to implementation involved two organization branches: product management (PM) and research & development (R&D). PM defined requirements for sellable features and the R&D specialists designed and implemented functionality to fulfill the requirements.

In other words, PM acted as a client that ordered software functionality from R&D. As such it was ultimately PM's responsibility to define features, but in practice the feature definition involved much collaboration between PM, designers from R&D and other stakeholders, e.g. other in-house specialists, customers and sales representatives. That is, while the final say on which features are picked up was with PM, initiatives for new features came from many sources [Sillanpää, 2019].

PM was the authority on business priorities by which they decided and prioritized features. Conversely, it was PM's responsibility to decide when the implementation presented by R&D fulfilled the agreed feature. However, defining features itself required knowledge on user needs. There were some problems with impacts on design and development process that stemmed from this division of responsibility between PM and R&D designers. The division of responsibility and its impacts is discussed considering the Munzner's nested model in chapter 4.

3. Munzner's framework: the four nested levels of vis design

In her book *Visualization Analysis and Design* Tamara Munzner presents a range of concepts that pertain to creation of information visualizations and a framework for reasoning about this creation process.

The core of Munzner's framework is a nested model of visualization design and validation [2015, p. 67]. It formulates the creation of information visualizations as stages of different abstraction levels from abstract user needs to concrete implementation as shown in Figure 3.

The rationale for this separation is to reason about how the requirements of higher levels impact lower (or inner) levels of design. This way it is possible to verify that the design motivations from the user needs carry throughout the whole design and implementation process, resulting in visualizations that satisfy the user needs. To support this reasoning Munzner presents a range of concepts for each of the levels, save for the fourth level, algorithm implementation. She does so by using a holistic vocabulary to disambiguate concepts, as there is a considerable number of concepts at each stage. She acknowledges, however, that out in the wild there is no clear one to one mapping between the concepts and words that can be used to label them. As such, the designers that use this framework must match the concepts to the vocabulary used in their own context.

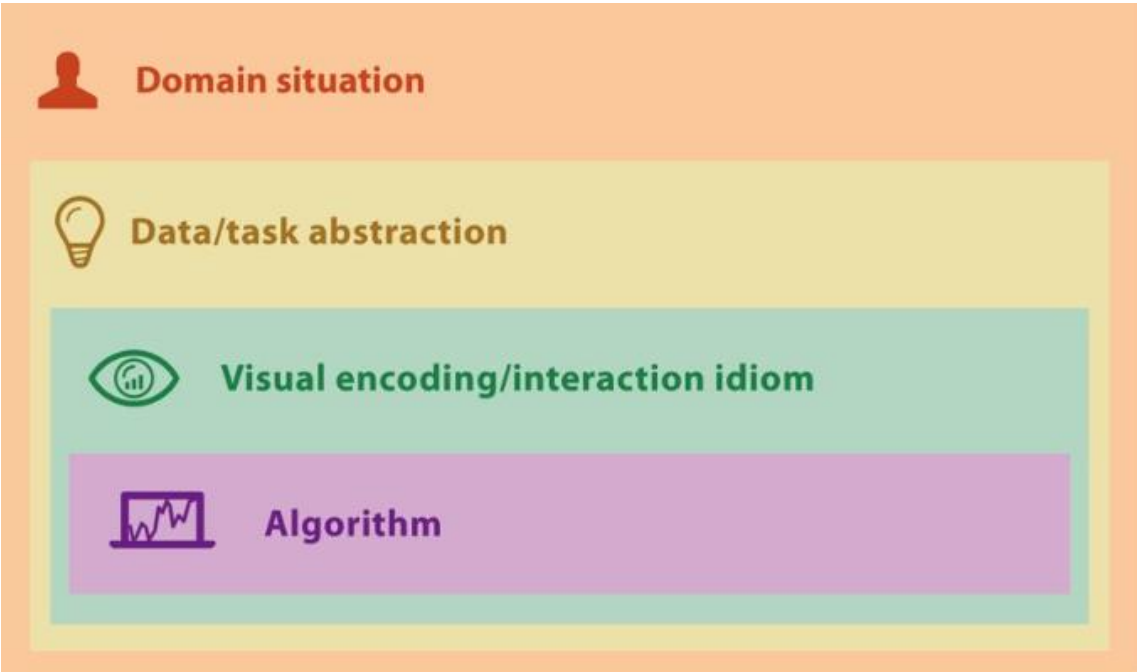

Figure 3. "The four nested levels of vis design" [Munzner, 2015, p. 68]

At **domain situation** level the designers work to identify who are the users, what is the data available and what interests do the users have of the data. This identification work produces information on which designers base their design decisions. As the outcome of the identification work the designers should know what questions the users have of the data, what actions they need to perform on the data and the exact nature of the data. In terms of the geographic vis this thesis focuses on, the users are system planning and management specialists whose interest can be formulated on a high level as such: "I want to identify any devices that are causing the system perform below expected level of quality so that I can take corrective action on those devices". Munzner notes that developing clear understanding of the domain situation requirements is tricky, as "while it might seem obvious to you that it would be a good idea to understand requirements, it's a common pitfall for designers to cut corners by making assumptions rather than actually engaging with any target users" [2015, p. 69].

At **data/task abstraction** level the designers refine descriptions for user tasks and the involved data based on the information gained from domain situation level. That is, designers ought to look past any domain-specific vocabulary to identify the needed user's tasks in plain terms on which they can apply vis for solutions. Such plain tasks may be e.g. finding outliers or identifying patterns. On the other hand, Munzner notes that whereas user tasks are *identified*, the data for a vis is *designed* [2015, p. 70]. That is, as vis always requires processing the data so that it can be displayed, the vis designers are in control how the data is processed for presentation. Munzner points out that while sometimes the raw data might be usable for the vis as is, often the best match for supporting the user's tasks requires transforming the data one way or another [2015, p. 71]. In the context of this thesis, the vis product transforms the data by aggregating it over geographic regions, devices and time periods.

At **visual encoding/interaction idiom** level the designers decide how to present the data (visual encoding idiom) that was designed and what kind of interactivity to provide to let the users carry out the tasks that were identified at data/task abstraction level. Munzner uses the term *idiom* to refer to distinct possible approaches for visual encoding and interactions. There are usually multiple ways (idioms) to present any given task or data. Munzner does not state it directly, but visual encoding/interaction idiom level appears to be what may sometimes naively understood as "design", as that is where the designer makes the concrete decisions on how the vis is to be implemented by choosing from available alternatives. Munzner refers to the range of possible idioms as **design space**, from which designers pick idioms best suited for given data and task, or from which they derive inspiration for new idioms. Using this metaphor Munzner illustrates the importance of being aware of a wide range of idioms and being ready to evaluate which of them is the best fit for the situation at hand (see Figure 4). There often are

multiple idioms that could be applied to a given situation, so to come up with a design that adequately satisfies given tasks, the designer must be able to rule out the ill-suited idioms and to pick an adequate one. In her book *Visualization Analysis and Design* she presents a large number of concepts for analyzing data/task abstractions so that designers could make well founded choices on idioms.

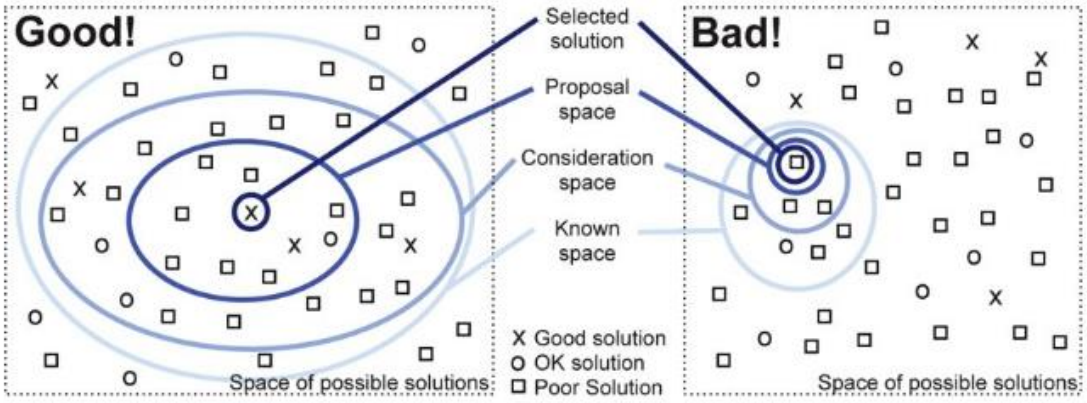

Figure 4. “A search space metaphor for vis design.” [Munzner, 2015, p. 13]

At **algorithm** level the visual encodings, data transformations and interactivity that were designed at previous levels are implemented with computer programming. The designs guide the implementation, but implementations also facilitate designs: Munzner states as an example that “a design that requires something to change dynamically when the user moves the mouse may not be feasible if computing that (sic.) would take minutes or hours instead of a fraction of a second” [2015, p. 73]. Ultimately the goal at algorithm level is to design the used algorithms so that the best possible visual encoding and interaction idioms can be realized. This is often not a given, as required data transformations and interactions can be computationally quite expensive. Munzner goes on to note that “clever algorithm design could save the day if you come up with a way to precompute data that supports a fast enough response” [2015, p. 73]. This is very much the case with the vis product that this thesis investigates: large amounts of data are made available for interactive visual representation by highly performant preaggregation.

Figure 5. "The four nested levels of vis design have different threats to validity at each level." [Munzner, 2015, p. 75]

Each of the design levels has certain **threats to validity** as Munzner puts it [2015, p. 75]. These refer to different ways that the design work can go wrong at different levels, resulting in a vis with little usefulness. These threats are illustrated on a high level in Figure 5. Munzner presents a range of means for tackling each of these threats, both before making decisions at each of the levels, and also after implementing the vis according to the decisions.

I go to into detail in Munzner's model, data/task abstraction concepts and validity threats in context of the vis product in Chapter 4 where I apply them on the vis product and its development process, and conversely evaluate how the model and concepts lend themselves to this task.

4. Two-way evaluation

The relationship between the vis product and Munzner's framework is explored in this chapter.

Chapter 4.1. focuses on providing overview for how the concepts of Munzner's framework appear in the vis product.

Chapter 4.2. contains evaluation of the vis product's design process based on methods Munzner proposes for ensuring soundness of design decisions.

Chapter 4.3. delves deeper in the issues identified in chapters 4.1 and 4.2.

Chapter 4.4. explores some interesting topics and gaps in Munzner's concepts.

4.1 Overview

In this chapter I look at an overview on how different aspects of the vis product and Munzner's model correspond to each other. First, I investigate how the whole design process maps to the nested design levels, and then I investigate how Munzner's semantics for data abstractions (*What*), task abstractions (*Why*) and design choices (*How*) play out in the vis product's design. This overview provides context for more in-depth look on potential issues in the design and gaps in Munzner's framework that are discussed in later chapters.

4.1.1 Vis product's design process vs. four levels of design

The process of feature specification and design that was explained in chapter 2 follows the four levels of design described by Munzner. The features correspond to *domain situation* level: they describe domain-specific needs in domain language. It is from these descriptions that designers derive their design motivations, which correspond to *data/task abstraction* and *visual encoding and interaction idiom* levels.

Features (*domain situation* level) are responsibility of PM, while designs (*data/task abstraction* and *visual encoding / interaction idiom* levels) are responsibility of R&D. This division presents some challenges in the design work, as the organizational divide falls along what should ideally be a continuum from domain descriptions to implementations.

As Munzner [2015, p. 68] notes, poor decisions at higher levels will restrict successful design on lower levels. If *domain situation* is not defined with enough precision, or if it is defined in a way that does not correspond to actual user needs, the decisions made at lower levels cannot save the whole design: the result may be a perfect system for solving problems that nobody has. Because of this the designers should be able to formulate as precise *domain situations* as possible at the start of the design work, and to return to refining the *domain situations* if the later design stages require it.

4.1.2 What: data abstractions

The *data abstractions* of the vis product focus on making large volumes of data fathomable to the user. The raw data is of *table* type. That is, it consists of data items, each of which has several *attributes*. Some of these *attributes* are positions that pinpoint the data item to a geographic location. This makes the data also *spatial* in nature. Raw data items can be seen in Figure 6.

The data is continuously accumulated, that is, it is *dynamic*. The data is also stored in databases, becoming *static*.

In the real-time vis, the *dynamic* data is *spatially aggregated* into a geographic *grid*: the *geotiles*. This is to allow user to see overview of attribute values over geographic areas (see Figure 7).

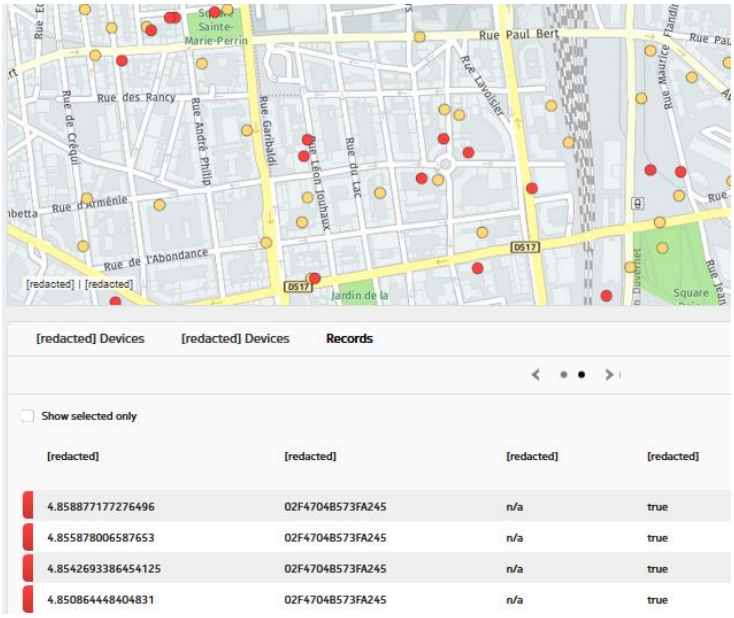

Figure 6. Individual geographic data items displayed on map and in table in the historic vis.

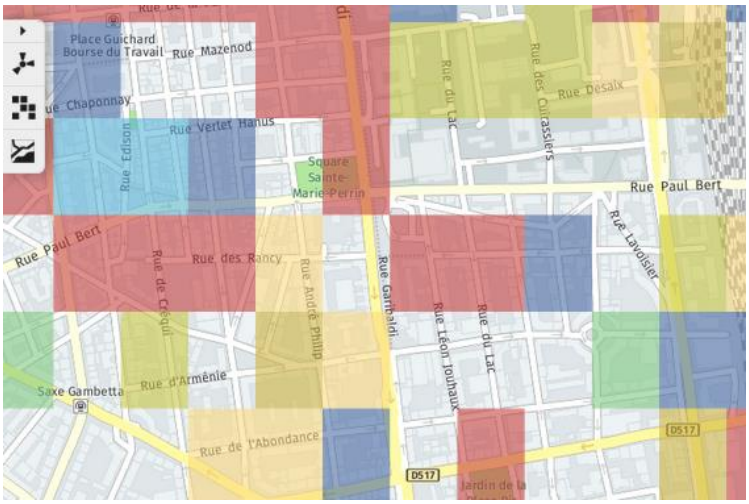

Figure 7. Geotiles aggregated from geographic data items in the real-time vis.

Another set of data is the devices. They are likewise *table* items with *attributes*, which contain coordinates that pinpoint them to a geographic location. They are, however, only *static* and are not *spatially aggregated*. Instead, each item of the dynamic dataset relates to some item in the device data. With this information new attributes are derived for the device data items by *aggregating* attributes from the dynamic data set also per device items (see Figure 8).

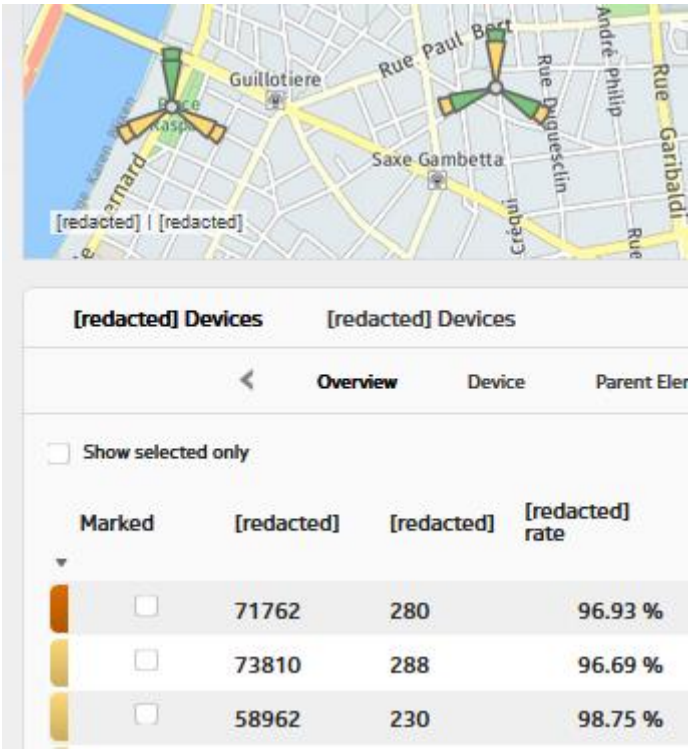

Figure 8. Device items on map and in table in the real-time vis. The cone-shaped markers encode devices by location and direction. The coloring is based on aggregated data for each device.

Data aggregation makes use of the parent product's real-time aggregation capability [Sillanpää and Koivuniemi, 2019]. This capability has been proven to be performant and stable in the parent product's history. This is a significant argument in its favor in software development context: it saves a large amount of work to use an existing, proven functionality instead of developing new functionality. However, there is question of existing solution's fit for new requirements: while the existing solution is powerful in its real-time performance, this performance comes at cost for data amount. The existing solution works in-memory, that is, the aggregated data is not stored on disk. As such there is a size limitation in the aggregated data. Because of this the aggregated data is only available for a limited, moving time window. This naturally limits the potential *domain situations* that the aggregated data could otherwise serve if it were available for a longer history. This is one motivation for the historic vis, which uses the raw data stored on disk.

4.1.3 Why: task abstractions

The *task abstractions* of the vis product revolve around *discovering* phenomena in the data and *identifying* geographic locations and devices associated with the data. That is, users *browse* and *explore* to gain insight on *outliers* and *features*, and *lookup* regions and devices with interesting properties based on external input.

The users are interested in monitoring data attribute values over geographic areas. That is, they want to discover unexpected phenomena from the data and *verify* that the data is as expected. They want to be able to act immediately if they discover unexpected phenomena.

Upon discovering interesting phenomena, or upon being externally informed of interesting phenomena, the user wants to identify the involved regions and devices and *explore* their relationships to other regions or devices. Ultimately, the users are interested in identifying devices that are involved with any problematic data, so that they can take action to correct the situation.

4.1.4 How: design choices

The *aggregations* described earlier are instances of *reducing items and attributes*. As explained by Munzner [2015, p. 305], *aggregation* has the tradeoff of not conveying all information from the aggregated data. Additionally, in this situation the availability of aggregated data is limited in time duration.

These limitations are the motivation for the historic vis. The historic vis enables querying of raw unaggregated data from a much longer time range than what aggregated data is available. To find data items of interest, the queries are performed with criteria given by user. This is effectively *filtering*, an alternative way of *reducing* data. Unlike *aggregating*, *filtering* does convey all information from the data that remains after *filtering*. Its tradeoff is that as data that is *filtered* out is not visible, user may have trouble staying aware of its existence [Munzner 2015, p. 300]. Providing both means of *aggregation* provides refief both tradeoffs. In practice this means that users can *search* an interesting geographic area from the *dynamic* aggregated data with the real-time vis, and then use this information to *derive* more detailed information by *browsing* the *static* unaggregated data for the same area in the historic vis. This is supported by allowing users to launch the historic vis from the real-time vis so that the user can use the context identified in the dynamic vis as the starting point for their activities with the historic vis.

In next chapters I will explain the vis idioms and the main interactions offered by the vis.

4.1.4.1 *Filtering data*

Before introducing the vis idioms, it is necessary to explain how the user can filter data globally across the vis tool's view.

As described earlier, the real-time data items are aggregated by time period. After that they are further aggregated by other factors such as the associated device, but common time period is something that ties together all aggregated data.

At the top of the map area there are filters that apply to all aggregated data. The first of these is the time period selection (see the leftmost pulldown control with label “60 min” in Figure 9) As the user changes this selection, new data is loaded into the vis from the selected real-time aggregation period. This filtering operation is computationally inexpensive, since data has been pre-aggregated, and changing the filter selection merely defines what data is retrieved from the in-memory storage.

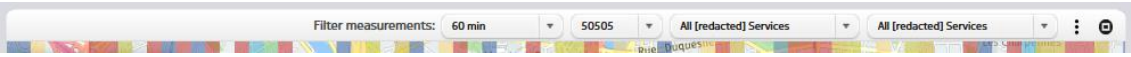

Figure 9. Filter components at the top of the map.

The aggregated data storage is continuously updated by the underlying data collection functionality as new data is collected from the monitored system. To reflect this, the data in the real-time vis is updated periodically to display the most recent aggregated data.

Other top-level filters leverage other pre-aggregations (see other pulldown controls in Figure 9) that represent certain top-level categorical attributes of the data.

In addition to the top-level filters that serve as selectors for different aggregated data sets, there are also filters for geotile and device data (see Figure 10 and Figure 11). These filters allow selecting which data attribute (“KPI”, known performance identifier) to display for geotiles and devices. These filters for geotile and device vis are discussed in detail in next chapters

One of the geotile filters warrants special attention: the location algorithm filter (see the bottom of Figure 10). It filters data based on an attribute that indicates *certainty* of the data item location. That is, the location of data items may be defined by the monitored system in several ways. These methods have varying accuracy: one method can pinpoint data item location quite accurately, while others rely on algorithmically deriving the information from a number of other available attributes. The less accurate location methods are necessary, since not all data can be located with the most precise method due to technological limitations.

Because of this varying *certainty* of data item location, designers of the tool have considered it necessary to enable user to filter the data based on this *certainty*. That is, one of the filter pulldowns allows the user to select which location methods to exclude: they can see all data (no location methods excluded), or only the most precise data (all but the most accurate location method excluded) or anything in between. This is made possible by aggregating the data also by the used location method.

Figure 10. The geotile data filters.

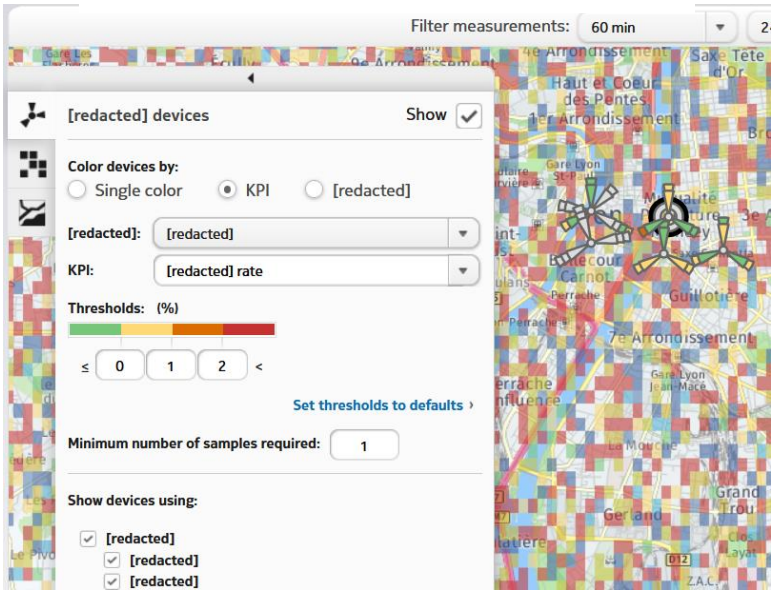

Figure 11. The device data filters.

4.1.4.2 *Geotiles*

Figure 12. Geotiles on the map.

Idiom	Geotiles
What: Data	Spatial, dynamic
What: Derived	The area under study may have been found with another monitoring tool or process
Why: Tasks	Lookup area of interest, explore to identify areas of interest, discover devices of interest
How: Reduce	Preaggregation into geographic grid, filter by attributes
How: Encode	Use given geographic context, map to color hue
How: Manipulate	Navigate map, select individual geotiles

Table 1. Geotile idiom taxonomy.

The *geotile* idiom is the most significant idiom of the vis product, as it visualizes largest quantity of data with the largest number of pixels of all the vis tool's idioms.

As described in chapter 2.1, geotiles are a way to split Earth's surface into rectangular areas each denoted by a *geohash*. Geohash is a geographic location encoding method presented by Gustavo Niemeyer [2008] that uses strings of alphanumeric characters to denote rectangular geographic areas so that each added character creates a string that denotes a sub-area within the area that is denoted by the preceding string. That is, geohash "udby5" denotes an area of about 4.89 km × 4.89 km around coordinates 61.5305, 23.7088, while geohash "udby51" denotes an area of about 1.22 km × 0.61 km around coordinates 61.5317, 23.692. As such, a geohash is an identifier that references a geographic area. While it obfuscates the actual coordinates, the character string is arguably much easier for humans to memorize than numeric coordinates, making it better

suited for assigning human-readable identifiers to geographic areas. Technological tools can then decipher the geohashes into actual geographic locations.

The concatenating, nested quality of geohashes is not used by the vis tool however, as each geotile is 7 characters long, referencing a 153 by 153 meter area. Nevertheless, the geotile-based approach to data aggregation was chosen to support possible future functionality where data could be aggregated at different geographic granularities with geotiles of different sizes.

Data is aggregated into geotiles as follows: per each time period (e.g. 15 minutes), all data points of the raw data that fall to some geotile's area are aggregated so that for some attributes, average and median are calculated, and for some categorical attributes value sums are calculated.

Aggregated values are *encoded* with *color hue*. Their value range is split into 5-8 discrete bins, each of which is assigned a hue (see Figure 12). The used hues depend on type of the encoded attribute. Some attributes are *counts*, that is, their values range from zero to theoretical infinity. Such attributes are assigned hues that represent a *heatmap*. Smallest values are encoded with blue hues, then with green, followed by yellow and orange, ending with a red hue. These values are aggregated simply by counting the sums of certain *categorical* attributes in the data.

Other attributes have values that are calculated with certain domain formulas, and their ranges are defined by these formulas. Depending on variable, one end of the range signifies good performance, while the other end signifies bad performance. Such variables are assigned hues that represent their *valence* (good/bad). These hues range from green (good) through yellow and orange to red (bad) (see Figure 11 for the same encoding for device items). Such values are aggregated for example by calculating median and average for certain attributes that have been aggregated in the same geotile in the same time period.

The user selects the encoded attribute for geotiles from the pulldown control on the map edge (see Figure 10). In the same element there are also other controls for configuring the geotile vis for the current user session. These are the controls for threshold values used for dividing the data into discrete bins for the vis, and so-called *guard value* for statistical KPI attributes.

The use of discrete value bins for encoding with *hue* as encoding method was chosen because of an identified domain need. It was identified that users are mostly interested in identifying values that fall below certain thresholds, as opposed identifying the differences in geotile values with maximum accuracy. As described earlier, some attributes range in *valence* from "good" to "bad". For such attributes there are domain-specific thresholds, by which the threshold of "bad" values can be defined. This threshold information is used in geotile vis to define the default thresholds for the value bins, and

values in the lowest bin are given red hue. The rationale for this has been that the red hue will stand out in the geotile colormap, allowing the user to quickly see the interesting "bad" values.

The so-called *guard values* are a response to an identified need to deal with the potentially misleading effect that calculated attributes have with the geotile idiom. As the statistical aggregations such as average are calculated from what data items are available in the aggregated set, they are impacted by the total number of data items in the set. That is, if there is a small number of items, even a single item with anomalous value may have a great impact on the calculated average. With median a handful of anomalous values will not have as big an impact, but even then, there is a certain risk of a handful of anomalous values tilting the aggregated value. The vis risk comes from the fact that **each geotile occupies the same number of pixels, but they represent different amounts of data**. As such a geotile that was aggregated from 10 data items has the same visual weight as a geotile that was aggregated from 1000 items. A geotile with only 10 items usually is not as interesting to the user as a geotile with 1000 items. The *guard values* are a filter that filters out geotiles that have less aggregated data items than the number of the *guard value*. This way the *guard values* "guard" the user from being misled by geotiles that encode an insignificant number of items so that they can concentrate on more significant geotiles. As there is no certain definition of "significant" number of items per geotile, the user can set the *guard values* themselves, although the vis also has default values configured for them. Figure 13 and Figure 14 demonstrate the guard value behavior. In Figure 13 the guard value is 1 meaning that all geotiles are shown. In Figure 14 there is guard value 50 which causes many geotiles to be hidden.

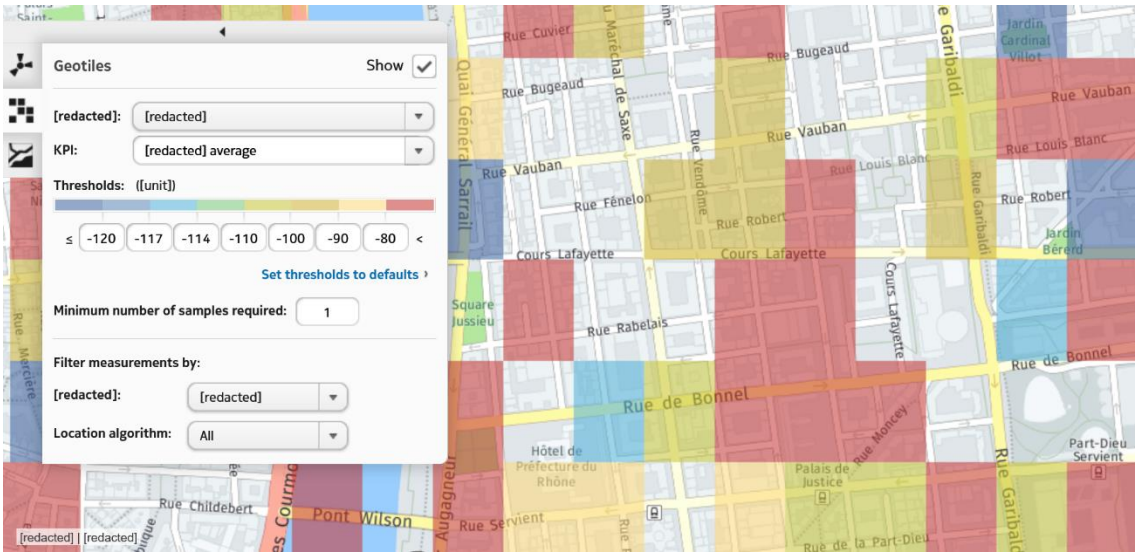

Figure 13. The effect of guard value "1" on geotile visibility. All available geotiles are shown.

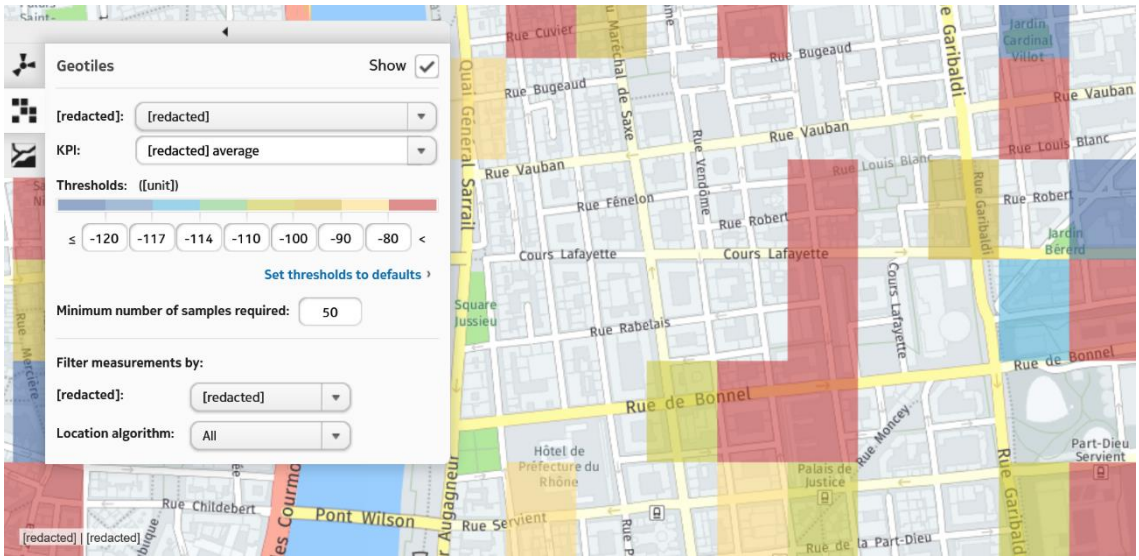

Figure 14. The effect of guard value "50" on geotile visibility. Geotiles with fewer than 50 aggregated data items are hidden.

Geotiles make up a *colormap*, which is a powerful vis idiom but comes with certain risks that may harm the effectiveness of the vis, especially when using *hue* encoding as with geotiles. Munzner [2015, p. 234] notes that "segmented rainbows could also be used for ordered data; while not ideal, at least the perceptual nonlinearity problem is solved because the colormap range is explicitly discretized into bins." She points out that "using a segmented colormap on quantitative data is equivalent to transforming the datatype from quantitative to ordered" and that "this choice is most legitimate when task-driven semantics can be used to guide the segmentation into bins." This is exactly the rationale behind discrete geotile colormap: it was deemed important to make especially the "bad" values stand out as their own category.

Another risk with geotiles is how they are affected by *modifiable areal unit problem* (MAUP). Wong [2004] defines geographical MAUP to be caused by the fact that "boundaries of many geographical units are often demarcated artificially, and thus can be changed" and "when data are gathered according to different boundary definitions, different data sets are generated. Analyzing these data sets will likely provide inconsistent results". The risks about the hue colormap and MAUP and how they have been considered when designing the geotile vis idiom are discussed further in later in this chapter.

As geotiles occupy the largest number of pixels in the vis and they are visible whenever the user pans the map to an area with data, they are starting point for many workflows. When the user has selected to see a *KPI* attribute with encoded valence, they may see some geographic areas where geotile *color hue* indicates poor KPI values. Alternatively, the user may select to see a *count* attribute, in which case they may see areas where there is an unexpected amount activity.

The user can *select* a geotile to see window with a full presentation of its associated data (see Figure 15). From this window they can browse through all attributes of the geotile, in which case they may notice other values of interest. They can then go to top level filtering and select to see a different attribute for the geotiles, which may lead to new discoveries. In the window the user can also see a list of devices that are associated with the data in the geotile. They can then *select* devices from this list to open a similar window for the devices.

When the user selects a geotile, they will also see lines on map that connect that geotile to all devices that are associated with it (see Figure 16). This way the user may *discover* if there are any unexpected devices associated with the selected geotile.

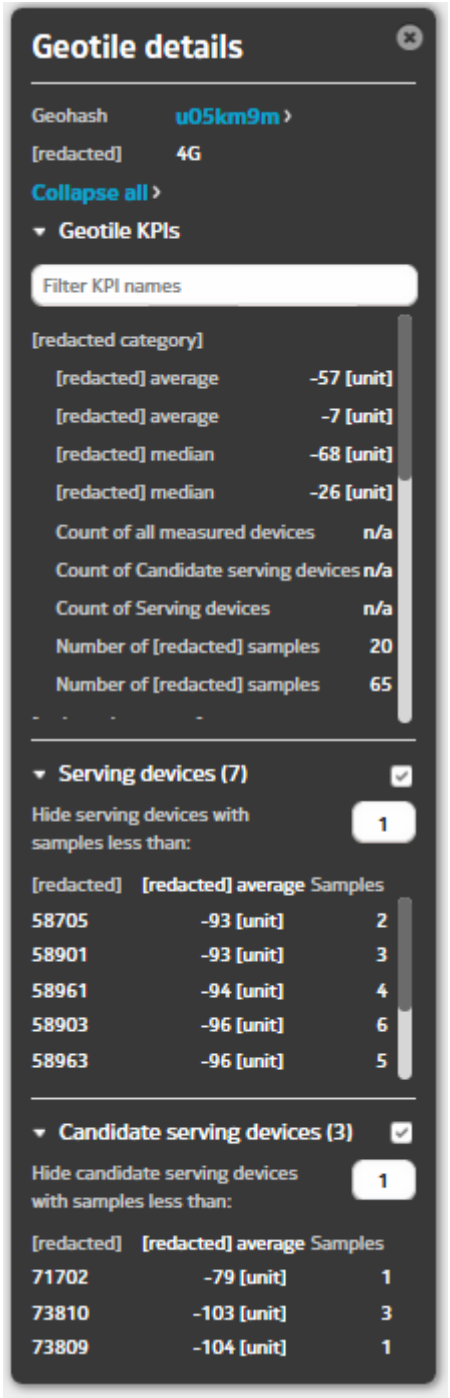

Figure 15. Window for full data of the selected geotile.

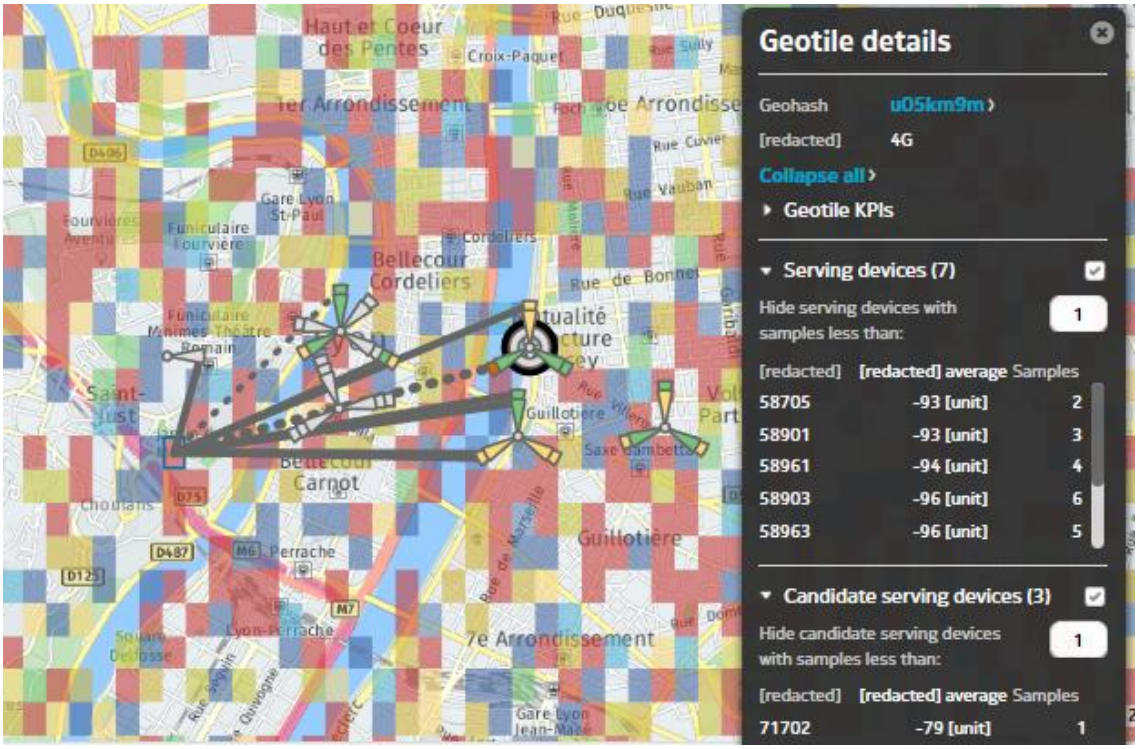

Figure 16. Lines that connect the selected geotile to its associated devices.

4.1.4.3 *Device items*

Figure 17. Device items on the map.

Idiom	Device items
What: Data	Tabular list of device items, spatial location, part static device attributes, part dynamic attributes
Why: Tasks	Locate device with known identity, explore device items at a location, identify associated devices, derive devices associated with geotiles, derive geotiles associated with devices, find extreme values by sorting table
How: Encode	Use given geographic context, shape to indicate orientation, map attributes to color hue
How: Order	Sort table to find extreme values
How: Reduce	Filter by attribute values
How: Facet	Juxtapose views with geographic location and position in ordered table

Table 2. Device item idiom taxonomy.

In addition to geotiles, the map view shows items that encode devices that are involved with the data (see Figure 17). These items encode geographic location of the devices as their location on the map, and the physical direction of the devices with their shape. This information on device locations and directions is *static* and comes from a different source than the vis tool's *dynamic* geographic data.

As for the dynamic, aggregated data, in addition to aggregation into geotiles, the data is also aggregated per device. That is, same way that certain attributes are *counted,* and some are aggregated as KPIs with statistical signifiers like medians and averages per geotile, certain attributes are aggregated per device. This is possible because each raw data item has, in addition attributes for geographic location, also attributes that identify a device that is associated with that data item.

This aggregated, dynamic data is encoded the same way as with geotiles: with color *hue*. As with geotiles, the value range for the attributes is split into discrete segments, and each segment is assigned a *hue*. Also similarly with geotiles, the utilized hues depend on the kind of encoded attribute: some range conceptually as *quantities* from "low" to "high" and are encoded with hues from blue to red, while others range as *valence* from "bad" to "good" and are encoded with hues from red to green.

As with geotiles, the user can select the displayed device attribute from the pulldown on the map edge (see Figure 18). The same location contains also controls for the discrete value bin thresholds and *guard values*. These controls function for device data the same way as similar controls do with geotiles. One difference to note is that where geotiles that are filtered out with guard values are hidden completely, device items that are filtered out remain visible, but are given a grey "no data" color (see Figure 17). Geotiles are arbitrary aggregations of data and as such they exist only when there is data to show, whereas device items represent actual devices that exist even if there are no dynamic data values to display for them.

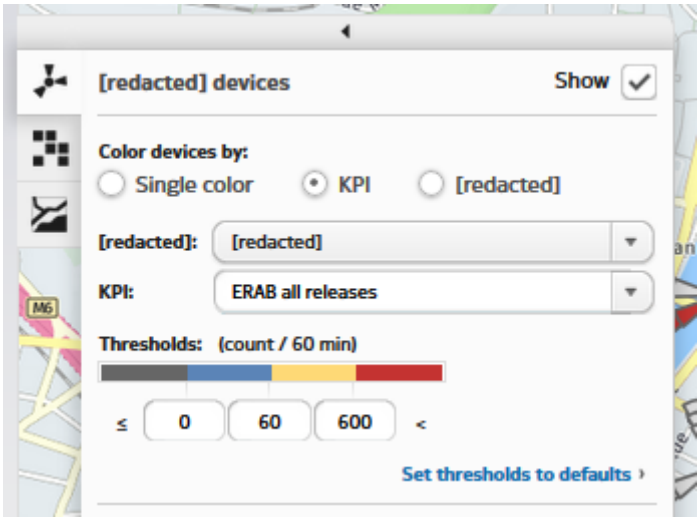

Figure 18. The device attribute selectors and device color bin thresholds.

In addition to controls for encoding the aggregated data, the device items can be made to show also certain *static* device data (see Figure 19). This domain specific data represents a certain *categorical* attribute of the devices. This attribute is encoded with a set of *hues* that were chosen so that they won't be confused with aggregation-encoding *hues*. The device items can also be filtered out based on these values with the checkbox controls in the control pane.

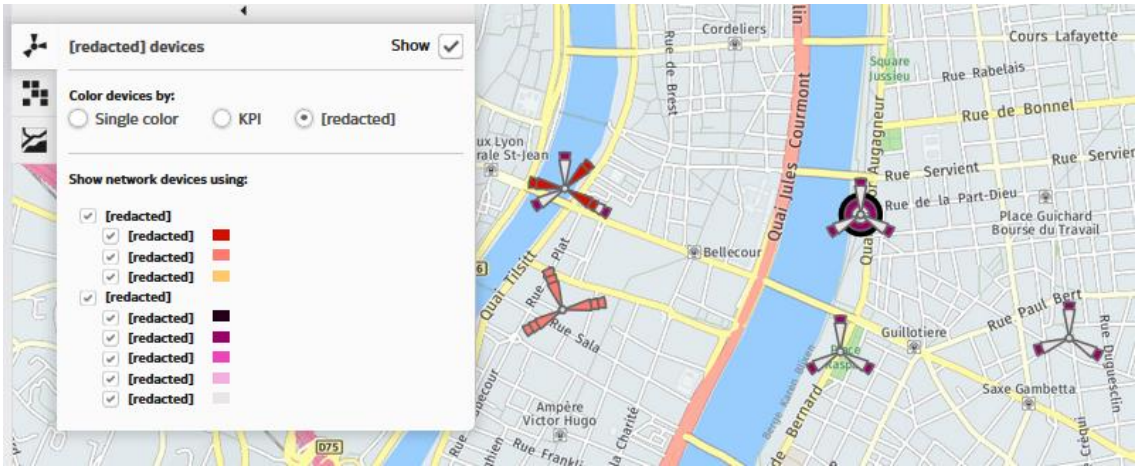

Figure 19. Device coloring by a categorical attribute.

Also similarly as with geotiles, the user can select a device item to see a window that contains a full presentation of that device's data in the selected time period (see Figure 20). The selected device item is highlighted with a pin icon on the map (see Figure 21).

The selected device receives special treatment in the map's *semantic zoom*. The device items are hidden from view when user zooms the map far enough. This is to prevent occlusion. Geotiles have fixed geographic size, and they get smaller or larger depending on the zoom level, up to the point where one geotile may occupy only several pixels. On the other hand, device items cannot be scaled this way: due to their shape, visual borders and potential close proximity to other device items, they would lose all useful information if scaled to very small sizes. Because of this the device item's size is scaled only slightly with the zoom level. This in turn would cause severe occlusion on the map on higher zoom levels if all the device items would remain visible while geotiles become miniscule. To prevent this occlusion, the device items are hidden from view at higher zoom levels, except for the selected device item. This makes the zoom a semantic zoom vs. ordinary graphical zoom: the effect of the zoom is adjusted so that the as much of the visual representation's meaning is carried over on each zoom level.

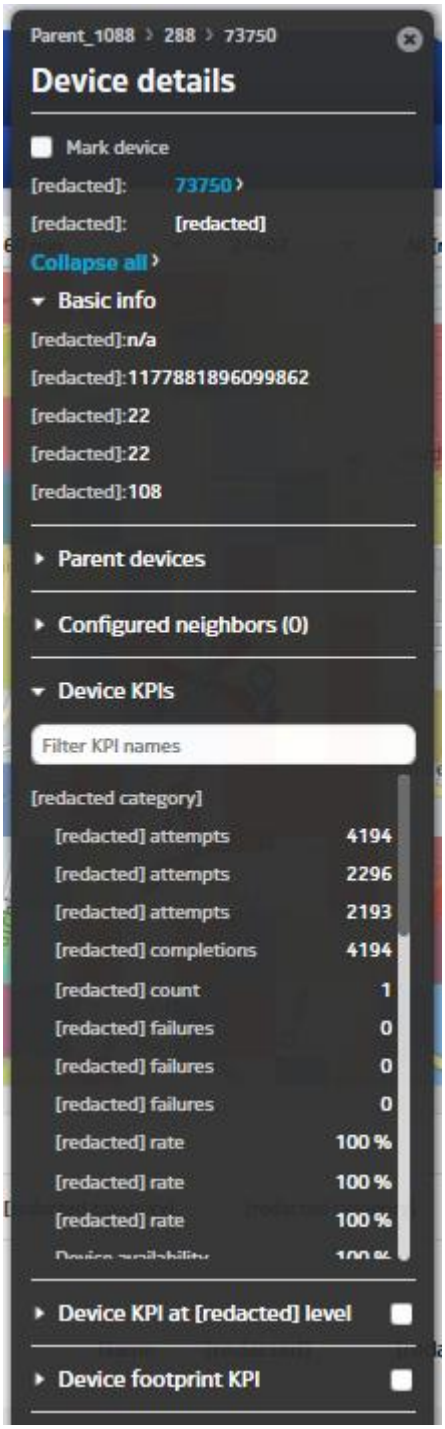

Figure 20. Window for full data of the selected device.

From the device detail window, the user can *filter* geotiles so that only geotiles associated with the selected device are visible (see Figure 21). When *filtered* this way, the user will see data that has been aggregated *both* by geotile and by device. This way the user can see if data per geotiles in certain area and per certain device contains unexpected values, or if there is data in unexpected location associated with the selected device.

When user sees this kind of per-device filtering for geotiles, the displayed geotiles are encircled by an outline. This way the user may more easily notice if there any geotiles scattered outside the main distribution.

In the device-specific window the user can also see list of other devices that are associated with the selected device. This is relevant in the user domain, as certain devices may be linked so that they are expected to take part in processing the same data. By seeing the associated devices, the user may see if there are any unexpected devices associated with each other. If some undesirably located devices are associated with each other, this may lead to undesired behavior in the monitored system.

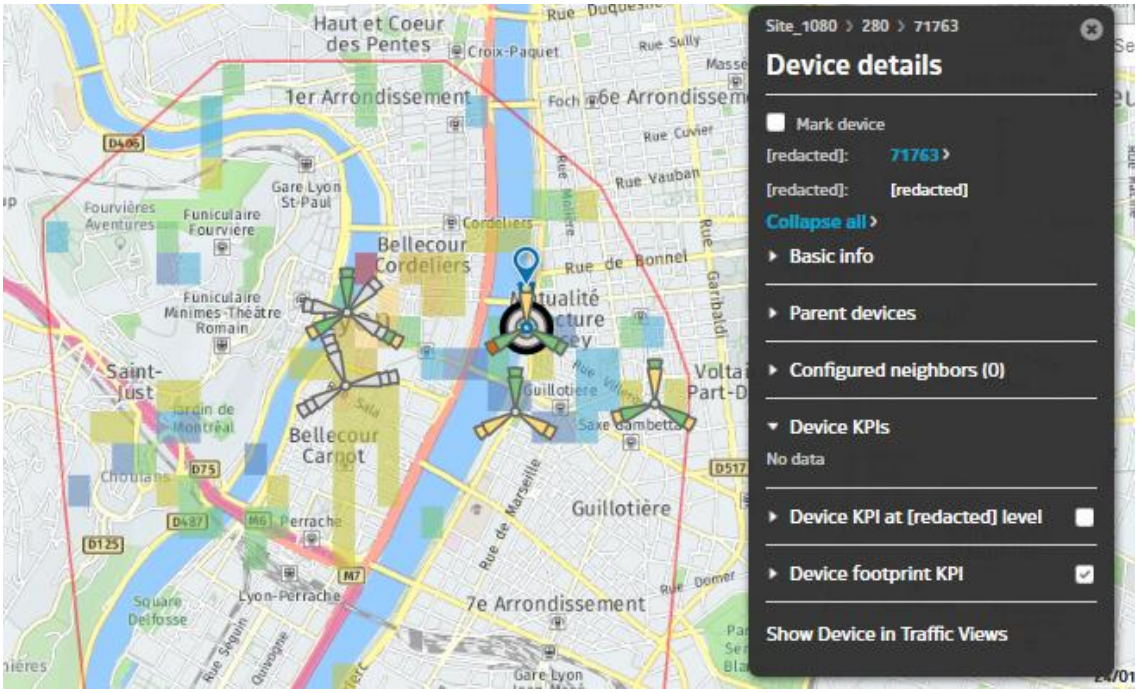

Figure 21. Geotile data as aggregated by both geotile and associated device ("device footprint"). Note the line that shows the boundary of the area with geotile data.

The devices that are in the map's visible area are shown also in *table* below the map (see Figure 22). The table is *sorted* by default by the selected device attribute (see device attribute selector in Figure 11) and the user can also sort the table by other attributes as well. The displayed attributes include certain *static* device attributes and all *dynamic* per device aggregated attributes. The map serves as control for the table contents, as the table content is updated to show the devices from the map area as the user pans around the map area.

Figure 22. The shown map area's device data in a table.

The table's *sorting* feature allows the user to quickly see any extreme values in the dynamic aggregated per-device data. The default sorting of the table by selected device attribute follows the semantics of each attribute so that the values that are judged to be of most interest to the user are sorted to the top. Thus, for valence attributes the values indicating poor performance are sorted to the top, while for quantity attributes the largest values are at the top. This means that the sorting direction may change between ascending and descending order as user changes which device attribute is displayed. While this breaks consistency of behavior (where all attributes would be sorted in either ascending or descending order), it was judged based on expert feedback that this will aid users to discover the most interesting devices.

The device items on map and the rows of the table are *facets* of the same data that provide complementary views on the data. The table allows the user to have a more comprehensive view of the device values than what the vis alone can provide, as the table displays simultaneously many values with numeric presentation, while the map shows the geographic information for each device. The relationship of map items and table rows is communicated to the user with mouseover highlight: as the user moves the mouse cursor over either a map item or table row, the corresponding table row or map item in the other view is given the same visual highlight as the highlighted item or row. The user can also select rows from the table, in which case the selection opens the device-specific information window the same way as if user had selected a device item from the map. Same way as with mouseover highlight, the selection highlight is also displayed both for related map item and table row.

In addition to device selection and highlight, the user can also mark devices. This is done with checkbox components in the device table (see column "Marked" in Figure 22)

or device window (see “Mark device” checkbox in Figure 21). The marked devices are highlighted on the map with black pin icons, and they will receive special treatment in the map’s semantic zoom. They will remain visible on the map the same way selected and highlighted device items do at higher zoom levels, where other device items are removed from view. This supports the user’s memory by allowing the user to keep track of multiple interesting devices as they pan the map and inspect other devices. Figure 24 shows the device items displayed at a zoom level where they all remain visible. Figure 25 show the same view one zoom level higher: device items are hidden, save for the marked one.

The device marking plays an important role in user workflow from the dynamic vis into the historic vis by allowing user to mark devices in dynamic vis and then having them displayed in the historic vis.

The device marking also has a specific use within the historic vis. In the historic vis the list of the marked devices’ IDs is automatically inserted in the query component. This allows the user to easily pick devices from the map and the table for use in the historic data query.

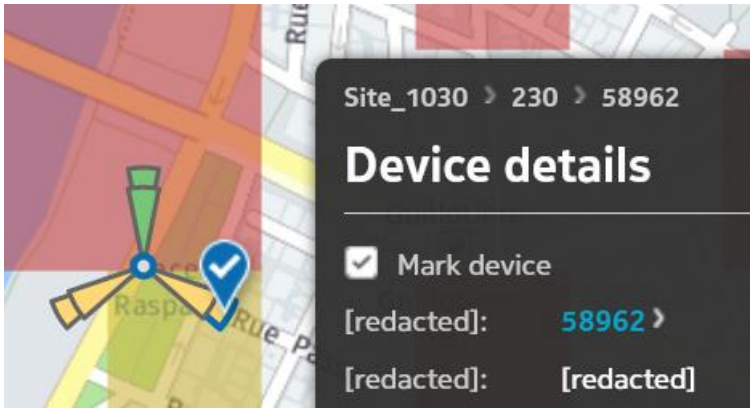

Figure 23. Marker pin on a selected device item.

Figure 24. Marker pin on on a device item that is not in selected state.

Figure 25. Marker pin on on a device item that is not in selected state, with map zoomed out so that device items are otherwise hidden due to semantic soom.

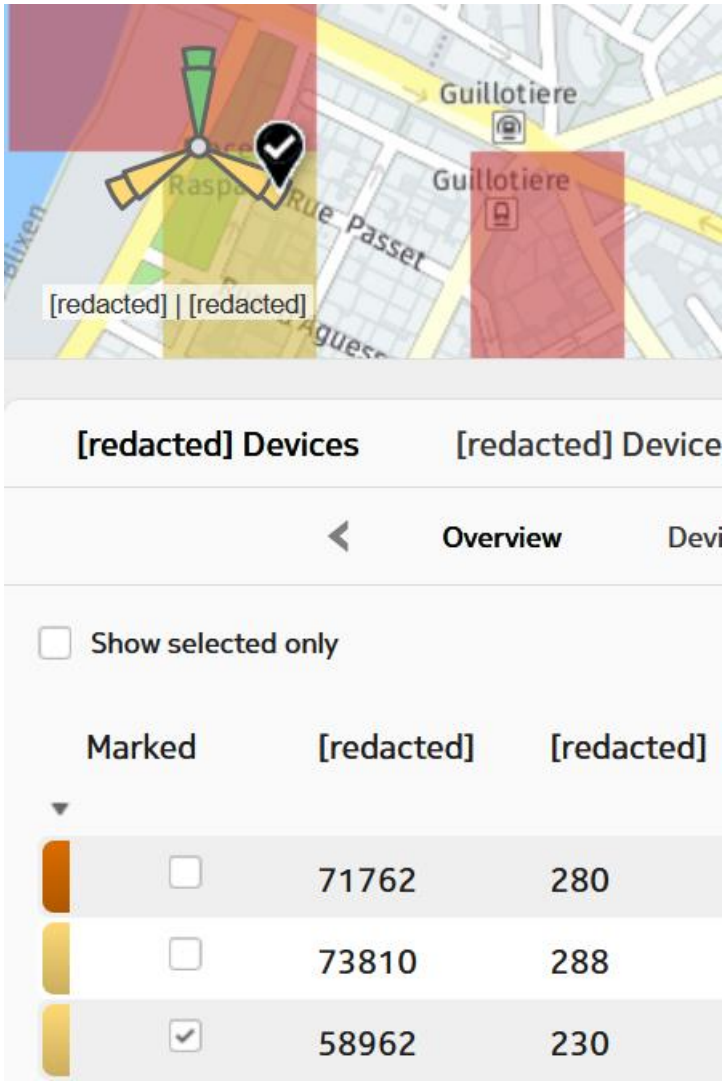

Figure 26. Marking on a device displayed both as pin on the map and as checked checkbox in the table.

4.1.4.4 *Device data by distance*

Figure 27. Device-specific data aggregated by distance from the device.

Idiom	Device data by distance
What: Data	Tabular data items with derived rough geographic location
Why: Tasks	Browse to find outliers with unexpected location, find extreme values by sorting table
How: Encode	Use given geographic location, shape to indicate orientation, map value to color saturation
How: Order	Sort table to find extreme values
How: Facet	Table in the device data window, geographic location derived from the associated device location and the data item's distance to the device

Table 3. Device data by distance idiom taxonomy.

In the device window, the user can select to see yet another aggregation vis of the dynamic data: data per distance from certain device (see Figure 27). This vis displays device-specific data aggregated into segments by the data item's distance from the device. The cone shape is directed into same direction as the device itself.

This vis provides a complimentary view on the data. To aggregate data items into geotiles, the data items must have coordinate data. This is not the case for all data items, as the coordinate data may sometimes be missing or corrupted. In such case, the data item may still contain a certain attribute value that indicates its distance from its associated device. With this information, the data is aggregated also by this distance information combined with identity of the data item's associated device so that data that would otherwise be useless can still be put into use in the geographic vis.

The cone displays the values for the attribute that user has selected for device KPI. This attribute is encoded by *color saturation*, making it different from *color hue* encoding used for geotiles and device items. This approach was chosen because the cone vis is overlaid on top of geotiles and overlaying two hues would distort them both. By overlaying only one hue with varying luminance on top of many hues with single luminance the distorting effect is limited.

The cone angle width is defined so that it corresponds to generally expected direction and distribution where data items are in relation to the device. This is however only an educated guess based on domain knowledge since each aggregated segment only contains information in its distance from the device location.

The vis provides additional help in discovering locations with undesired data values. As the user identifies a location of interest with the distance vis, they can then inspect geotiles at that location to gain more insight into the data in the location. Usually the data location is interesting if it is at an unexpected distance away from the device, as this may indicate undesired behavior in the monitored system.

The vis allows the user to browse through devices by selecting them one by one and quickly glancing the cone vis for each. Similar glancing can also be achieved by enabling geotile filtering per device, but in such case the user would lose the general view of all geotiles. This combined use of general geotiles and device-specific data with the overlaid cone vis enables the user to see how device-specific data compares to the whole data while browsing through devices.

4.1.4.5 *Individual data items*

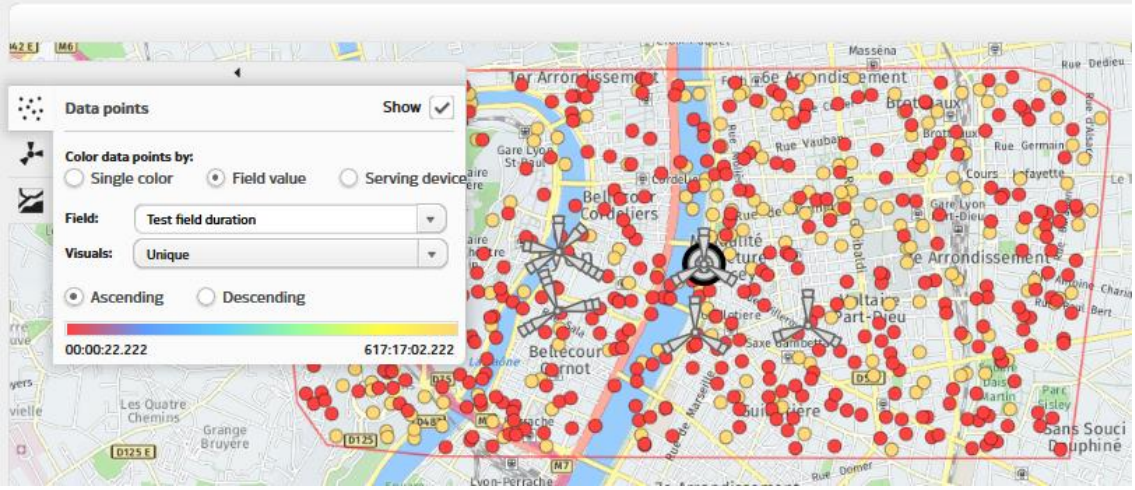

Figure 28. Data points for individual data items on historic vis map.

Idiom	Data points
What: Data	Data items with geographic location
Why: Tasks	Browse data items returned by given query parameters, find extreme values by sorting table
How: Encode	Use given geographic context, map to color hue
How: Order	Sort table to find extreme values
How: Reduce	User queries data from limited time and geographic area
How: Facet	Juxtapose geographic context and numeric data values by displaying the same data items in table next to the map

Table 4. Data points idiom taxonomy.

The vis idioms described in previous chapters were for the dynamic aggregated data. As mentioned, the motivation for performing this data aggregation and providing vis for it is to enable users to gain overview of large quantities of data, and to reduce the data into a smaller quantity so that it can be retrieved for inspection quickly.

This aggregation comes at a cost however: aggregation loses information as it compresses data items [Munzner, 2015, p. 300]. In the aggregated dynamic vis there is no way to see the actual individual data item from which the aggregated representation was created. This limits the user's ability to gain insights from the data, as some insights may require investigating the data down to the level of individual data items.

To offset this cost of aggregation, the vis tool provides a vis for individual data items. As opposed to the dynamic vis that uses the quickly accessible dynamic aggregated data, the vis for individual data items requires the user to query the raw data items from database and displays them as circular items on the map (see Figure 28). This vis occupies an entirely separate, alternative view from the dynamic vis, as it uses different kind of data and a vis idiom that does not coexist nicely with the geotile idiom.

At the top of the page there are controls by which the user can input conditions for the database query. This is a significant difference from the previously described real-time vis idioms. With the real-time vis idioms, the displayed data is automatically updated at regular intervals to reflect the newest status of the data, whereas to view the individual data items from the historic data, the user must first specify by which conditions they want to retrieve data for inspection.

The user must always provide time range parameter for the query, and some query types also take in geographic area as a query parameter. The tool allows the user to use the map component as input element for specifying the geographic area: the user can pan and zoom the map to view their desired geographic area and click the button in the query conditions to use the map's visible area as bounds for the geographic query conditions. Figure 29 shows the query input component in state where it automatically picks up the map area as query parameters.

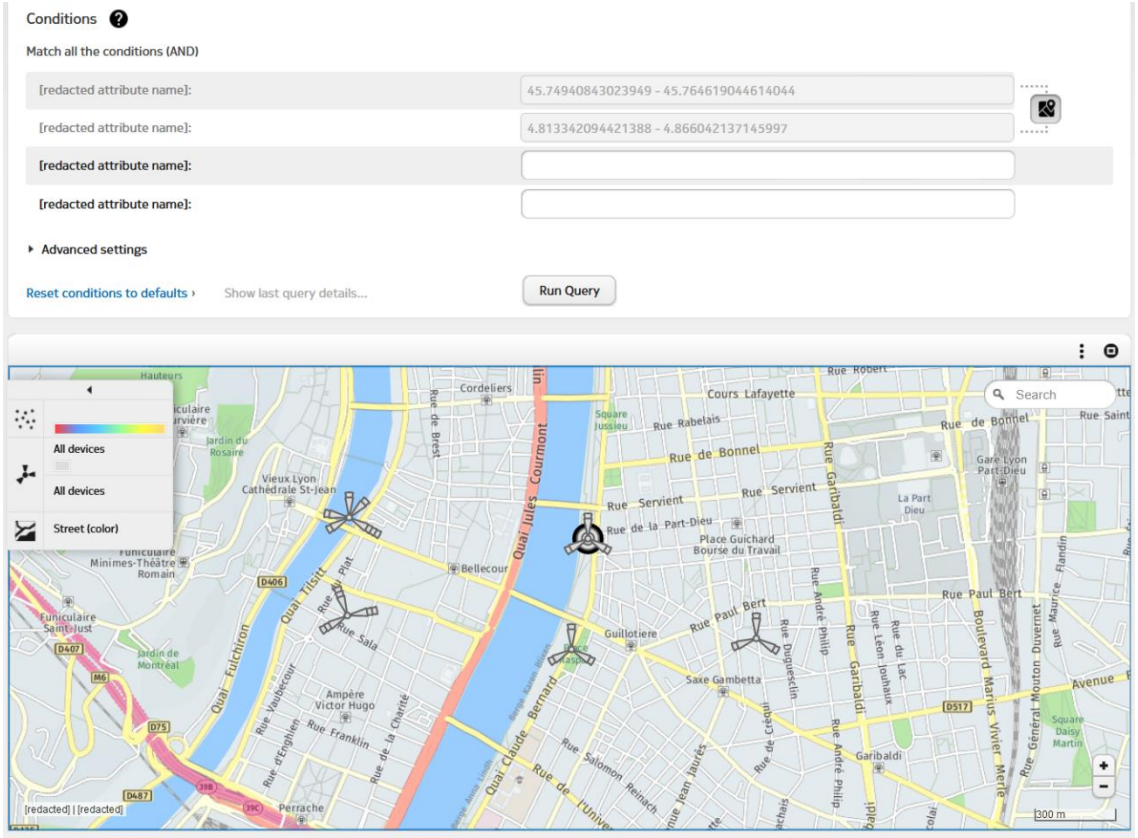

Figure 29. Visible map area used as input for query's area parameters. Note the blue outline of the map, depressed visual appearance of the toggle button on top right and greyed out text inputs to indicate that the automatic area input selection is active.

When the user provides conditions and launches a query, the matching data items are retrieved and displayed on the map as circular data point items whose location on the map encodes the data item's coordinate values.

Each data point item also contains a color *hue* encoding for a data item's attribute value. The same way as with the dynamic vis idioms, the user can choose the displayed attribute from the controls at the map edge (see Figure 28).

As the vis for individual data items requires the user to provide the conditions before any data is displayed, it requires the user to have a pre-existing knowledge on what to look for. This is where the complimentary nature of the historic data point vis and the dynamic aggregated vis comes in. The user can identify phenomena with the dynamic vis and go to historic data point vis to inspect the involved data more closely. In fact, this workflow is supported in the dynamic vis with a control element in the map corner. This pulldown menu contains certain pre-defined so-called launches that will take the user to a new browser tab that displays the page for the data point vis and fills out the query conditions based on the state of the dynamic vis from which the launch was done. Figure 30 and Figure 31 show how marked devices in the dynamic vis are carried over into the historic vis. When user selects a query template item from the pulldown component as

shown in Figure 30, a new browser tab is opened with historic vis as shown in Figure 31. This approach follows the principle of "Overview first, zoom and filter, details on demand" [Shneiderman, 2003], where the user uses the given vis to narrows down the data they inspect if their domain need requires them to inspect locate and closely inspect some limited amount of data.

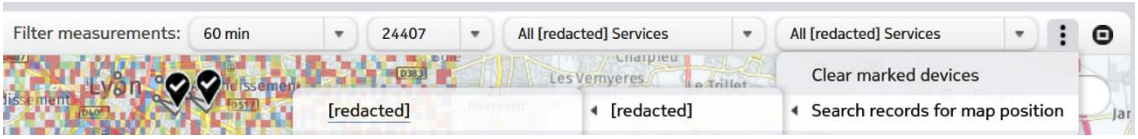

Figure 30. The nested pulldown for selecting a query template to launch from the dynamic vis to the historic vis.

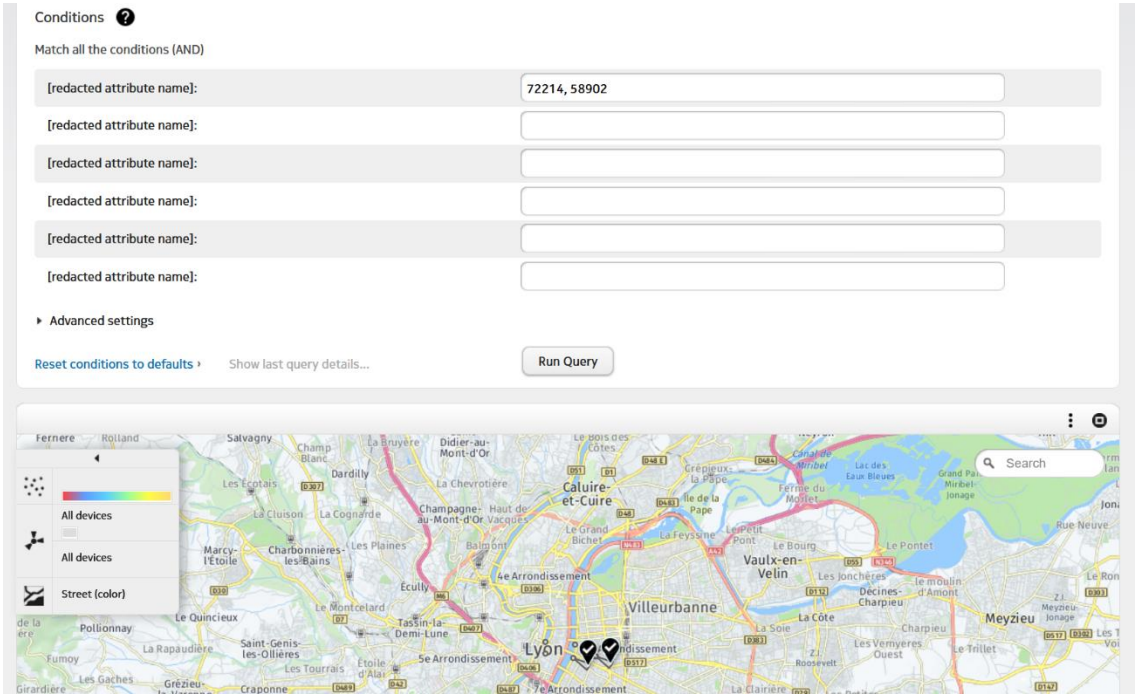

Figure 31. A query template launched from the dynamic vis to the historic vis. The marked devices have carried over from the dynamic vis as paremeters on the launched query template.

4.1.4.6 *Interactions*

The vis tool employs multiple vis idioms to display many views on its data. To carry out their tasks the user must make use of all of them using the interactions the vis tool provides. This chapter explains the most significant interactions and the user tasks that they support.

When the user opens the dynamic vis, the map shows a pre-configured area. The system administrator can configure this default map location so that it will be close to where the users need it. The default zoom level is such that if there are geotiles in the map area, they will be visible. At this stage the user may want to either monitor *overview* of the data over time to *discover* any unexpected phenomena in it or to seek to *identify* some locations and devices that are associated with some problematic data.

If the user wants to monitor *overview* of the data they will select the displayed geotile attribute and then *pan* the map to shift the displayed map area and *zoom* in and out in the map to balance the size of the shown geographic area with the discernibility of the geotiles. They can also toggle the map to fill the entire browser window so that they can use maximum pixels for the vis. The user can set the displayed time period to something that fits their interest, like one hour. Then the user can start focusing on some other tasks while the vis updates periodically with the newest available data. In this case the vis works as an *information radiator* that ambiently displays information, and users can glance it amidst their other tasks to get an *overview* of the data. This way they can discover undesired phenomena in the data, if the color *hue* in geotiles in some location changes to indicate extreme values. Ott and Koch [2019] have conducted a study where they explored the usefulness of information radiators.

If the user wants to *identify* locations or devices that are associated with some undesired behavior in the system, they may have gotten to the starting point in two ways. They may have *discovered* extreme values by monitoring the *overview* as explained earlier, or they may have been alerted some other way in their work environment to *locate* certain problematic area or device.

If the user has spotted an abnormality in the map, they will already know what location to inspect more closely. If the user has been alerted of abnormal behavior in the system at some geographic location, they will know a street address near a location where abnormal behavior has been reported or identity of a device associated with the abnormal data. They can *locate* either street address or a device with the search input at the map edge. As the user inputs an address or a device ID, they are presented with a list of matches from geolocation database and device database. The user can then select items from this list and the map is automatically panned to the location of the selected street or the device. When selecting a device from the list, the device is also selected the same way as if the user had selected a device item from the map, that is, the user will see a window with

details of the device and the device will be highlighted on the map and in the adjacent table.

When the user is investigating abnormality in a geographic location, they may need to investigate the data on a more detailed level than what the aggregated and time-fixed data of the dynamic vis allows. This is where the historic vis comes in.

The workflow is fundamentally different in the *historic* vis, where there is no quickly accessible overview of the historic data. The user must first provide the query parameters for the database query that retrieves the data for display. What parameter values to input may come from external systems, whence the user inputs the parameters entirely manually, but the main interaction flow has been designed to start from the *dynamic* vis. In the *dynamic* vis, there are nested pulldown controls at the top of the map which launch the *historic* vis with certain input data from the current state of the *dynamic* vis (see Figure 30). This data includes the current map bounds and the time period of the data in the dynamic vis, and a list of marked devices. The devices are marked as explained earlier. The options in the pulldown represent different query templates with different data attribute query fields and possibly with some default values. When user selects a launch option, a new browser tab is opened with the *historic* vis so that the map area, time period and the list of marked devices carries over from the *dynamic* vis and are automatically inserted into the query fields of the *historic* vis. From this point forward the user can query the individual data items and inspect their exact data values with the *historic* vis. From the individual data items the user can identify devices that relate to the individual data items and their exact, unaggregated data values. This way they have more precise information on the relationship of devices and their associated data items so that they can pinpoint devices that require corrective action with greater confidence.

Figure 32 shows a representation of the main interactions between idioms in the vis. The graphic and its relation to Munzner's concepts is discussed further later in this chapter.

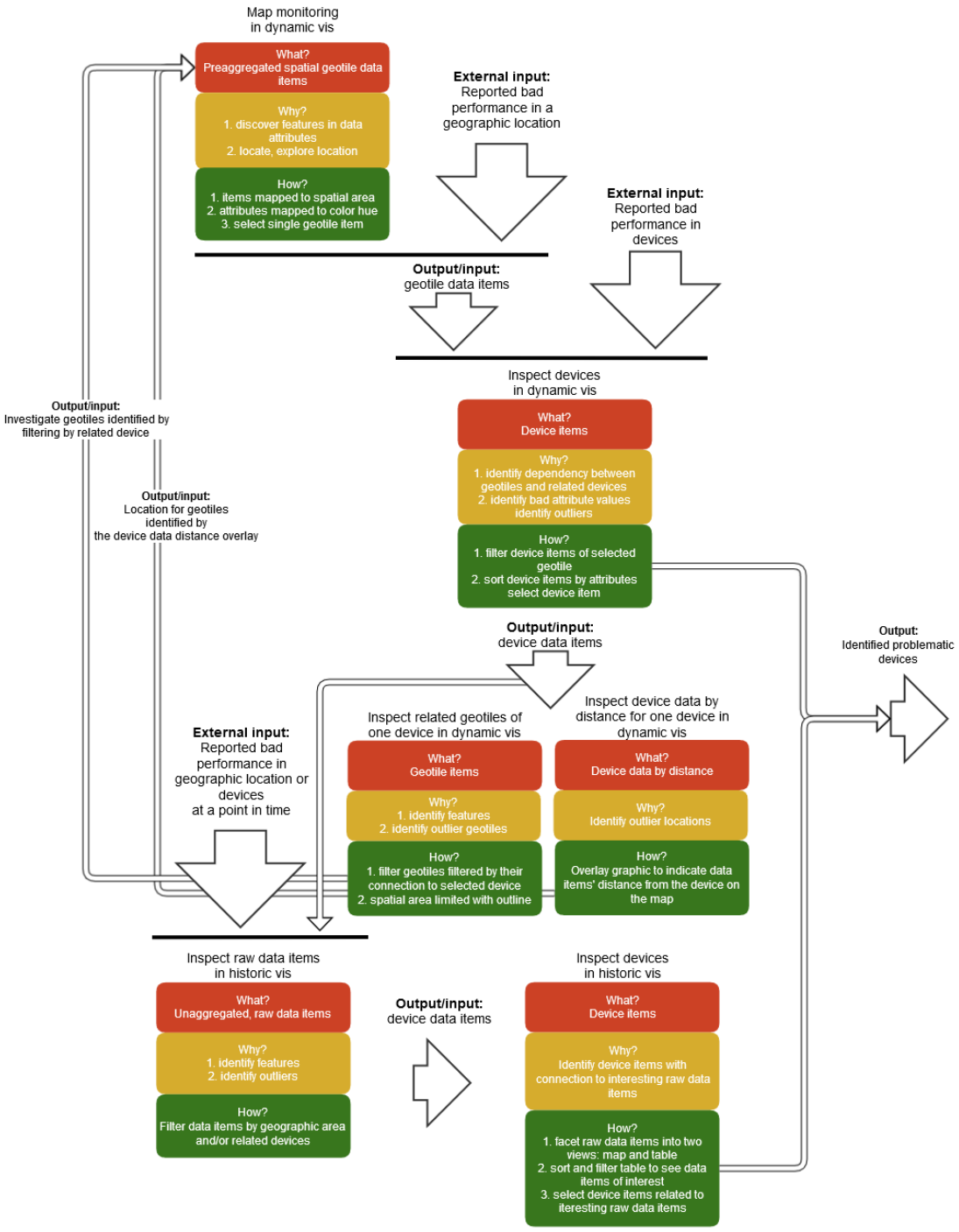

Figure 32. Interaction graph between the idioms of the vis tool.

4.2 Vis product: addressing threats to validity

In this chapter I analyze the validation of the vis product's designs from the perspective of what Munzner [2015, p. 74] describes as **threats to validity**. At each level through the design process there are threats that can result in a vis that does not satisfy user needs. These threats can be mitigated by certain actions throughout the design process, both for validating decisions before implementing the vis (*pre-validation*) and for validating the vis after it is implemented (*post-validation*). Figure 33 shows an overview of these threats across design levels and their mitigation actions.

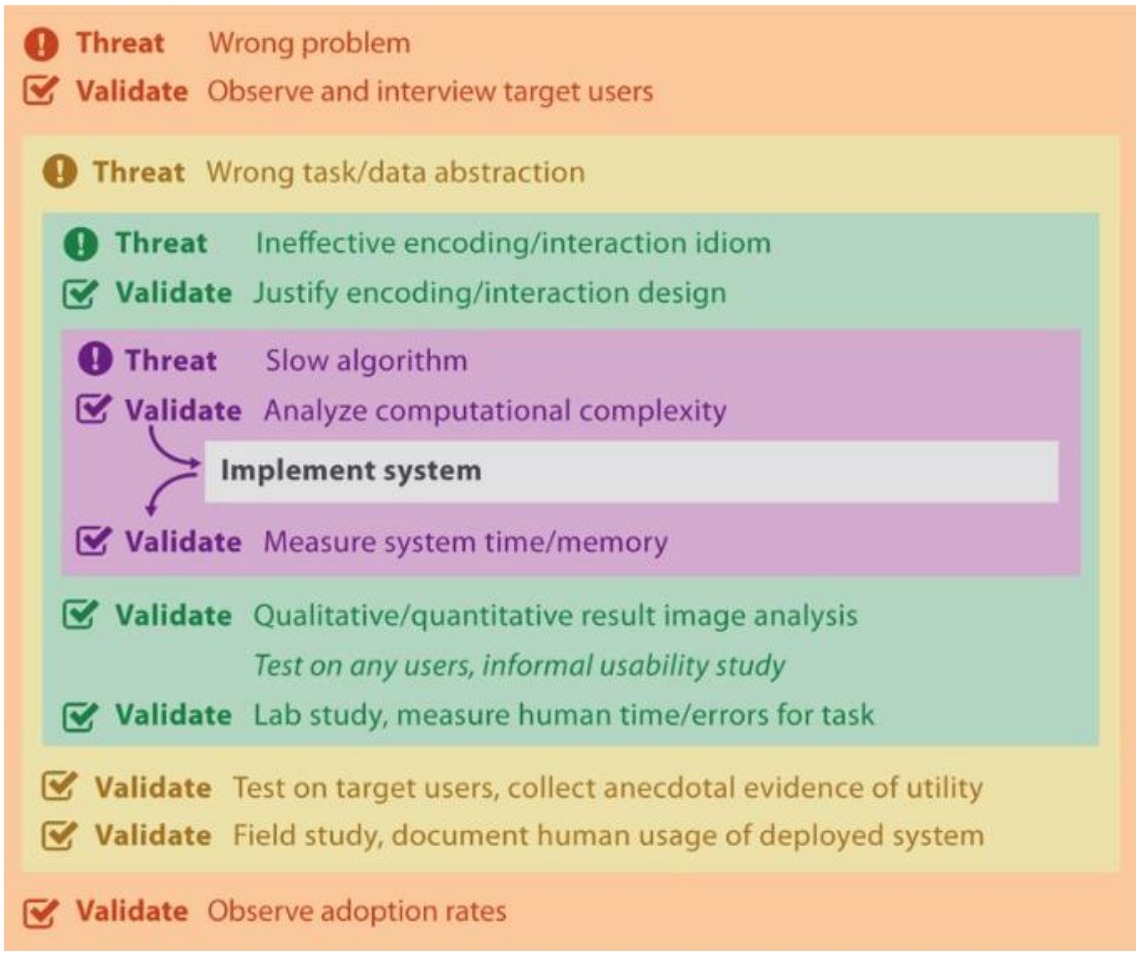

Figure 33 "Threats and validation at each of the four levels. Many threats at the outer levels require downstream validation, which cannot be carried out until the inner levels within them are addressed, as shown by the red lines. Any single project would only address a subset of these levels, not all of them at once." [Munzner, 2015, p. 76]

The following chapters deal with how the vis product has been validated against these *threats to validity* at each level of design. Chapter 4.2.1 delves in *pre-validation* actions and chapter 4.2.2 delves in *post-validation* actions.

4.2.1 Pre-validation

4.2.1.1 *Domain situation*

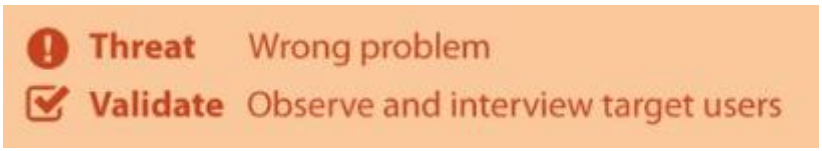

Figure 34. Design validity threat and pre-validation action at domain situation level. [Munzner, 2015, p. 76]

The way for validating *domain situations* is to observe and interview target users. In the design work of the vis product a designer interviewed target users at potential customer organization. Thus, pre-validation has been done at domain situation level, but it must be acknowledged that interviews are not as powerful tool as field studies where users are observed in their work. The weakness of interviews is that there is no guarantee on how conscious the users are of their workflows or how aptly they can describe them verbally. It isn't feasible to require this of them either. User observation overcomes this problem by allowing designer to gain first-hand insight on user workflows, which may contain seemingly minor yet valuable details that can be used to formulate domain situations.

Munzner [2015, p. 68] points out that if designers struggle with defining *data/task abstractions* at the lower design level, the reason may be inadequately formulated domain situation. In such case the designers should return to domain situation level and try to improve or extend the domain situation.

4.2.1.2 *Data/task abstraction*

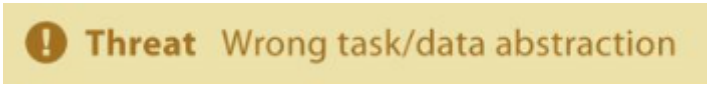

Figure 35. Design validity threat data/task abstraction level. [Munzner, 2015, p. 76]

There are no pre-validation methods for *data/task abstraction* level. This makes intuitive sense: At this level the designs are abstractions of the concrete user needs defined at *domain situation* level. Due to their abstract nature, they cannot be validated, that is, demonstrated as right or wrong. They must first be translated into concrete *visual encoding/interaction idioms* and possibly also implemented at *algorithm* level before they can be validated. As such there was no validation at this level for the vis product either.

4.2.1.3 *Visual encoding/interaction idiom*

Figure 36. Design validity threat and pre-validation action at visual encoding/interaction idiom level. [Munzner, 2015, p. 76]

At *visual encoding/interaction idiom* level the design can be validated by justifying the design choices against the existing body of design guidelines. That is, designers must be able to view the proposed designs critically. This is of course easier when there are multiple designers: a single designer may not see all potential issues in their own designs.

The design work for the vis product involved two designers who peer-reviewed each other's designs. They also discussed the design with a group of stakeholders in regular meetings throughout the project. These measures brought a lot of valuable input by which proposed designs were chosen and rejected. Thus, it can be concluded that pre-validation at this level has been adequate.

4.2.1.4 *Algorithm*

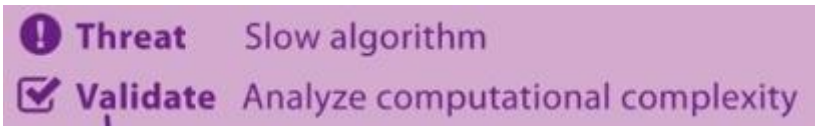

Figure 37. Design validity threat and pre-validation action at algorithm level. [Munzner, 2015, p. 76]

At *algorithm* level the implementation can be pre-validated by analyzing the computational complexity and thus efficiency of the chosen algorithms that process the data for the visualization.

The vis product's functionality leverages the parent product's existing data aggregation capability. Thus, the computational complexity was tackled by the existing functionality that was found performant through the parent product's history as a commercially successful software tool.

The algorithms in the vis tool itself were not analyzed before implementation. This was compensated by testing the whole implemented system's performance both by automated and manual performance testing after.

4.2.2 Post-validation

4.2.2.1 *Algorithm*

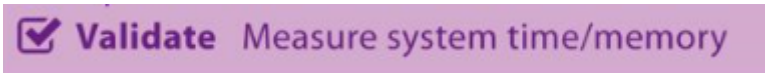

Figure 38. Post-validation action at algorithm level. [Munzner, 2015, p. 76]

Post-validation at *algorithm* level means measuring the computation speed and resource usage of the system.

The performance and stability of the vis product's upstream components was tested extensively with automated performance and stability tests. These tests perform requests on the interfaces of the client component's data over time periods ranging from hours (performance tests) to days (stability tests).

The client code's performance was validated manually by using the system with large data amounts. The time and memory use were measured quantitively only in rare cases where the performance of some particular functionality was at question. Otherwise, the validation was done qualitatively: the client's performance was adequate when the people using it (designers, developers, other stakeholders) were satisfied with the response times.

4.2.2.2 *Visual encoding/interaction idiom*

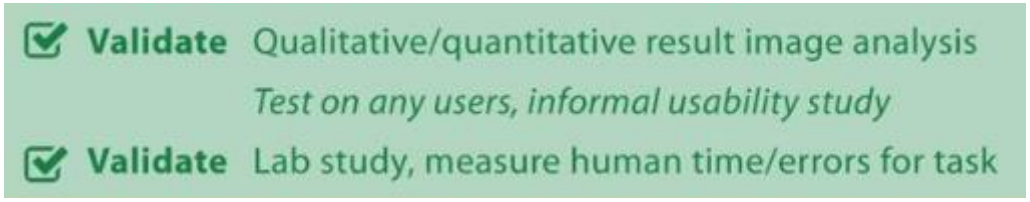

Figure 39. Post-validation actions at visual encoding/interaction idiom level. [Munzner, 2015, p. 76]

The chosen visual encodings and interaction idioms at *visual encoding/interaction idiom* level can be validated in three ways: 1) The visualizations produced by the system can be analyzed *qualitatively* in a discussion to validate that they support the intended user tasks, 2) the visualizations can be analyzed *quantitively* by measuring their visual properties against certain *quality metrics*, and 3) the system can be tested with users in a controlled *lab study*.

Of these three validation methods only *qualitative* discussions were used with the vis product. This was done by regularly demonstrating new implemented visualizations to PM, proxy user, domain specialists and other stakeholders. This way the visual encodings and interaction idioms were regularly subject to critical discussion on their validity for intended tasks.

Quantitative measurements were not performed. The concept is unfamiliar to the designers, though Munzner [2015, p. 79] also only mentions them in passing and points out that only some of such metrics have been empirically tested.

There were no *lab studies*. The designers conducted several usability tests on the vis product following basic usability test procedure [Sillanpää and Koivuniemi, 2019], but as Munzner [2015, p. 79] points out, usability tests validate usability of a system, not how effectiveness of a visual encodings and interaction idioms of a visualization design. Where usability tests validate how effectively users can carry out the interactions that designers have intended, controlled tests for visual encodings and interaction idioms would require test scenarios that would validate the insights that a visualization enables users to gain from data qualities.

4.2.2.3 *Data/task abstraction*

Figure 40. Post-validation actions at data/task abstraction level. [Munzner, 2015, p. 76]

Validation at *data/task abstraction* level requires having a target user use the tool. This differs from *lab studies* at *visual encoding/interaction idiom* level in that here actual intended users of the tool use the tool for their actual work. This validation can be made either by *collecting feedback* from users using the tool, or, more rigorously, by conducting *field study* to observe and document target users using the system in their work.

With the vis tool a designer visited a customer to collect feedback from target users. The designers also received continuous feedback from proxy user who evaluated design's usefulness for intended tasks whenever new designs were implemented. Field studies were not done.

4.2.2.4 *Domain situation*

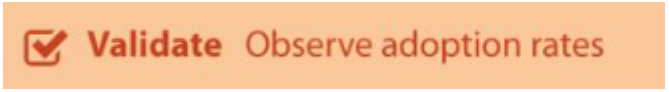

Figure 41. Post-validation action at domain situation level. [Munzner, 2015, p. 76]

Designs are validated at domain situation level by observing how well the designed system spreads. This is a difficult measure, as the quality of designs is not the only defining factor for a system's adoption rate, especially for commercial products.

As such in the situation of the vis product, the design's effectiveness is hard to judge from the product's commercial performance. At the minimum it can be concluded that

the *domain situation* definitions and the resulting designs have been effective enough for securing the deals that the product has seen. It does not, however, tell if the *domain situations* could have been defined more accurately. As such it might be useful to seek feedback from the sales organization on any customer contacts that have not resulted in a purchase. Such feedback exists in the organization but is not readily available to designers. It would be useful for designers to seek this feedback as they accumulate experience to support their future design work.

4.3 Vis product: Closer look at design and validation gaps

In this chapter I delve deeper into some aspects of the vis product's design work that arose from its investigation and interviews I did with its designers.

4.3.1 Domain situation's availability to designers

As described in earlier chapters, the design levels between *domain situation* and lower levels fall into different branches in the company. This means that effective designs require effective collaboration between the organization parts. However, there is some friction in this collaboration.

Designers reported that they had interviewed several users at customers at the start of the project and during development they had access to user at one customer [Sillanpää and Koivuniemi, 2019]. During the development the development team also closely collaborated with an in-house specialist who was in close contact with customers. That is, that specialist was a *proxy user*, with whom the designers and developers could define requirements even at low design levels and evaluate designs and implementations. The designers noted that this cooperation was very valuable. However, they also pointed out that the in-house specialist is still not an actual user, but a mediator between the users and the development team, who has their own opinions and preferences. Thus, the information gained from the proxy user must be considered critically against this fact. Also, using a proxy user does not allow for user observation.

The designers claimed insufficient access to end users from which they could gain feedback on the design's effectiveness, noting that this would require much more frequent co-operation with customers or easy internal channel to get information from regional sales teams [Sillanpää and Koivuniemi, 2019].

4.3.2 Support and hindrance from pre-existing data processing

The dynamic vis is built on the parent product's capability to dynamically aggregate data. Leveraging the existing data aggregation functionality comes at a cost of restricting data availability to a fixed time frame, thus restricting available tasks and influencing design choices. It has also limited the design space for the geographic data so that there hasn't really been other option that the geotile idiom, with the restrictions that is has.

However, this pre-existing aggregation capability may be the sole reason that the vis tool exists. It is uncertain if the vis tool would have been deemed worth creating if there had been additional cost of implementing new functionality that would have solved this limitation. As such it can be argued here that perfect is the enemy of good. On the other hand, it is also uncertain how good investment the product is to the product organization with this limitation. The question is hypothetical, but it should be acknowledged since the design's effectiveness is judged against the available budget for design and implementation.

4.3.3 Design choices

The design choices have been validated with cooperative evaluation by the involved designers and stakeholders as explained in chapter 4.2., and thus the tradeoffs and concerns of the designs have been justified. It is, however, worthwhile to look at some of those concerns here.

Usage of *categorical hue* color encoding is a major design choice with several chosen vis idioms. As the data attributes encoded this way are ordered and quantitative, a more natural encoding for them would use *saturation* or *luminance* encoding, which lend themselves for encoding continuous differences across the ended value range. However, it was identified in discussions with domain specialists that such comparison of differences is not what the users need. Rather, they need a way to spot values that fall below, above or between certain thresholds. Because of this the exactness of the encoded attributes was willingly sacrificed by using a *categorical* encoding. Munzner [2015, p. 234] notes how this kind of attribute transformation from *quantitative* to *ordered categorical* can be justified this way by task semantics.

A major tradeoff with aggregating spatial data into geotiles is how geotiles are affected by *modifiable areal unit problem* (MAUP) [Wong, 2004]. That is, as spatial data is aggregated into discretely bounded units, the placement of the bounds may break and obfuscate patterns in the data. For example, if some attribute in the data is aggregated by taking an average of certain other attribute, and the locus of some trend in that attribute falls on the border between two units, the data items of the trend fall into different units and are averaged together with data points where the trend does not occur. In such case the trend is diluted by this split and uninvolved data points, more so than if the whole of the trend would fall in a single unit. This diluting effect of the MAUP can be mitigated by using smaller units of aggregation. This, however, would come at a computational cost as the number of aggregate units increases.

There is also a concern of discoverability in the dynamic vis in general. As the user can at a time only choose to see a single attribute displayed for geotiles and another one for devices, they must know which one to choose. This problem cannot be tackled with a single map. As Munzner [2015, p. 267] notes, a single vis view can display only so many

attributes before the amount of visual clutter grows too large for the user to handle. This is especially true for colormaps: multiple colormaps cannot occupy the same the map.

As Munzner further notes, the limitation of a single vis view can be overcome by using multiple views. This is also the case with the vis tool. As the tool is implemented for web browsers, the solution leverages standard web browser functionality: the user may open multiple browser windows where they can see multiple maps with different data. As the views in different browser windows are independent from each other, any potential interactivity of these multiple views is sacrificed. However, this was deemed as worthy sacrifice against the cost of designing and implementing a multi-map view inside a single browser window, or the cost of adding a lot of complexity by introducing interoperability between the browser windows with web technologies such as cookies.

Another justification for limited discoverability is that the users operate in an environment with multiple complementary tools and are not solely relying on the vis tool. That is, it is expected that the users will have other sources of information for deciding what to look at with the vis tool.

4.3.4 Verification of implementation vs. validation of designs

The development work involved great effort on verification at many levels: unit tests, integration tests, stability and performance tests, end-to-end test automation and manual tests in end-user environment. This greatly helped ensure that the system worked in a performant and stable way and greatly helped to eradicate bugs. However, while valuable, this test load is *verification*, not *validation*. The tests ensured that the system works as designed, not that the designs truly fulfilled user needs.

Full validation of chosen design abstractions would require observing target users using the vis in their own environment [Munzner, 2015, p. 78], which has been limited, as actual users have largely been supplemented with in-house specialists as “proxy users” [Sillanpää and Koivuniemi, 2019]. A designer noted that they interviewed users at later stages of the project on their use of the system, something they found very valuable for their work as a designer. However, as with requirement definition, interviews as a form of validation are limited. Observational user research would yield stronger insights on design’s effectiveness. Interviews allow for identifying glaring issues in workflows and finding out new functionality ideas from users, but this is limited in usefulness in the same way as interviews are limited in requirement definition: users are experts in their domain, not in visualization design. They are not aware of the *design space* of possible designs [Munzner, 2015, p. 10] to the extent that designers are.

4.4 Munzner's concepts

4.4.1 Interactions

The vis tool displays multiple perspectives on its data with multiple vis idioms and the user navigates around the tool shifting focus from one idiom to another to investigate the behavior of the monitored system.

The concepts that Munzner presents offer a comprehensive terminology for describing and analyzing the different idioms. However, the interactivity that joins the idioms together is itself a major realm of design choices.

Munzner [2015, p. 9] stresses that interactivity is crucial for handling complex visualizations and points out that interactivity is integral part of many vis idioms [2015, p. 71]. She also explains how complex user tasks can be described as chains of simpler tasks, where outcome of one task provides starting point for the next [2015, p. 44, 49]. Figure 42 shows the approach Munzner takes at interactions between idioms: each idiom is to be considered as its own set of data-task abstractions, and one idiom feeds input context for next idiom.

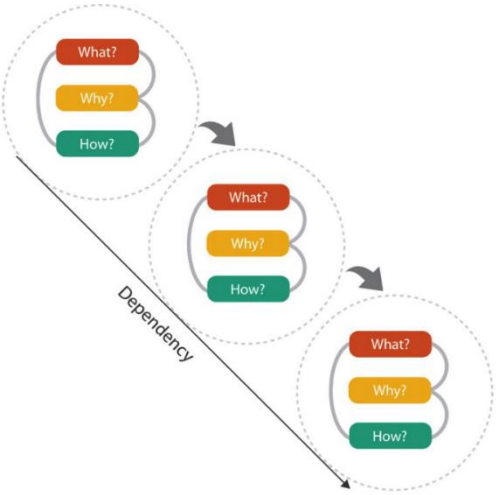

Figure 42. "Analyzing vis usage as chained sequences of instances, where the output of one instance is the input to another." [Munzner, 2015, p. 18]

This divide-and-conquer approach does allow breaking down complexity of multi-idiom vis systems, but it turns blind eye to more complex interactions between idioms. A system of chained interactions can be seen in Figure 32 where I illustrated the vis tool's interactions by applying the what-why-how idiom splits into a non-linear graph. It is evident from the figure that the relationships between idioms are much more complex than the linear cascade from one idiom to the next that Munzner presented.

This complexity can be intuitively understood. A multi-idiom vis system can be represented as a graph where each node represents a vis idiom and every link represents

an interaction where the user's focus moves from one idiom to another. Every new idiom in a vis system introduces a new node into the system graph, and at least one link as input into the idiom and one link as output from the idiom. Often one node links to multiple other nodes. In such a graph it is easy to see how adding new nodes may quickly increase the complexity of the graph. As each link, or interaction, represents a shift of one idiom's output to another idiom's input, each time involving the whole scope of idiom-specific design considerations, there is danger of not seeing the forest for the trees. That is, when the user's task involves moving over a network of idioms and connecting interactions, creating an effective design must also consider this whole idiom-interaction network. Designing and analyzing idiom by idiom is not enough, as the idioms must create a whole where the idioms work together to allow user to gain insights from the data. Such holistic design and evaluation is something where Munzner's single-idiom focused concepts offer little support.

4.4.2 Data uncertainty

There is one significant characteristic in the vis tool's data for which there are no supporting concepts from Munzner: uncertainty. The data's geographic location attributes may be inaccurate or altogether false, and with small sample sizes some aggregated attributes may be skewed by few extreme values.

There is inaccuracy in data location as mentioned in chapter 4 where it was explained how the user can filter the data based on the accuracy of the used location method. That is, this known inaccuracy is handled with design by enabling the user to make the choice on how accurate data they want to investigate. This selection is done with a pulldown component in the geotile filter pane (see Figure 43). The component shows the available location-method-specific aggregations: the default selection is data aggregation that contains all location methods, and the rest of the options are for data with varying levels of accuracy. Essentially each option shows data aggregation where every time the least accurate location method is excluded, finally leaving only data with the most accurate location method. The trade-off of the location method exclusion is decreased data quantity for the more accurate location data.

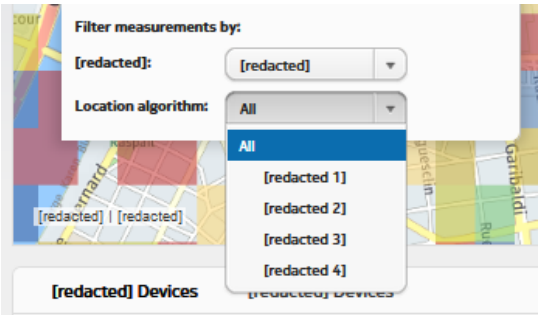

Figure 43. Filter for choosing geotile data set per certainty of used location method. The identities of the loaction methods have been redacted.

In addition to inaccuracies in data location attribute, the location may also be simply false due to several technological reasons in the data collection. In this case the error is in the data that is coming in from the monitored system itself, and the vis tool cannot do anything to correct it. The attribute is corrupted, and the actual location of the data item is not attainable. Fortunately, such corruption usually follows one of several possible patterns, that is, it is easy to spot the corrupted location data if it is displayed with a geographic vis. Because of this it was possible to automatically filter out corrupted data based on these patterns. The downside of this is that some actual data may also be discarded as false positives by such filtering, but this is a worthy tradeoff for preventing false data from cluttering the vis.

Third aspect of uncertainty in the vis tool is the potential overemphasis of minor phenomena in the data due to the geotile idiom. Each geotile has equal footprint in the vis, even though they may encode drastically different quantities of data. As the user may or may not be interested in geotiles that consists of few data items compared to most of the geotiles, a filter dubbed *guard values* was added to filter out geotiles based on sample count. While this allows the user to handle the issue, it pushes responsibility from the design to the user, and is quite heavy handed: the user either sees the "minor" geotiles or they don't.

These sources of uncertainty are something the designers wrestled with considerably, which makes the omission of uncertainty and its implications from Munzner's concepts unfortunate. The methods of accuracy-based aggregation, pre-filtering by known false values and filtering out small sample sizes with guard values appear relatively straightforward design decisions, so lack of vis theory about them is quite surprising.

As mentioned, the use of guard values is problematic due to their all-or-nothing effect: geotile and device items with small sample sizes are excluded completely from the vis. One approach to tackle this would be to encode the uncertainty itself with its own visual channel. That is, items that fall below the guard value threshold could e.g. be given lower saturation and higher transparency to differentiate them from more confident values and to give them less visual weight. This approach would however come with a cost of complexity: this would effectively double the number of colors that geotiles and device items could have. Correll, Moritz and Heer [2018] propose an approach for dealing with such added complexity of visualized uncertainty: *value-suppressing uncertainty palettes (VSUPs)*. Correll et al. propose VSUPs as an approach where the degree of uncertainty is discretized into value bins and the data items are assigned the uncertainty value based on which bin they fall into, and each uncertainty bin is split into discrete bins that encode the "actual" data value. The degree of uncertainty is encoded with its own visual channel (in the case of Figure 44 as lightness and saturation vs. the varying hue of the "actual" data value), but the range of used visual features decreases as uncertainty increases. As a

differentiator from a bivariate color map (as seen on the left side of Figure 44), the number of utilized visual bins for "actual" data values decreases as uncertainty increases. This helps give the values with high uncertainty less impact in the overall vis complexity.

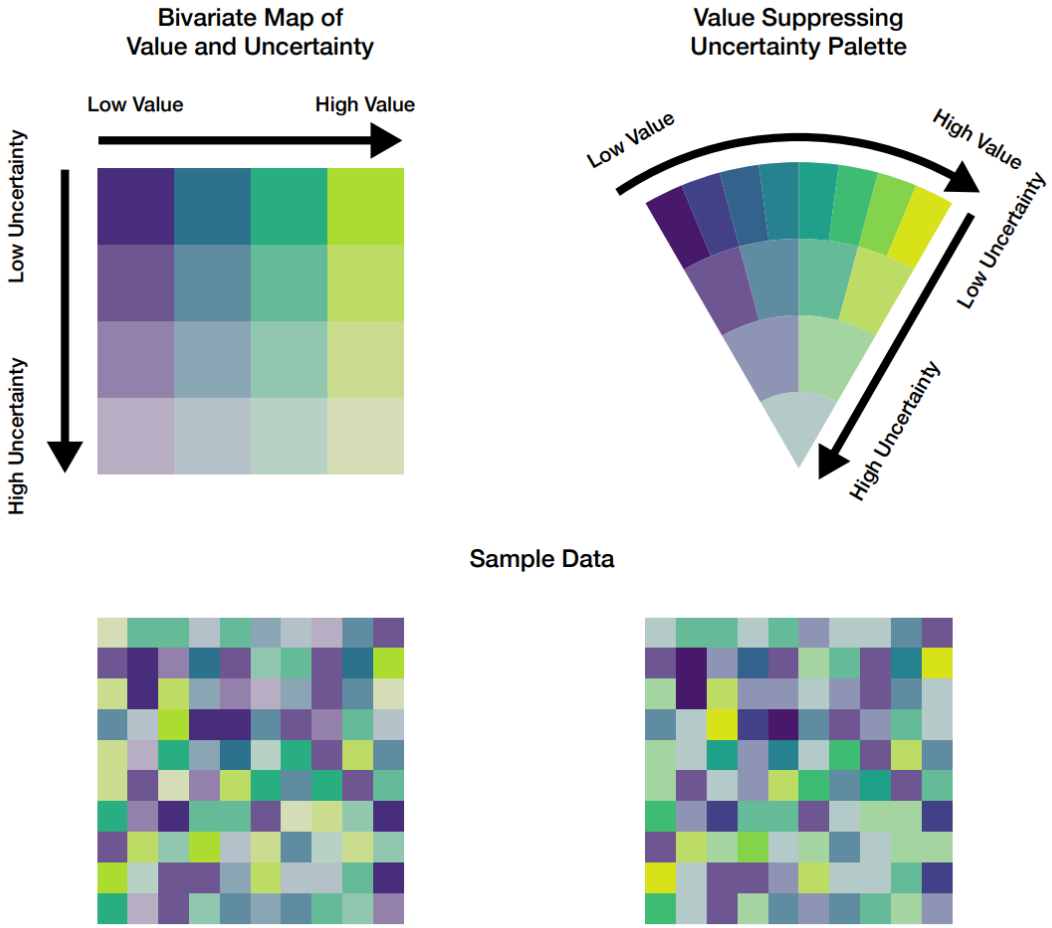

Figure 44. "A standard bivariate map (left) and a VSUP (right)…" [Correll, Moritz and Heer, 2018]

VSUPs could be adapted for use with guard values: data values falling below the guard value threshold could be encoded with a lower saturation and higher transparency, and the scale of the values could be truncated so that only extreme bad values would be given their signature red hue. All the rest of the values could be given a single muted grey hue to dilute their actual values from complicating the vis, while keeping the geotiles themselves visible, and thus keeping the user aware of their existence. This way the user can remain aware of even insignificant geotiles and be able to tell if one of such geotiles contain bad values. Figure 45 and Figure 46 show an example of a VSUP used on geotiles as described.

Figure 45. Geotiles shown without a guard value applied.

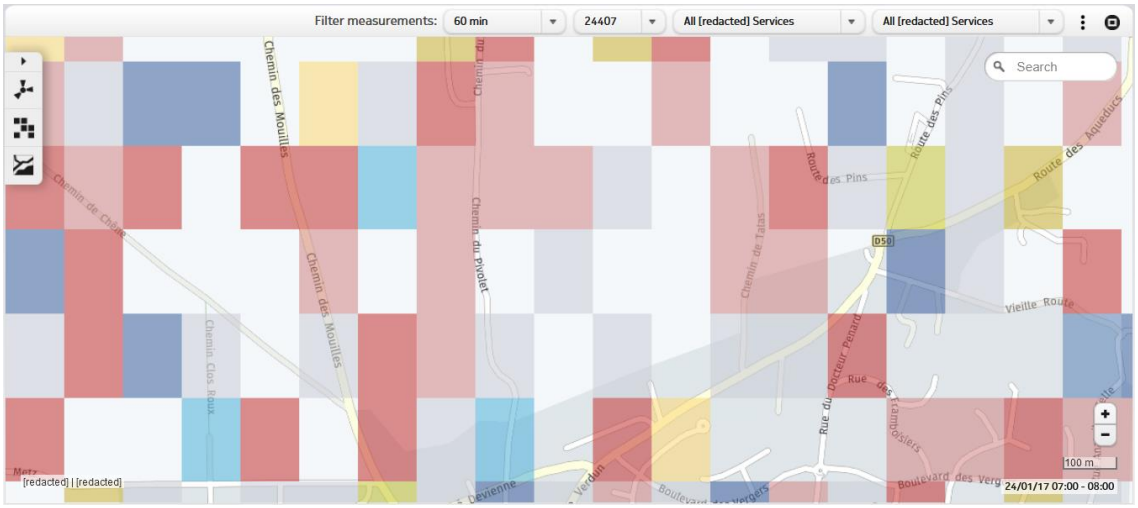

Figure 46. Geotiles shown with a guard value and a value-suppressing uncertainty palette applied. The red geotiles that fall below the guard value are given a less saturated color. Other geotiles that fall below the guard value are given a light grey hue.

5. Discussion

In this chapter I discuss the key findings that rose from analyzing the vis product's design work and Munzner's concepts' adequacy for the task.

5.1 The vis product

The main issues of the vis product's design work were about the disconnect of between user context and the designers. This manifested somewhat in the pre-validation of the user context definition as lack of in-depth user observation field studies, and especially as lack of post-validation of implemented designs through field studies or even anecdotal evidence from actual users. On the other hand, the inner idiom and implementation parts of the vis validation were quite extensive through peer-reviewing the designs before implementation, gathering regular feedback from proxy-users and domain specialists, and performance testing the implementations. However, these inner levels of design are subordinate to the decisions and validation made at design levels of domain situation and data and task abstractions. That is, this seems to be an unfortunate case of validating that the designs do things right, and not so much that the designs do the right things.

To improve the validation at domain situation and data and task abstraction levels, the designers would need closer contact with the target users. This would require prioritization from the business organization to provide time and resources for such activity. The same closer contact would allow for more detailed definition and pre-validation of the domain situation as the starting point of the design, and post-validation of the de-sign in the wild.

There were several trade-offs with many fundamental design choices such as modifiable areal unit problem (MAUP), issues with discretized color map and the limited time duration for aggregated data. These were for most part driven by the nature of the data, which in turn was dictated by the pre-existing data processing functionality. The design work could not alter this data collection functionality, since the prime motivation for creating the vis product was the fact that there was this pre-existing, highly performant data processing functionality.

5.2 Munzner's framework

It is evident from chapters 4.1 and 4.2 that Munzner's framework provides a great deal of theory by which to analyze vis design. Especially the concepts of validating designs proved useful in analyzing the strengths and weaknesses of the performed validations. Such structured thinking about the cascading nature of the different design levels would likely be useful in justifying the business needs for investing in pre-validation of designs.

In chapter 4.4 I discussed two areas where the theoretical foundation was less adequate: interactions where user's focus shifts between aspects of the vis, and data

uncertainty. In the same chapter I also suggested approaches for addressing these topics in Munzner's framework.

For interactions I proposed using graphs for describing how the user's workflow takes them from idiom to idiom as they carry out different tasks that the vis serves. I presented an example of such a graph where the graph nodes represent data abstractions and graph links represent task abstractions that move user's focus from one aspect of the data to the next. A graph appears quite an effective solution for describing the web of interactions and idioms of a vis, and I argue that presenting a more thought-out suggestion for such graph among the rest of Munzner's theory base would allow more extensive and detailed analysis of complex interactive vis tools.

Data uncertainty was tackled in the vis product by filtering out aggregations with small sample counts with guard values. It was a significant design choice because it mitigates the impact of a potentially significant issue in the vis while introducing some problems itself. The rationale for the design choice of guard values has no support from Munzner's concepts one way or the other. Considering the impacts of data uncertainty, it is quite surprising that it is not addressed among Munzner's concepts. Vis theory concerning uncertainty appears to be surprisingly scarce.

The field of statistical science, from which vis partially descends, is firmly rooted on methods for analyzing probability [Kalbfleisch, 2012], and many quintessential statistical visualizations such as the bell curve can be understood as illustrating probability distributions in data. Probability is a concept of for quantifying and managing uncertainty. When describing probability, the classical statistical probability distribution visualizations also indirectly describe uncertainty. Directly addressing uncertainty is done surprisingly little in the field of vis theory, considering how unaddressed uncertainty in data can have big negative impact on vis effectiveness. The value-suppressing uncertainty palettes (VSUPs) proposed by Correll, Moritz and Heer [2018] is promising work in that regard, and it would be welcome to see more research on vis idioms for addressing data uncertainty. In the world of ever-increasing quantity of data, and increasing usage of data in decision making, it would seem useful and moral to be explicit about the data's uncertainty. As such idioms and the theory for visualizing uncertainty would seem a useful addition to Munzner's concepts.

6. Conclusions

This thesis set out to apply the wide-reaching concepts in Tamara Munzner's book *Visualization Analysis and Design* on an existing, commercial vis product. Through this journey it became evident that especially the concept of nested design levels helps analyze the design work and identify pain points in it. The analysis indicated strongly that the design decisions at the level of domain situation (users' problems in their own terms) and data and task abstraction (framing the users' problems as visualization tasks that could be tackled with designs) should have seen more extensive validation to ensure the effectiveness of the designs. As it stands, it remains at question what shortcomings there might have been in the designs, and what could be done to correct them. These could guide improving the current product and be professional learnings for the designers and other stakeholders.

Analysis of the implemented vis designs revealed the difficulty of analyzing the entirety of an interactive vis that uses multiple vis idioms that display multiple views on multiple kinds of data and provides multitude of interactions for users to navigate their focus through them. Munzner's concepts lend themselves to deep analysis of one idiom at a time, but this piecemeal approach doesn't provide a complete view that would enable analyzing interaction flows through the whole vis in complex tasks where users are expected to leverage many vis idioms in often nonlinear sequences to derive insights from the data. I presented an attempt at visualizing such vis and interaction system as a graph, but for more usefulness it might be in order to create a more comprehensive taxonomy for such vis-interaction graphs. Such taxonomy could help guide design work with interactions such as when the user derives some insight from one part of the vis and uses that as a starting point for investigation of some other part of the vis (inputs and outputs) and when the user may have to return to an already visited part of the vis with newfound insights from elsewhere in the vis (cyclical interactions). Such taxonomy would seem to extend the existing Munzner's taxonomies so that such vis-interaction networks could more readily be analyzed and validated as solutions for complex vis tasks.

References

Card, Stuart K., Mackinlay, Jock D. and Shneiderman, Ben. 1999. Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann Publishers.

Correll, Michael, Moritz, Dominik, and Heer, Jeffrey. 2018. Value-suppressing uncertainty palettes. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems 1-11.

Erdogmus, Hakan, Medvidović, Nenad and Paulisch, Frances. 2018. 50 Years of Software Engineering. IEEE Software 35, 5, 20-24.

Friendly, Michael. 2008. A Brief History of Data Visualization. In: Chen, Chunhouh, Härdle, Wolfgang and Unwin, Antony (eds.), Handbook of Computational Statistics: Data Visualization. Springer Handbooks. 15-56.

Kalbfleisch, James G. 2012. Probability and statistical inference. Springer Science & Business Media.

Munzner, Tamara. 2015. Visualization Analysis and Design. CRC Press.

Niemeyer, Gustavo. 2008. geohash.org is public! (*http://blog.labix.org/#post-85* : 5 March 2008); archived at Wayback Machine (*https://web.archive.org*) > *http://blog.labix.org/* > 5 March 2008; citing a capture dated 1 September 2021.

Ott, Florian, and Koch, Michael. 2019. Exploring interactive information radiators – A longitudinal real-world case study. Mensch und Computer 2019 - Workshopband. Gesellschaft für Informatik.

Playfair, William. 1786. Commercial and Political Atlas: Representing, by Copper-Plate Charts, the Progress of the Commerce, Revenues, Expenditure, and Debts of England, during the Whole of the Eighteenth Century. Corry, London, England. Re-published in Wainer, H. and Spence, I. (eds.), The Commercial and Political Atlas and Statistical Breviary, 2005, Cambridge University Press.

Shneiderman, Ben. 2003. The eyes have it: A task by data type taxonomy for information visualizations. In: Bederson, Benjamin B. and Shneiderman, Ben (eds.): The craft of information visualization. Morgan Kaufmann. 364-371.

Sillanpää, Heli and Koivuniemi, Ari. 2019. Personal communication, June 19.

Tufte, E. R. 1983. The Visual Display of Quantitative Information. Graphics Press.

Tufte, E. R. 1990. Envisioning Information. Graphics Press.

Tufte, E. R. 1997. Visual Explanations: Images and Quantities, Evidence and Narrative. Graphics Press.

Ware, Colin. 2013. Information Visualization: Perception for Design. Morgan Kaufmann Publishers, 3rd edition.

Wong, David W.S. 2004. The modifiable areal unit problem (MAUP). In: Janelle, Donald G., Warf, Barney and Hansen, Kathy (eds.), WorldMinds: Geographical perspectives on 100 problems. Springer, Dordrecht. 571-575.