Unlocking visual understanding

Genon, Nicolas; Perrouin, Gilles; Le Pallec, Xavier; Heymans, Patrick

Published in:
Conceptual Modeling - 35th International Conference, ER 2016, Proceedings

DOI:
10.1007/978-3-319-46397-1_39

Publication date:
2016

Document Version
Peer reviewed version

Link to publication

Citation for pulished version (HARVARD):
Genon, N, Perrouin, G, Le Pallec, X & Heymans, P 2016, Unlocking visual understanding: Towards effective keys for diagrams. in *Conceptual Modeling - 35th International Conference, ER 2016, Proceedings.* vol. 9974 LNCS, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9974 LNCS, Springer Verlag, pp. 505-512, 35th International Conference on Conceptual Modelling, ER 2016 held in conjunction with Workshops on AHA, MoBiD, MORE-BI, MReBA, QMMQ, SCME and WM2SP, 2016, Gifu, Japan, 14/11/16. https://doi.org/10.1007/978-3-319-46397-1_39

Unlocking Visual Understanding: Towards Effective Keys for Diagrams

Nicolas Genon[1], Gilles Perrouin[1], Xavier Le Pallec[2], and Patrick Heymans[1]

[1] PReCISE, University of Namur, Belgium
{nicolas.genon,gilles.perrouin,patrick.heymans}@unamur.be
[2] University of Lille, France
xavier.le-pallec@univ-lille1.fr

Abstract. Diagrams are (meant to be) effective communication supports to convey information to stakeholders. Being communication supports, they have to be quickly and accurately understood. To enable immediateness, many disciplines such as cartography rely on keys, which categorise diagram symbols and bind them to their meaning. Software engineering extensively relies on visual languages such as UML to communicate amongst the many stakeholders involved in information systems' life-cycle. Yet, keys are barely used in these diagrams, hindering (immediate) understanding and limiting it to language experts. We provide a disciplined approach to design effective keys, by adapting graphic semiology theory and cartographers' know-how to software diagrams. We illustrate our method on a UML class diagram. Designing effective keys raises questions about the concerns and tasks to be addressed by the diagram, and even, reveals issues about the language itself.

Keywords: key, caption, legend, diagram understandability, visual immediacy, visual effectiveness, visual modelling language.

1 Keys in Visual Languages

While understandability is a major preoccupation in Software Engineering (SE) modelling, it still lacks a precise but consensual definition. Based on these references [14, 7, 8], an understandable diagram is a diagram that provides all (and only) the pieces of information the stakeholders look for, immediately perceptible and in a way suitable to the stakeholders' questions or tasks. We propose a definition of understandability that encompasses those notions but at a more operational level. Understandability is thus a combination of *accuracy* and *speed* at which the stakeholder processes the information conveyed by the diagram. Relying on a visual modelling language does not automatically make diagrams understandable or immediate. Understandability is a quality that requires diagrams to be designed in the appropriate way.

Over time, a series of theories about modelling language visual qualities have emerged but only a handful of them goes beyond the stage of a collection of abstract guidelines. The two most complete theories so far are the Cognitive Dimensions of Notations (CDs) [5] and the Physics of Notations (PoN) [10]. While the CDs provide a large set of high-level properties that help make visual languages cognitively effective (and hence, more easily comprehensible), PoN focusses on evidence-based principles formulated to address SE visual languages in practice. However, both theories still lack a detailed procedure to apply their properties/principles. Our work contributes to make up for this lack of support by proposing a disciplined method to design effective diagramming keys.

Keys – also called legends or captions – are usually thought about to be anecdotal pieces of documentation. On the contrary, we argue that designing effective keys is a real challenge that is worth the effort. Indeed, a key designed according to the method described in this work allows *(i)* to identify the questions to which the diagram can provide answers, *(ii)* to indicate the diagram context, and *(iii)*, to show how the components of the information are visually depicted. Software engineers may consider keys unnecessary given their mastering of modelling languages. However, it would restrict diagrams to information storage artefacts only, whereas they are also communication supports to non-expert stakeholders.

While keys are present on various kinds of documents that we consult in our every-day life, it is surprising that they are most of the time missing from SE diagrams. We searched several sources (Google Scholar, Dblp) and dedicated venues in computer visualisation (VL/HCC, VIS, EuroVIS, SIGGraph, and in Journals like the IEEE Transactions on Visualization and Computer Graphics) as well as cartographic domains. The searched keywords were: *key*, *legend*, *caption*, combined to the terms *model understandability*, *diagram understandability*, *model comprehensibility*, *diagram comprehensibility*. We finally unearthed three main articles, focussing on interactive environments. Interactive environments are a sub-category of dynamic environments where visual representations contain animations and the tool hosting the diagram is equipped with functions allowing the user to directly interact on the diagram.

Dykes *et al.* elicit high-level guidelines to design keys for maps in a dynamic environment [3]. These guidelines are driven by distinct strategies governing the display of the key (e.g. embedding the key in the map itself, or revealing the information on demand). Most of their guidelines involve dynamism, which is not easily transferable to the diagrams we target. More importantly, they do not cover fundamental constituents (such as shapes and colors), which are essential for a principled approach to keys design. Tudoreanu and Kraemer's work suffer from the same limitations, with a greater focus on user interaction [17]. Interaction is also the focus of cartographers ([9, 12, 15]).

Riche *et al.* [6] present and evaluate interactive keys by mentioning fundamental notions. They demonstrate how these notions can be used to empirically assess keys in dynamic environments. In contrast, we use these notions to focus on (static) key design, thus both works are complementary.

2 What Should Be in a Key?

What elements should appear on a diagramming key and how (or where) to place these elements on the diagram are the true questions to be asked when designing a key. Our approach is based on the Semiology of Graphics (SoG), a cartography theory published in 1963 by Bertin [1]. In a nutshell, keys designed according to the SoG make the stakeholder (i.e., the reader of the visual representation) able to understand a map without any kind of prior learning (except for the notion of coordinates conveyed by the map). We pursue the same objective but for SE diagrams, independently of the visual modelling language.

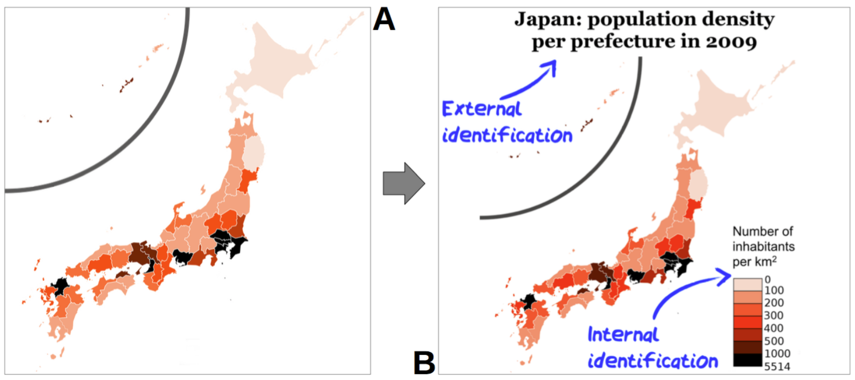
A
B
Japan: population density per prefecture in 2009
External identification
Internal identification
Number of inhabitants per km²
0
100
200
300
400
500
1000
5514

Fig. 1. On the left, a map without any key – on the right, a map with a well designed key

We introduce the elements that compose a key by referring to a (cartography) map example. Considering Fig. 1(a) on its own, the stakeholder can recognize the country under consideration (i.e., Japan). He can also identify distinct levels of colour brightness that seem to express distinct values, but nothing can be certifiably inferred about the semantics of these values. Providing a key to the map actually solves these issues. A key is the visual information added to a visual representation that allows to perform the *external* and *internal* identifications. The internal identification indicates the *components* and their *categories*. *Categories* are the distinct values depicted by visual artefacts on the representation. As depicted by Fig. 1(b), the categories are the numeric ranges associated to each level of colour brightness. In this example, all these ranges are related to a common concern – the number of inhabitants per km^2, which is called a *component*. The external identification provides information about the context (i.e., the *invariant*) of the information conveyed by the visual representation and it consists in wording (i.e., by providing a title to the representation). Referring to the example, the context of Fig 1(b) is the Japanese population density per prefecture in 2009.

Hereafter, we introduce the hypothesis that structures the relationships among the three core notions. It states that the information conveyed by a visual representation consists of a set of pieces of information. All pieces of information (aka., tuples) are specific combinations of *categories* taken from the same set of *components*. The *components* are defined for the entire visual representation in the context of the given *invariant* and each combination has to be composed of at least one *category* of each *component*. A *component* is defined by several properties: a **name** which is a string denoting the concern that is modelled by the component, a **length**, which is an integer indicating the number of categories defined for this component, and a **organizational level**, which is a value from the set {qualitative, ordered, quantitative} that means that the categories of this component are not naturally ordered (i.e., qualitative), or that there exists a natural order (i.e., ordered), or that it is ordered and the distance between two categories is significant (i.e, quantitative). The *categories* are described by exhaustively eliciting their values.

Referring to the Japan map, we can now define the properties of the component and its categories. The component is labelled *inhabitants / km*^{2}, has a length of *seven*, and its organizational level is *ordered*. The categories for this component are the ranges: [0-99], [100-199], [200-299], [300-399], [400-499], [500-999], [1000-5514]. Every piece of information displayed in the Japan map (see Fig. 1) is a tuple (coordinates, density range). This is a particularity of maps: the (x,y) coordinates are part of the primary notation, even if they do not usually appear in the key.

Components are visually depicted by visual variables. There are eight variables that belong to two distinct categories (Fig. 2): two *planar* variables and six *retinal* variables. The planar variables locate any graphic artefact on the 2D plane as a pair of two *coordinates* (x,y). The six retinal variables are: the *shape*, the *size*, the *colour*, the *orientation*, the *value*, and the *texture*. Every variable is characterised by three properties: *steps*, *length*, and *level*. We do not detail them due to space limitation (see [1]).

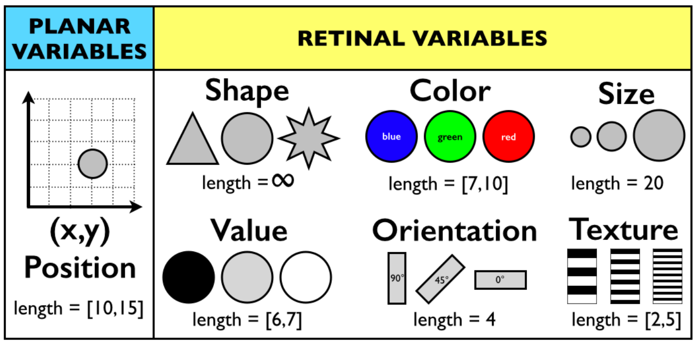

Fig. 2. The 8 visual variables from Bertin [1]

The mapping of a component to a visual variable actually defines the *primary notation* of the modelling language. Every step of the variables from the *primary notation* is semantically meaningful. Hence, it is not allowed to introduce a new step (or change current steps) of those bound variables to carry extra semantics or perceptual attitudes. This is actually the purpose of the *secondary notation.* It is composed of the visual variables that are not bound to any component. These free variables are available to draw attention to certain locations of the diagram, to annotate the diagram, or to add extra (i.e., not part of the modelling language) concerns on the diagram.

3 Method

1. External identification. Wording the diagram (i.e., providing a title).

2. Internal identification. Performed after or in parallel with the external identification. It is composed of three sub-stages:

(a) **category elicitation:** the visual artefacts depicted on the diagram are elicited and those that are visually distinct (i.e., text excluded) are kept.
(b) **component elicitation:** the categories are gathered according to their semantic proximity in order to form components. The components' label, length, and organizational level are also provided (as defined in Section 2),
(c) **pre-mapping of visual variables:** potentially appropriate visual variables are elicited. The final choice is postponed to after secondary notation requirements elicitation.

3. Identification of secondary notation requirements. Visual techniques (e.g., perceptual pop-out effect [16, 13]) to support foreseen uses of the diagram are elicited. Candidate visual variables are chosen according to their length and level.

4. Mapping components to visual variables. Given the list of variables eligible to the primary notation and the candidate secondary notation variables, the designer selects the variables that constitute the primary notation.

5. Selection of the visual variables to be part of the secondary notation. Using the outcome of Stages 3 and 4, the choice of variables for the secondary notation is ratified.

6. Writing down the key onto the diagram. The component to visual variable mapping is depicted on the diagram itself, because it is required to correctly interpret the meaning of the conveyed information. It is visually structured in this way: *(i)* each component is labelled; *(ii)* the categories and the steps of the visual variable(s) to which each category is mapped are placed below or beside the corresponding component (only those appearing on the diagram); *(iii)* the secondary notation is added (i.e, the addressed concern(s), and below or beside, its/their categories and steps).

4 Running Example: SuperElectronicMarket

In this section, we give an overview of the result of applying our method. We consider a UML class diagram representing an excerpt of the static domain of the SuperElectronicMarket information system. There are four classes, `Product`, `Producer`, `Customer`, `Order`, one association class, `OrderDetail`, two binary associations and one composition. A `Customer` orders `Products` that are built by a `Producer`. For each `Order`, there is a series of `OrderDetails`, each one corresponding to the purchase of a certain quantity of a given `Product`. Every `Order` is at least composed of one `OrderDetail`. This example does not strive to be realistic w.r.t. to an actual information system: we focus on visual representation of the information, not its relevance.

Let's now assume that we are free to change the UML class diagram concrete syntax (but neither its semantics nor its abstract syntax). We detail each step of our method:

Stage 1. The diagram is given a title: the invariant relate to the (static) concepts of the SuperElectronicMarket domain.

Stage 2. As we do not afford to change the language semantics, this component elicitation may be puzzling and lead to ill-formed components (e.g., a component with a length of one, or a component whose categories are not mapped to the same visual variable(s)). There are four categories and their associated steps:

– classes that are depicted with rectangles (visual variable: shape); Enumerations are distinguished from classes by a textual annotation (i.e.., ≪ enumeration ≫), which is not visually discriminant according to the SoG;
– associations that are depicted with plain lines (visual variable: shape);
– association classes that are depicted with a rectangle and a dashed line (visual variable: shape);
– compositions that are depicted with plain lines with a diamond head at the source of the relationship (visual variable: shape).

Grouping these four categories together is not allowed because *(i)* classes and enumerations are implanted as a point on the diagram, while the relationship types adopt a linear implantation (for details about implantation types, see [1]); *(ii)* class diagrams are SoG networks, which implies that the existence of relationships prevails over nodes. Hence, we define two components: *concept types*, which comprises the *class* and *enumeration*, and *relationship types* that gathers the association, composition and association class relationships[3].

Stage 3 (and 5). We choose to not use any secondary notation.

Stage 4. The *concept types* has a length of 2 and is qualitative. The *shape* variable is suitable, but *class* and *enumeration* symbols are visually speaking identical.

[3] According to the UML standard [11], association classes are associations with specific properties.

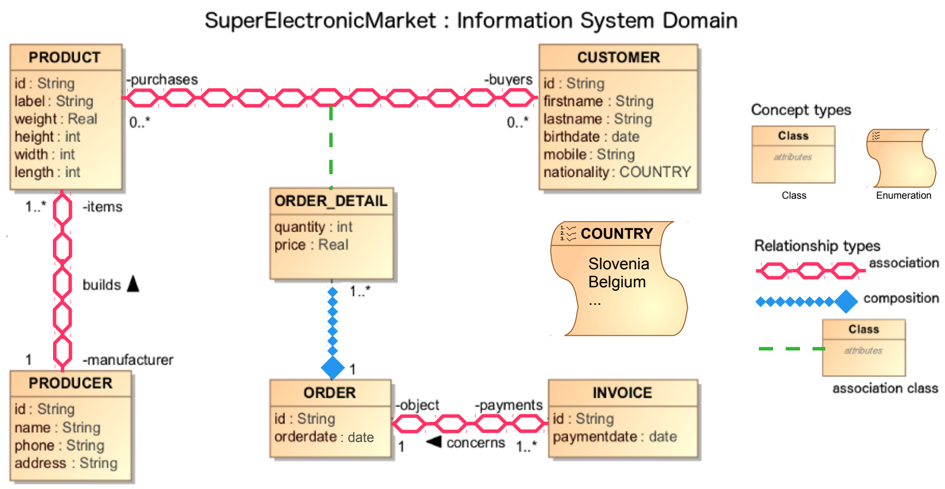

Fig. 3. SuperElectronicMarket UML class diagram with a revised concrete syntax

Hence, we provide a new shape to the *enumeration* that is a rectangle whose vertical sides are curved. We also add a special mark at the top left corner. The rationale sustaining our choice is the following: rectangle with curved sides suggests the meaning of a sheet of paper and the special mark denotes a list of items. The *relationship types* is qualitative, and it counts three categories. Again, any retinal variable could be appropriate. One of the major issues when reading a class diagram is to distinguish between association types. Associations do not carry any head, while compositions and aggregations have a black (respectively, white) diamond. We choose to increase the discrimination of those association types by using colour and by changing the step of the shape variable. Associations are now red, compositions are blue and association classes are green. The shape of the lines is changed this way: chain link shape for associations, diamond link with a large diamond head at the source side for compositions, and a dashed line for the association class. We have chosen to use two variables for the *association types* because it allows to perform redundant coding.

Stage 6. We design the component to visual variable mapping as described in Section 3. The resulting diagram is depicted in Fig. 3.

5 Discussion

In this paper, we presented a method to systematically design diagramming keys, which gives concrete validation rules to ensure the diagram is SoG-compliant. SoG is a theory to design visually effective representations in which the information is understood accurately and (almost) immediately. Inspired by cartography, we argue that it is useful for SE diagrams in which neophytes get accustomed with the language concepts progressively and experts are given dedicated reminders on complex notations (such as UML). We improved the concrete syntax of UML class diagrams to make information quickly and accurately perceptible (other examples: `https://staff.info.unamur.be/nge/effective-keys`). Full power of the SoG can be unleashed when designing a new language: in this context, even abstract syntax (metamodel) will be impacted when performing the internal identification. To empirically validate our method, we plan to set up an experiment where participants would have to answer a series of questions or perform some tasks by relying on a set of diagrams with their keys. A control group would be asked the same actions but they would work with regular diagrams (i.e., without keys). Evaluation and empirical validation of visual concerns have been shown feasible in the context of [2, 4]. Additionally, CASE tools could ensure the automatic validation of SoG's constraints. Finally, diagram design could be assisted by the use of interactive keys.

References

1. Bertin, J.: Sémiologie graphique: Les diagrammes - Les réseaux - Les cartes. Gauthier-VillarsMouton & Cie (1973)
2. Caire, P., Genon, N., Heymans, P., Moody, D.: Visual Notation Design 2.0: Towards User Comprehensible Requirements Engineering Notations. In: RE'13
3. Dykes, J., Wood, J., Slingsby, A.: Re-thinking Map Legends with Visualization. IEEE Transactions on Visualization and Computer Graphics 16(6), 890–899 (2010)
4. Genon, N., Heymans, P., Amyot, D.: Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation. In: Proc. of SLE'10. vol. 6563, pp. 377–396 (2010)
5. Green, T., Blandford, A., Church, L., Roast, C., Clarke, S.: Cognitive Dimensions: Achievements, New Directions, and Open Questions. VLC 17, 328–365 (2006)
6. Henry Riche, N., Lee, B., Plaisant, C.: Understanding Interactive Legends: a Comparative Evaluation with Standard Widgets. CGF 29(3), 1193–1202 (2010)
7. Houy, C., Fettke, P., Loos, P.: Understanding Understandability of Conceptual Models – What Are We Actually Talking about? In: Proc. of ER. pp. 64–77 (2012)
8. ISO/IEC: ISO 9126. Software Engineering – Product Quality. ISO/IEC 9126 (2001)
9. Kraak, M., Edsall, R., MacEachren, A.: Cartographic Animation And Legends For Temporal Maps: Exploration And Or Interaction. In: Proc. of ICC'97. pp. 23–27
10. Moody, D.L., Heymans, P., Matulevičius, R.: Visual syntax does matter: improving the cognitive effectiveness of the $i*$ visual notation. RE 15(2), 141–175 (2010)
11. OMG, Inc.: UML 2.4.1 Superstructure Specification (Aug 2011)
12. Peterson, M.P.: Active Legends for Interactive Cartographic Animation. International Journal of Geographical Information Science 13(4), 375–383 (1999)
13. Quinlan, P.T.: Visual Feature Integration Theory: Past, Present and Future. Psychological Bulletin 129(5), 643–673 (2003)
14. Selic, B.: The Pragmatics of MDD. IEEE Software Journal 20(5), 19–25 (2003)
15. Sieber, R., Schmid, C., Wiesmann, S.: Smart Legends – Smart Atlas. In: Proc. of the 22nd International Cartographic Conference (ICC'05). pp. 11–16 (2005)
16. Treisman, A., Gelade, G.: A Feature Integration Theory of Attention. Cognitive Psychology 12(97–136) (1980)
17. Tudoreanu, M.E., Kraemer, E.: Legends as a Device for Interacting with Visualizations. Tech. Rep. WUCS-01-44 (2001)