Rethinking Model Representation - A Taxonomy of Advanced Information Visualization in Conceptual Modeling

Giuliano De Carlo, Philip Langer and Dominik Bork

To appear in:

41st International Conference on Conceptual Modeling (ER'22)

Final version available soon:

www.model-engineering.info

Rethinking Model Representation - A Taxonomy of Advanced Information Visualization in Conceptual Modeling

Giuliano De Carlo[1], Philip Langer[2], and Dominik Bork ✉[1] [0000−0001−8259−2297]

[1] TU Wien, Business Informatics Group, Favoritenstrasse 9-11, 1040 Vienna, Austria
[2] Eclipsesource, Vienna, Austria
dominik.bork@tuwien.ac.at

Abstract. Conceptual modeling is an integral part of computer science research and is widely adopted in industrial practices, e.g., business process and enterprise architecture management. Providing adequate and usable modeling tools is necessary for the efficient adoption of modeling languages. Meta-modeling platforms provide a rich set of functionalities and are mature in realizing state-of-the-art modeling tools. However, despite their maturity and stability, most of these platforms did not yet leverage the full extent of functionalities and the ease of exploitation and integration enabled by web technologies. Current web technologies now enable much richer, advanced opportunities for visualizing and interacting with conceptual models. However, a structured and comprehensive overview of possible information visualization techniques linked to conceptual models and modeling tools is lacking. This paper aims to fill this gap by presenting a taxonomy of advanced information visualization, albeit its generic nature, applicable to conceptual modeling. We believe this taxonomy greatly benefits researchers by providing a standard frame to position their works and for method and tool engineers to spark innovation.

Keywords: Information visualization · Human Computer Interaction · Taxonomy · Conceptual modeling · Modeling tools · Notation.

1 Introduction

Technology usage forms an essential part of our private and professional lives. Having access to the right tools and the knowledge to use them correctly can save time and effort. The connection between the user of a tool and the tool itself is usually its user interface and the supported interactions that come with it. While the functionalities of a tool also play a significant role, without a graphical user interface that is well designed, tools are often labeled as not very useful for a user [26]. This is especially important in the field of conceptual modeling, where information visualization makes up a central aspect that directly influences the comprehensiveness of models and the usability and ease of use of modeling tools [9,23]. Tool development is therefore denoted as an essential part of enterprise modeling [23] and modeling business information systems [9] research. However, past research primarily focused on the development of new and the evaluation [4,22] and improvement of existing modeling languages [5].

Today, a wide range of different modeling tools are available. Most of these tools are mature and established applications that have been actively worked on over a relatively long period. Because of their age, their functionalities are often built on older technology stacks, i.e., not compatible with state-of-the-art platforms built on web technologies. Although the results produced with such tools are still unsurpassed, the functionality, especially concerning the user interfaces and the information visualization, often lacks advanced techniques like zooming. The usage of such techniques, we believe, could speed up the model development process. It could also improve usability and ease of use of the tools and comprehension of conceptual models by humans. Web technologies have been heavily used and improved over the previous years and offer a wide range of great functionalities, which is why they are the perfect fit to develop such advanced techniques. Compared to platforms used in most traditional modeling tools, web technologies provide a future-proof, feature-rich, robust, and efficient foundation for state-of-the-art visualization and interaction techniques.

Only very few research can be found focusing user interface design for modeling tools [27] and visualization techniques used in conceptual modeling have barely evolved in the last years [14,13]. A structured and comprehensive overview of information visualization techniques with an emphasis on conceptual modeling and modeling tools is lacking. This paper aims to fill this gap by presenting a generic taxonomy of advanced information visualization that is also applicable to conceptual modeling. According to Tory and Moeller [29], visualization taxonomies can: (i) *guide people* outside the core community by classifying existing works and pointing them to possible, current visualizations, and (ii) *guide research* by enabling researchers to position their contributions within a well-defined classification scheme, and by establishing a structured foundation for progressing the field in different dimensions. We believe that our advanced visualization taxonomy for conceptual model representation will activate a rethinking in the conceptual modeling and modeling tool communities. This rethinking might question state of the art in conceptual model representation and, hopefully, steer the focus of research toward novel and advanced visualizations. We also demonstrate the broad spectrum of advanced visualization techniques with the proposed taxonomy, which have not yet found wide adoption in current modeling tools. With the flexibility of web technologies, introducing them in modern tools becomes increasingly accessible and supports users to more efficiently comprehend and interact with conceptual models. Thus, we hope that the proposed taxonomy sparks innovation in future modeling tool development.

In the remainder of this paper, Section 2 briefly reports on related information visualization taxonomies. The applied research method is then presented in Section 3. The taxonomy for advanced information visualization in conceptual modeling is presented in Section 4 and consecutively evaluated in Section 5. Eventually, Section 7 concludes the paper and provides some directions for future research.

2 Related Approaches

Information visualization is a long-lasting topic in academia. We, therefore, first look at existing taxonomies that could provide valuable input for the development of our tax-

onomy. Although we could not find an existing taxonomy that perfectly fits our scope, the subsequently described works partly align with our goal. We briefly introduce these taxonomies' core dimensions and categories before excerpting the relevant ones for our scope. Shneiderman [24] proposes a task by data type taxonomy with seven data types for applications with advanced graphical user interfaces. These tasks are: *Overview*, *Zoom*, *Filter*, *Details-on-demand*, *Relate*, *History*, and *Extract*. The data-types are: *1-dimensional*, *2-dimensional*, *3-dimensional*, *temporal*, *multi-dimensional*, *tree*, and *network*. The relevant tasks within the scope of our study are mainly Overview, Zoom, Filter, and Details-on-demand. Moreover, shapes and forms of conceptual models can be classified as 2-dimensional concerning the data type. Silva and Catarci [25] categorize temporal-data features by *visualization* and *interaction* features. Visualization features describe visual techniques of a system, as in their example, Snapshot View or Multiple Calendars. Interaction features are categorized very similarly to Shneiderman's categories mentioned above.

Tory et al. [29] categorize visualization techniques based on their design model instead of their data. The authors propose to categorize design models into two higher level groups: *discrete* and *continuous*. Continuous models assume that data can be interpolated, and discrete models assume that they cannot. Data can often be visualized in multiple ways, and therefore it is possible to present the same data with continuous models and discrete models.

Cockburn et al. [7] categorize graphical user interfaces into four categories: *overview-plus-detail*, *zooming*, *focus-plus-context*, and *cue-based*. Overview-plus-detail represents the spatial separation of information. It splits up information into two separate views: overview-view and detail-view. Zooming represents the temporal separation of information. It allows magnification and demagnification of information. Focus-plus-context seamlessly combines a focused representation of information within its context. Cue-based techniques change how an object is displayed and rendered and are often combined with search criteria or off-screen elements.

3 Taxonomy Development Research Method

Nickerson et al. [20] propose a development method to create taxonomies for information systems. They define a taxonomy as a set of n dimensions $D_i(i = 1, ..., n)$ each consisting of $k_i(k_i \geqslant 2)$ *mutually exclusive* and *collectively exhaustive* characteristics $C_{ij}(j = 1, ..., k_i)$ such that each object under consideration has one and only one C_{ij} for each D_i. To create a valid taxonomy, Nickerson et al. propose an iterative approach that is applied until all ending conditions are met. One iteration can either consist of an *empirical-to-conceptual* step, or a *conceptual-to-empirical* step. Which one to choose depends on the researcher's knowledge and the available data. An empirical-to-conceptual iteration should be chosen when there are many objects available, and the researcher is familiar with them. It consists of looking at these objects and identifying characteristics based on their qualities. A conceptual-to-empirical iteration should be chosen when there are few objects and the researcher has a broad understanding and knowledge base of the relevant domain. Instead of primarily looking at objects, the researcher will identify characteristics merely based on her knowledge.

3.1 Problem Identification and Motivation

Kundisch et al. [19] recently updated the taxonomy development method proposed by Nickerson et al. They argue that most past taxonomies possess inconsistent adoption of existing methods and a non-transparent reporting of relevant design decisions. To overcome this limitation, they present an extended taxonomy design process (ETDP) and give examples of well-written taxonomies for each step in their process. The additional steps in their ETDP focus mainly on problem identification, motivation, and taxonomy evaluation. More accurately, they add three initial steps which should be conducted before the taxonomy is designed and developed. These steps consist of specifying: $i)$ the *observed phenomenon*, $ii)$ the *target user group(s)*, and $iii)$ the *intended purpose* of the taxonomy, which also influences the *ex-post* evaluation of the taxonomy.

The phenomena observed in this research are visualization and interaction features applicable to conceptual modeling. We are interested in concrete examples and theoretical concepts of features that allow users to modify underlying data by utilizing graphical user interface interaction methods. The target user groups are conceptual modeling researchers, method engineers, and developers of modeling tools interested in realizing advanced model visualization and interaction features. Our taxonomy is supposed to help identify defining characteristics of such features and provide aid during the conceptualization and integration of them into new methods or tools. Furthermore, our taxonomy should give a basic understanding of the opportunities and limitations of the existing features, which should establish a foundation for designing new features.

3.2 Solution Objectives

Two essential qualities of a valid taxonomy were already mentioned: mutually exclusiveness and collectively exhaustiveness. This means that every object has to have precisely one characteristic in each taxonomy dimension. These two qualities form two of ten objective ending conditions (cf. [19,20]), which, together with five subjective ending conditions, are used to determine when a taxonomy is considered complete. Consequently, these ending conditions need to be applied during the iterative application of either an *empirical-to-conceptual* or *conceptual-to-empirical* step until the ending conditions are met.

Subjective and objective ending conditions are essential to determine when the iterative process can be stopped, and the taxonomy holds enough characteristics to classify the phenomenon it is supposed to describe. For this reason, they are both used as an *ex-ante* evaluation. Subjective ending conditions are, e.g., **Robustness:** Do the dimensions and characteristics enable differentiation among objects sufficient to be of interest? And **Comprehensiveness:** Can all objects or a (random) sample of objects within the domain of interest be classified? Are all dimensions of the objects of interest identified? Objective ending conditions are, e.g., All objects or a representative sample of objects have been examined while no new dimensions or characteristics were added in the last iteration, and no dimensions or characteristics were merged or split in the previous iteration. For the sake of brevity, we refer the interested reader to the literature proposing the well-established ending conditions in great detail [20,19].

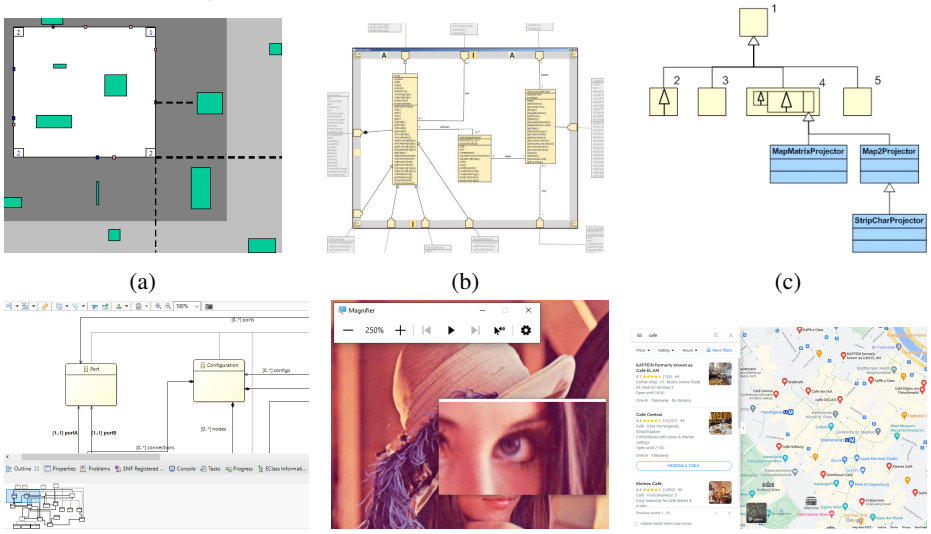

Fig. 1: Examples of survey results: **a)** Off-screen elements are shown with their orthographic direction, the color shows the distance, and markers indicate the number of off-screen elements in that direction [30]; **b)** Off-screen elements are represented as proxies on the border of the focused view [10].; **c)** Onion graph visualization by [18]. Blue elements are focused whereas yellow elements are not focused. Nodes 3 and 5 represent individual classes while Nodes 2 and 4 represent multiple generalizations; **d)** A minimap in the Eclipse IDE. The main view shows only a part of a class diagram. The minimap in the bottom left shows the entire diagram with a much smaller zoom factor. The blue square provides orientation inside the main view; **e)** The Windows 10 Magnifier app magnifies parts of the view while hiding others; **f)** Google Maps' search results at the left act as a proxy. Clicking on one, zooms and scrolls the map to the corresponding position.

3.3 Design and Development

The first iteration for developing this taxonomy was a conceptual-to-empirical one. We thereby looked at taxonomies and related literature (see this paper's appendix[1]). Multiple empirical-to-conceptual iterations followed this. Initially, we looked at features of typical and widely used software applications that possess interactive graphical user interfaces. These software tools receive constant feedback and are being maintained and improved by leading companies in their field. We investigated 46 tools and platforms; among them were, e.g., Google Maps, Microsoft PowerPoint, or JetBrains IntelliJ IDEA (e.g., see Fig. 1 d, e, and f). A full list of all investigated tools is given in[1].

The derived dimensions and characteristics were then re-evaluated with another empirical-to-conceptual iteration that considered past literature's visualization and interaction features. This iteration thus further incorporated conceptual designs of features and provided more insights about the reasoning behind design decisions and technical conditions instead of focusing solely on already realized features from a user perspective as in the first iteration. Examples identified in the second iteration are City Lights [30], EdgeRadar [15], and Onion graphs [18] (see Fig. 1).

[1] https://www.dropbox.com/s/1oudgdyb0xqi6ns/taxonomy_appendix_0_3.pdf?dl=0

3.4 Demonstration and Evaluation

While it is impossible to consider all existing features of today's tools and literature, the sample of features was expanded until all subjective and objective ending conditions were met. It is to note that, ideally, this taxonomy should only be used to categorize concrete examples of features in the end. We realized that conceptual designs of features could often be interpreted and implemented in many different ways during the development. E.g., the concept of a magnifying glass feature can be implemented in a separate and independent view or by magnifying the current view. In the first case, it would be classified as an overview-plus-detail interface, but in the second case, it would be classified under focus-plus-context. When classifying conceptual designs of features that have not been implemented yet, one must be aware that it may include a subjective bias. Often, it is not immediately obvious how such features operate, which is why it is even more important to describe them accurately.

After the description of the first steps of the extended taxonomy design process in this section, we will, in the following section, present the final taxonomy for advanced information visualization in conceptual modeling. Eventually, the comprehensive evaluation of the final taxonomy is reported in Section 5.

4 Taxonomy

For our taxonomy, three meta-characteristics emerged which provide the structure of the following sections: **Presentation**, **Interaction**, and **Data**.

4.1 Presentation

This dimension mainly describes if and how a feature utilizes one or multiple views, it describes the dependency between views, and how views represent information.

Presentation/Interface Type: The first dimension mainly describes how a feature uses the available space, represents information to the user, and generally, how a user can interact with it. This dimension consists of four categories, which are based on Cockburn et al.'s work [7]: ***overview-plus-detail***, ***zooming***, ***focus-plus-context***, and ***cue-based***. This categorization is thus not new and can be found recurring when browsing the information visualization literature.

Overview-plus-detail: This interface scheme is used in many applications nowadays. It splits the information space into two physically separated views; one shows information at an overview level, and the other shows similar or even the same information in greater detail. Although they are physically separated, the two views do semantically depend on each other, and actions in one view are usually immediately reflected inside the other view. The essential characteristic of the dependency between both views is that they are usually not spatially dependent on each other. If one or even both views were to be moved to a different location, no issues would arise.

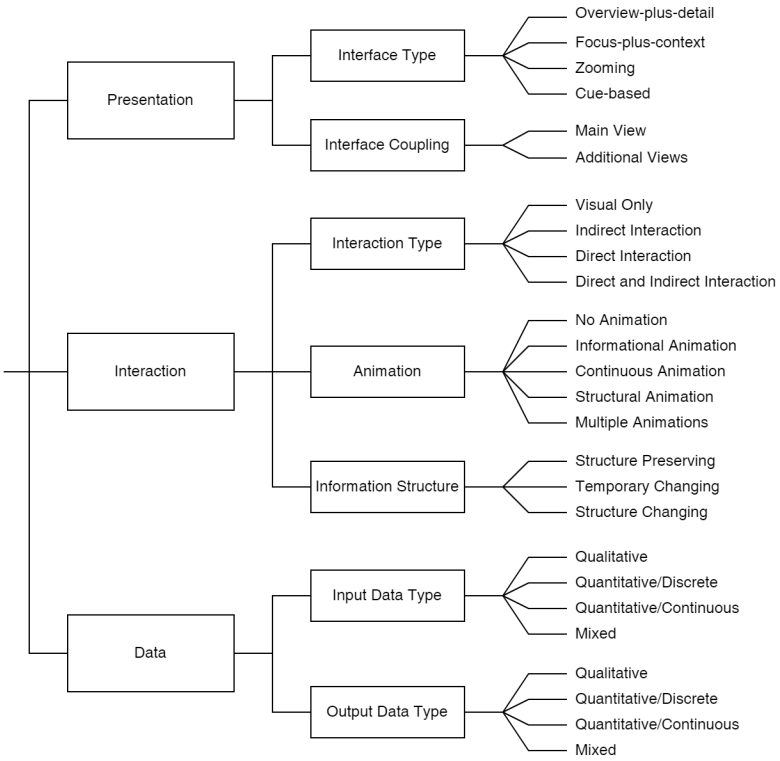

Fig. 2: Advanced Information Visualization in Conceptual Modeling Taxonomy.

An overview-plus-detail interface usually has two primary purposes. Firstly, it should give the user a better feeling about what subset of information they are currently looking at in relation to the entire information space. Secondly, they should give the user an easy way of navigating the information space by letting the user interact with the overview interface. Usually, they operate on the x- and y-axis and utilize interaction methods such as panning or scrolling. Operation on the z-axis is seen less often and mostly overlaps with the zooming category introduced later.

A good example is shown in Figure 1 d). An excerpt of a UML diagram is shown in detail on the center view, while in the bottom left corner, an overview is displayed, which shows the entire diagram. Actions like panning or zooming inside one view are directly reflected inside the other view. The overview interface does not always have to show the same type of information as the detail-view; it may also provide a completely different type of data, e.g., spatial data. An example of this is the scrollbar. Scrollbars can be seen as the overview interface that gives one-dimensional information about what the other view displays in relation to the entire information space. E.g., the Sublime Text 3 editor widened the vertical scrollbar to show spatial information and additional information concerning syntax highlighting. Another good example is Microsoft PowerPoint, where the overview view on the left shows a miniature version of the slide that is currently displayed inside the detail view along with the following and preceding slides.

Focus-plus-context: This scheme lets a user see specific parts of the information space in more or full detail while also getting an overview of the information around it (distortion of the information space). Unlike overview-plus-detail, both parts (overview and detail) are displayed *inside the same view* which is often accomplished by distorting the information space to fit the user's needs. The focus of the information that the user is interested in is shown in greater detail. At the same time, information around it, the context, is preserved and made visible to the user but in lesser detail.

The level of distortion that is applied differs from implementation to implementation. It goes from no/infinite distortion, e.g., by using the Windows 10 Magnifier app (Figure 1 e), to distortion that affects the entire view, as seen, e.g., in Google Street View. The distortion aspect is an essential feature of focus-plus-context interfaces, differentiating this category's components from others. Without distortion, as with the Windows 10 Magnifier app, a feature can often be categorized as a basic zooming feature or an overview-plus-detail feature instead.

An advantage of having only one view instead of two or more is that the user does not have to switch between multiple views and can rather keep focusing on the same view. In a field study conducted by Baudisch et al. [2], who compared overview-plus-detail, focus-plus-context, and zooming-plus-panning, all chosen tasks could be performed faster on the focus-plus-context interface by a margin of 21-36%. They attribute the differences to the context switches that do not have to be made on a focus-plus-context interface and the consistent scale that the focus-plus-context interface offers.

Zooming: Zooming utilizes the temporal separation of the information space. It is similar to overview-plus-detail with the difference that only one view is provided instead of two or more. Another difference is that the user has to utilize interaction methods (e.g., button click, CTRL+mousewheel, or CTRL+"+"), mainly the zooming method, to change the size in which information is displayed. This represents basic zooming which can be further advanced by combining it with other techniques such as fisheye zoom [12,1,21] or semantic zoom [11,10].

A precondition to making zoomable interfaces possible is to have information that can be magnified and de-magnified. The magnification process can be categorized into *continuous* and *discrete zooming*. Continuous zooming occurs when the subject does not have a countable amount of zoom levels. The simplest form of magnification is to increase the subject's size. This can be done on every subject with some visual representation. Since there is no clear separation of zoom levels, and the subject can theoretically be rendered in any size, this can be considered continuous zooming. An example of discrete zooming takes place in Google Maps (see Figure 1 f). With Google Maps, users can change the presented level of detail by zooming in and out of predefined zoom levels (i.e., discrete). Notably, during the zooming in one zoom level, Google Maps applies continuous zooming by increasing/decreasing the size of the presented elements.

Cue-based: Cue-based techniques often show alternative graphical representations of objects on the stage to give cues that lead to other information in the information space. These alternative representations, often in the form of simple labels, can then be used to, e.g., navigate to the actual object or notify the user that the object exists. Examples of cue-based techniques are the visualization of off-screen objects with interaction

functionality as seen in Figures 1 a) and b). Cue-based techniques are also often used in combination with search criteria. An example is the clickable search results presented to the users of Google Maps (see Figure 1 f). Clicking on them pans and zooms the map to the position of the search result.

Presentation/Interface Coupling: This category distinguishes features that require **additional views** from those operating inside the **main view**. This is an essential aspect as it impacts how a user uses the feature and how to implement a feature. From an implementation perspective, having to add another view can become complicated because, depending on the size and significance of the implemented feature, new space has to be found on the user interface, and, instead of managing just one view, multiple views have to be managed and kept consistent [3]. From the user's perspective, additional views mean multiple separated points of interest. Switching between them demands a focus switch in our brains, which requires a mental effort that can quickly become exhausting.

As Baudisch et al. and Hornbak et al. [2,16] showed, tasks could be performed faster on focus-plus-context interfaces (one-view) than on overview-plus-detail (two views). Contrarily, Thabet et al. [28] analyzed the positive effect of automated consistency management in multiple views.

4.2 Interaction

This dimension focuses on the characteristics of a feature that the user directly interacts with. The attributes in this dimension significantly impact how a feature is being used by the target group and, therefore, its usability. It should give insight into critical user-related aspects of a feature and help make design-related decisions. Unlike the first dimension ***Presentation***, which is mainly based on previous literature, this dimension was conducted by looking at different tools and our experiences made during software development.

Interaction/Interaction Type: This characterization distinguishes the kinds of interaction that a feature offers. While this category can be further extended by going more into detail and considering all kinds of interaction types and events, this taxonomy remains on a more abstract level with only four main categorizations: ***visual only***, ***indirect interaction***, ***direct interaction***, and ***direct and indirect interaction*** to also account for the subjective goal of *Conciseness* [20].

Visual Only: Features which do not give a user the ability to interact with it, i.e., features that only add visual benefits. An example of a visual-only feature is a grid system that helps users position elements but does not provide direct interaction possibilities.

Direct Interaction: Features that add new interaction possibilities. They are usually intentionally performed by a user, and their primary purpose is to manipulate feature-specific data directly. An example of this is the *Peek Definition* feature in conventional

IDEs like IntelliJ and VS Code, where programmers can directly interact with a function/method inside the code by triggering a specific context menu on that feature. This interaction process is specific to the feature, and the context menu would have no use if this feature did not exist.

Indirect Interaction: Actions of a feature with indirect interactions can usually be triggered by performing interactions that are not part of the feature itself. Actions controlled by indirect interactions are usually triggered either concurrently alongside actions of other features or as a side effect of such. An example of an indirect interaction feature is the ruler visible in many text or diagram editors that provides basic functionality to related elements on the stage, e.g., a spatial position.

Direct and Indirect Interaction: Some features utilize direct and indirect interactions. An example of such would be the basic scrollbar commonly known from text editors like Notepad and Word. A direct interaction would be to click and move the scrollbar. An indirect interaction would be to move the position of the current viewport by different means (e.g., with the mouse wheel). This would indirectly automatically trigger the scrollbar to move as well.

Interaction/Animation: This categorization gives an overview of the types of animations a feature uses. Animations are essential in visualizing even complex changes to the users intuitively and understandably. The proper use of animations prevents users from getting confused about these changes and increases their sense of orientation (cf. [6]). Here, it is crucial to understand the difference between a separate animation played alongside a feature and its functionality. In this category, animations are considered visual techniques that physically move graphical objects and are triggered by the user but not directly controlled by them. Most of the time, they are played right after the user's interaction has finished.

No Animation: Features that do not utilize animations in any form, like adjusting the physical positioning of visual objects through basic scrollbars.

Informational Animation: Animations, which give the user additional information (often in the form of text) but do not directly interfere with elements on the stage. An example of this is given by Igarashi and Hinckley in [17]: "..., when the user presses the mouse button, a pink slider appears." Another example would be a small label that is transitioned in and out during a zooming interaction, showing the current zoom level. This could be done with discrete zoom levels (as is done by yEd[2]) or with continuous zooming actions which show the current zoom level relative to a base value.

Continuous Animation: Animations, that take existing elements on the stage and change, in a continuous process, the way they are represented. The characteristic of this kind of animation is that no additional information is added or removed. Examples are a simple magnification of an element (see Figure 1 e) or by clicking on a proxy element (e.g., $+/-$ symbols).

Structural Animation: This type of animation is used when the structure of elements or the stage is changed. Unlike continuous animations, new information is added, old is removed, or existing is changed. This new information could, e.g., be entire elements or just properties of elements. An example of such an animation is the already mentioned *Peek Definition* feature.

Multiple Animations: Some features utilize multiple animations independently from each other in different areas of an application. For such cases, we decided to include a *Multiple Animations* categorization, which consists of features that utilize a combination of informational, continuous, and/or structural animations. An example would be the already mentioned feature Speed-dependent automatic zooming [17]. It does utilize not only an informational animation in the form of a pink slider but also a continuous animation: "When the user releases the mouse button, an animated transition gradually returns the document to the original base scale." [17, p. 142]

Interaction/Information Structure: Many tools, especially modeling tools, give the user the option to adjust the structure of the information space. The user can personalize the information space by adjusting the position of objects inside it. Positions are saved, and objects are positioned at the previously defined position upon re-opening a file. Not having to recreate the mental map of the information space every time a file is opened saves a lot of the user's time and effort. Because of that, this characteristic is precious and should not be carelessly taken away. The larger an information space is, the more time a user requires to get used to a new structure, and therefore, the more critical it is to keep the structure intact.

Nevertheless, preserving the structure is not always easily possible. E.g., problems can arise when dealing with the typical "expand/collapse" functionality. An expanded element may become too large, pushing other elements out of the way. Another common cause for a change in the information structure is the automatic process of rearranging elements. Some modeling tools offer a "center all elements" functionality, which automatically adjusts and centers the position of all elements. Such functionality can be a dangerous game, and finding a good algorithm that prevents the user from having to recreate the mental map of their workspace is a challenging task. For that reason, it is often the best solution to stay away from implementing features that frequently change the layout of a user's workspace.

Structure Preserving: This category includes all features that do not change the structure at all. The critical point of features in this category is that users do not have to recreate the mental map of their workspace. An example for such a feature is the semantic zoom in yEd[2] that adds/removes information but preserves the structure.

Structure Changing: Features in this category adjust the structure to the point that causes users to recreate their mental map. An example is the grouping feature in yEd. Unlike their semantic zooming implementation, closing or opening groups automatically adjusts the structure. When opening a group, the position of elements inside the

[2] yEd [online], `https://www.yworks.com/products/yed`, last visited: 26.04.2022

group is adjusted. When closing a group, the position of elements outside the group is changed to utilize the previously occupied space.

Temporary Changing: Features of this category temporally adjust the structure. Often, features visually change the structure, only for a limited amount of time, to execute a specific functionality (e.g., the *Peek Definition* feature).

4.3 Data

This dimension describes the feature-specific data used by a feature and can be distinguished from the tool-specific data. Just like the previous dimension, this one is also based on inspection of different tools and software development experiences. If we use the scrollbar of a PDF reader software as an example, the tool-specific data would be the PDF itself, and the feature-specific data would be the x- and y-coordinates of the viewport. The scrollbar feature does not manipulate the PDF at all. Instead, it moves the viewport to different locations by changing its coordinates. Unlike the tool-specific data, the feature-specific data is often independent of the tool itself, which helps to keep this taxonomy more abstract.

Data can be split into input data that is read by a feature to perform an action and output data modified or returned by an action of a feature. Both sub-dimensions, ***Data/Input Data Type*** and ***Data/Output Data Type***, consist of the same characteristics: *Qualitative*, *Quantitative/Discrete*, *Quantitative/Continuous*, and *Mixed*.

Qualitative data is semi-structured data, such as labels, attributes, or entire domain model elements. Quantitative data can be counted or measured and is expressed as numbers. Furthermore, quantitative data exists in two variations, i.e., *quantitative/discrete* and *quantitative/continuous*. Quantitative/discrete data is countable and can only take specific values. Quantitative/continuous data, on the other hand, is measurable and can be split into smaller parts.

Similar to the *Interaction/Animation/Multiple Animations* categorization, some features operate with multiple different sets of data. E.g., the visualization of off-screen elements (Figure 1a and 1b) work with information about position, color, and size of off-screen elements. Such features should be classified as *Mixed*. It is ideal for a feature to work with continuous data most of the time. This is because continuous data reflect user interactions more directly and responsively. It is easier to follow continuously rendered changes than discretely rendered ones. This is also reflected in [8].

5 Evaluation

In the following, we report on the application of two widely used taxonomy evaluation techniques in our ex-post evaluation, i.e., *Illustrative scenario with real-world objects* and *Illustrative scenario with existing research* [19]. We classified 33 features that were also used during the iterative steps of the taxonomy design. Features of existing commercial tools represented real-world objects while visualization features of past literature represented existing research. Table 1 exemplifies the classification of features. The complete classification can be found in the online appendix[1]. In the future, we plan

Table 1: Selected features classified with the taxonomy.

Category	Minimap[1]	Magnifying App[2]	Search Results[3]	Zooming [17]
Interface Type	overview-plus-detail	focus-plus-context	cue-based	zooming
Interface Coupling	additional views	main view	additional views	main view
Interaction Type	direct + indirect	direct	direct	indirect
Animations	no animation	continuous	continuous	multiple (continuous + informational)
Information Structure	structure preserving	structure preserving	structure preserving	temporary changing
Input Data Type	quantitative/discrete	quantitative/discrete	qualitative	quantitative/continuous
Output Data Type	quantitative/discrete	quantitative/discrete	qualitative	quantitative/continuous

[1] Eclipse IDE [2] Windows 10 [3] JetBrains IntelliJ IDEA, Google Maps

to use this taxonomy to classify academic prototypes presented at tool and demo tracks and also have external tool developers use and evaluate it. In a first step, we classified two prototypes featuring advanced interaction techniques in a state-of-the-art modeling environment (Eclipse GLSP) that we developed in a separate work. The taxonomy greatly supported the design of a concept for these prototypes.

6 Challenges & Limitations

The main challenge during the conceptualization of the characteristics for this taxonomy was to keep the mutual exclusivity intact. Some features were too complex or had too many functionalities to only have one characteristic in a single dimension. For that reason, we created extra categories that combine multiple other characteristics, e.g., *Interaction/Animation/Multiple Animations*. A feature with animations of multiple types can then be placed under this category. Here, in some cases it may also help to split up a feature into sub-features and categorize each sub-feature individually.

During some of the empirical-to-conceptual iterations, it was not always clear under which of Cockburn et al.'s [7] interface types a feature should be placed. Some features have characteristics of multiple interface types and could therefore be assigned to multiple categories. We felt that especially the *overview-plus-detail* and *focus-plus-context* category could often be used interchangeably. To mitigate this issue, we added the additional constraint of spatial dependency to the *focus-plus-context* type. We further limited our scope mainly to existing and widely used tools. While this provides a good foundation for categorization of established features, it lacks relevance with respect to prototypical tools originating from research.

Another limitation might be the number of characteristics of this taxonomy. There might be a need for additional characteristics in the dimension *Interaction*. Similarly, to how some user interactions are followed by animations, they could also trigger other functionalities such as, e.g., audio playback. This is just a theoretical example as we could not find features that rely on audio playback. According to Nickerson et al. [20] there is no agreed upon maximum for what represents an appropriate number of dimensions or characteristics. Nevertheless, with their proposed subjective ending condition *concise*, they suggest having a limited number to keep the taxonomy comprehensive and easy to apply. Besides *Interaction*, we also considered the characteristics of the *Pre-*

sentation/Interface Type dimension problematic at times. As already mentioned above, classification in this dimension was not always easy, and, although we were able to place all our observed features into one of the proposed interface types, we did get the impression that this space is not fully covered yet and new types will come up in the future, leading to potential extensions of the taxonomy.

7 Conclusions

We presented a taxonomy of advanced information visualization with applications to conceptual modeling. The taxonomy structures visualization features along the three higher-level dimensions *Presentation*, *Interface*, and *Data* and further seven lower-level dimensions, each of which with specific characteristics. We evaluated the taxonomy with ex-ante and ex-post taxonomy evaluation methods by relating it to existing research and by applying our taxonomy to real-world objects[1]. This taxonomy combines established categorizations with new and original ones. From a scientific viewpoint, we contribute a novel taxonomy with an expressivity in classifying visualization features lacking in past literature. The presented taxonomy can facilitate a new feature's ideation and conceptualization phases from a practical viewpoint. The taxonomy can push method engineers and tool developers toward rethinking model representation and designing and implementing advanced information visualization features. We thus hope that the proposed taxonomy sparks innovation in future conceptual modeling research.

References

1. Bartram, L., Ho, A., Dill, J., Henigman, F.: The continuous zoom: A constrained fisheye technique for viewing and navigating large information spaces. In: Proceedings of the 8th annual ACM symposium on User interface and software technology. pp. 207–215 (1995)
2. Baudisch, P., Good, N., Bellotti, V., Schraedley, P.: Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming. In: SIGCHI conference on Human factors in computing systems. pp. 259–266 (2002)
3. Bork, D., Buchmann, R.A., Karagiannis, D.: Preserving multi-view consistency in diagrammatic knowledge representation. In: Knowledge Science, Engineering and Management - 8th International Conference, KSEM. pp. 177–182. Springer (2015)
4. Bork, D., Karagiannis, D., Pittl, B.: Systematic analysis and evaluation of visual conceptual modeling language notations. In: 2018 12th International Conference on Research Challenges in Information Science (RCIS). pp. 1–11. IEEE (2018)
5. Bork, D., Roelens, B.: A technique for evaluating and improving the semantic transparency of modeling language notations. Softw. Syst. Model. **20**(4), 939–963 (2021)
6. Card, S.K., Robertson, G.G., Mackinlay, J.D.: The information visualizer, an information workspace. In: Conference on Human factors in computing systems. pp. 181–186 (1991)
7. Cockburn, A., Karlson, A., Bederson, B.B.: A review of overview+ detail, zooming, and focus+ context interfaces. ACM Comput. Surv. **41**(1), 2:1–2:31 (2008)
8. Cockburn, A., Savage, J.: Comparing speed-dependent automatic zooming with traditional scroll, pan and zoom methods. In: People and Computers XVII, pp. 87–102. Springer (2004)
9. Frank, U., Strecker, S., Fettke, P., Vom Brocke, J., Becker, J., Sinz, E.: The research field modeling business information systems. Bus. Inf. Syst. Eng. **6**(1), 39–43 (2014)

10. Frisch, M., Dachselt, R.: Visualizing offscreen elements of node-link diagrams. Information Visualization **12**(2), 133–162 (2013)
11. Frisch, M., Dachselt, R., Brückmann, T.: Towards seamless semantic zooming techniques for uml diagrams. In: 4th ACM Symposium on Software Visualization. pp. 207–208 (2008)
12. Furnas, G.W.: Generalized fisheye views. Acm Sigchi Bulletin **17**(4), 16–23 (1986)
13. Gulden, J.: Recommendations for data visualizations based on gestalt patterns. In: Li, G., Yu, Y. (eds.) International Conference on Enterprise Systems. pp. 168–177 (2016)
14. Gulden, J., Reijers, H.A., Grabis, J., Sandkuhl, K.: Toward advanced visualization techniques for conceptual modeling. In: CAiSE Forum. pp. 33–40. Citeseer (2015)
15. Gustafson, S.G., Irani, P.P.: Comparing visualizations for tracking off-screen moving targets. In: Extended Abstracts on Human Factors in Computing Systems. pp. 2399–2404 (2007)
16. Hornbæk, K., Bederson, B.B., Plaisant, C.: Navigation patterns and usability of zoomable user interfaces with and without an overview. ACM Transactions on Computer-Human Interaction (TOCHI) **9**(4), 362–389 (2002)
17. Igarashi, T., Hinckley, K.: Speed-dependent automatic zooming for browsing large documents. In: ACM symposium on User interface software and technology. pp. 139–148 (2000)
18. Kagdi, H., Maletic, J.I.: Onion graphs for focus+ context views of uml class diagrams. In: Int. Workshop on Visualizing Software for Understanding and Analysis. pp. 80–87 (2007)
19. Kundisch, D., Muntermann, J., Oberländer, A.M., Rau, D., Röglinger, M., Schoormann, T., Szopinski, D.: An update for taxonomy designers: Methodological guidance from information systems research. Bus. Inf. Syst. Eng. **63** (2021)
20. Nickerson, R.C., Varshney, U., Muntermann, J.: A method for taxonomy development and its application in information systems. Eur. J. Inf. Syst. **22**(3), 336–359 (2013)
21. Reinhard, T., Meier, S., Glinz, M.: An improved fisheye zoom algorithm for visualizing and editing hierarchical models. In: Second International Workshop on Requirements Engineering Visualization (REV 2007). pp. 9–9. IEEE (2007)
22. Roelens, B., Bork, D.: An evaluation of the intuitiveness of the pga modeling language notation. In: Enterprise, Business-Process and Information Systems Modeling, pp. 395–410. Springer (2020)
23. Sandkuhl, K., Fill, H.G., Hoppenbrouwers, S., Krogstie, J., Matthes, F., Opdahl, A., Schwabe, G., Uludag, Ö., Winter, R.: From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. Bus. Inf. Syst. Eng. **60**(1), 69–80 (2018)
24. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: The craft of information visualization, pp. 364–371. Elsevier (2003)
25. Silva, S.F., Catarci, T.: Visualization of linear time-oriented data: a survey. In: First Int. Conference on web information systems engineering. vol. 1, pp. 310–319. IEEE (2000)
26. Stone, D., Jarrett, C., Woodroffe, M., Minocha, S.: User interface design and evaluation. Elsevier (2005)
27. Ternes, B., Rosenthal, K., Strecker, S.: User interface design research for modeling tools: A literature study. Enterprise Modelling and Information Systems Architectures (EMISAJ) **16**, 4–1 (2021)
28. Thabet, R., Bork, D., Boufaied, A., Lamine, E., Korbaa, O., Pingaud, H.: Risk-aware business process management using multi-view modeling: method and tool. Requir. Eng. **26**(3), 371–397 (2021)
29. Tory, M., Moller, T.: Rethinking visualization: A high-level taxonomy. In: IEEE symposium on information visualization. pp. 151–158. IEEE (2004)
30. Zellweger, P.T., Mackinlay, J.D., Good, L., Stefik, M., Baudisch, P.: City lights: contextual views in minimal space. In: CHI'03 extended abstracts on Human factors in computing systems. pp. 838–839 (2003)