Practical Guidelines for the Readability of IT-architecture Diagrams

Henk Koning Claire Dormann Hans van Vliet

Faculty of Science

Vrije Universiteit, Amsterdam

De Boelelaan 1081a,

1081 HV Amsterdam, The Netherlands

henk@cs.vu.nl claire@cs.vu.nl hans@cs.vu.nl

ABSTRACT

This paper presents the work done to establish guidelines for the creation of readable IT-architecture diagrams and gives some examples of guidelines and some examples of improved diagrams. These guidelines are meant to assist practicing IT-architects in preparing the diagrams to communicate their architectures to the various stakeholders. Diagramming has always been important in information technology (IT), but the recent interest in IT-architecture, the widespread use of software and developments in electronic communication, make it necessary to again look at the 'art of making diagrams' for this particular class and its users. The guidelines indicate how various visual attributes, like hierarchy, layout, color, form, graphics, etc. can contribute to the readability of IT-architecture diagrams. The emphasis is on the outward appearance of diagrams. Some additional support is given for the thinking/reasoning processes while designing or using a set of diagrams and an attempt is made to arrive at a rationale of these guidelines. An evaluation process has been performed with three groups of practicing IT-architects. The outcome of this evaluation is presented. This work is part of a more comprehensive research project on "Visualisation of IT-architecture".

GENERAL TERMS

Documentation, Design, Human Factors.

KEYWORDS

Visualization, Architecture, Diagrams, Guidelines, Readability, Hierarchy, Layout, Color, Text, Form, Size, Width, Graphics.

1. INTRODUCTION

We present work done to establish guidelines for the creation of readable IT-architecture diagrams and give some example guidelines and some examples of improved diagrams. We do not come up with new insights into the use of diagramming and diagrammatic reasoning in general. Rather, we try to bring general research findings to the domain of information technology (IT).

This work is part of a more comprehensive study about "Visualisation of IT-architecture".

In this introduction we say a few words about IT-architecture in relation to diagramming, we mention some current developments therein, and indicate the current status of our work in establishing guidelines. In section 2 we give a short description of the approach we took and we discuss the evaluation process. In section 3 we dig into the rationale of the proposed guidelines. In section 4 a condensed subset of the guidelines is presented and illustrated. In section 5 we give some support for the design and use of diagrams. We end in section 6 with conclusions and future work.

To view or print in color this document can be downloaded from the website of this research[1].

1.1 IT-architecture

Computer science has a long history of creating and using diagrams. Flowcharts, functional decomposition diagrams, input/output schemas are examples of such diagrams. See Martin [19] for the state of the art in software diagramming before the general introduction of the personal computer. Methods for analysis and design come with a drawing standard. A recent upsurge is UML for OO design, see Fowler [8]. A well-trained IT-specialist naturally draws good diagrams, doesn't he?

Software architecture is a relatively new branch within software engineering. The recent IEEE Standard 1471 [13] defines it as "Architecture is the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution". Van Vliet [26] places the architecture phase in the software life cycle between the requirements engineering and design phases. During the architecture phase, the interests and concerns of all stakeholders are taken into account to come to a well-balanced solution.

The result of the architecture phase is a series of major design decisions that put constraints on the building process as well as the product delivered. These design decisions are represented in a

[1] http://www.cs.vu.nl/~henk

description, which is often in the form of a series of diagrams. These diagrams give different views of the system, such as the decomposition of a system into major logical building blocks, or the mapping of software elements onto hardware elements. Well-known diagram-based models for describing software architectures were proposed by Kruchten [16] and Soni et. all. [17]. Boar [2] describes in detail a set of IT-architecture diagrams and gives advice on managing architecture on a companywide scale. See Koning [15] for an annotated set of IT-architecture diagrams found on the Internet.

Although the focus is on IT-architecture many guidelines in this research apply to diagrams in related areas as well. As an example of that kind of diagramming can serve the task models in user interface design in Welie [28].

1.2 Current Developments

So why look again at diagramming for IT? Several factors make it interesting to look at diagram representations of IT-architecture. New, less-technical roles emerge in the field of information technology, which are filled by people with no technical background. These non-technical stakeholders play decisive roles in the development, assessment and use of IT-architectures. Another development is that society is becoming more geared to visual communication and there is a demand for more appealing and better visualisations. Much research is done in developing new graphical metaphors. Spence [25] gives a state of the art overview. Next is the widespread availability of drawing software and graphic libraries. These makes 'old style' pencil and template drawings look outdated. A last reason to be mentioned is, that IT-architecture diagrams seem to be less formal than design-diagrams and new (fuzzy) rules must be developed for dealing with them.

Hofmeister [11] and Brown [5] have contributed to a debate about the use of UML for architecture diagrams. The adoption of the IEEE 1471 standard has spurred interest in creating specific viewpoints, based on (visual) models. The question 'how to represent software architecture?' is inspiring SEI [22] and SIGDOC [23] workshops. The outcome of our research can be of value for these developments.

1.3 Communicating architecture

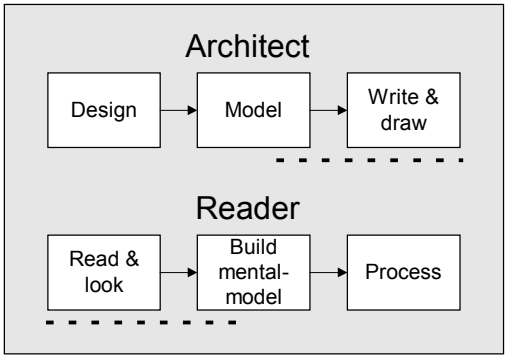

Figure 1 Steps in communicating architecture

For the sake of positioning the developed guidelines, the communication process can be divided in these broad general steps on the part of the author/architect:

1. Designing the architecture
2. Modeling the specification of the architecture for the communication, and
3. Preparing the text and diagrams.

And on the part of the reader:

4. Reading the text and diagrams
5. Making a mental model of the architecture
6. Processing the received information to validate, to accept or reject, to project consequences, etc.

The design step consists of understanding and structuring the needed features, understanding new technology, outlining various solutions, taking major design decisions, etc. It is a goal oriented, free format, creative thinking process. To get abreast of new concepts and technologies books, journals, conferences and seminars support the architect.

In the modeling step parts of the abstract architecture are made concrete. Patterns can be used. Formal semantic models like UML can be used. Company standards and culture can influence the partitioning.

The final step for the architect is preparing the actual deliverables.

In a design project these steps are not clearly divided. Actually there is a constant switching between free format thinking, modeling and drafting. The accent gradually shifts from the first step to the third.

Readers can be end-users, managers, other architects, developers, evaluators, etc. Comprehending the architecture starts with reading the documents. While reading, a mental model is created of the architecture, and the information is rationally and emotionally processed. Here also there is no sharp distinction between the steps, but the focus gradually shifts from mere reading/looking to realizing the consequences. When readers start to discuss with designers both ends meet.

The aim of the guidelines is to strengthen the transition from designer to reader. The scope of the readability guidelines is indicated in Figure 1 by the dotted lines: what can the architect do at the level of preparing the diagrams, to support the reader in quickly and correctly extracting the information? The focus is on the outward appearance of diagrams.

2. ACTIVITIES AND EVALUATION PROCESS

2.1 Activities

We started this research phase by establishing whether comparable work already had been done in this field. As far as we know this is not the case. So we decided to take a intuitive, broad-brush approach and establish a (rather informal) starting point.

Then we identified possible knowledge domains from which guidelines could be extracted. We used Ware [27] as a source on human perception, Horton [12] on technical documentation, Harst [10] on user interface design, Borchers [3] on website design. Other areas to be mentioned are: psychology (gestalt), diagrammic reasoning, information visualization, IT-modeling standards (UML) and icon libraries. Especially Horton [12] proved very useful.

In discussions with IT-architects of the participating companies heuristic rules (personal favorites) were captured. The guidelines were extracted based on our intuitive estimate of their possible value, based on our own experience as software developer and IT-architect.

In case of doubt about the usefulness of a guideline it was incorporated, on the premise that architects can for themselves decide what to use or not.

This resulted in a document with about 200 do's and don'ts, see Koning [14]. About this time we started rereading the same things again in other sources, which signaled to us that we were reaching a saturation point.

The result was presented to three groups of five to seven IT-architects of the participating companies (an international bank, a software house and a government institution). In the workshops that were held, the guidelines were applied as an exercise on existing diagrams. In the discussion of the guidelines in section 4, we show some real-life examples of IT-architecture diagrams that were used and improved in these workshops. The architects were asked to keep the guidelines at hand and apply them in their work as it suited them. They would be asked to give their personal estimate of the value after several months in the form of a questionnaire.

To strengthen the understanding of the guidelines a document was prepared in this period with examples of architecture diagrams that can be found on the Internet. Each diagram was commented on with respect to the various visual attributes. These example diagrams were distributed to the architects and can be found on the website of this research.

We concluded with a questionnaire.

2.2 Evaluation process

To keep it simple, in view of the large number of guidelines and the broad brush approach, the questionnaire consisted of one simple question for each guideline: keep or discard? It was possible to add a comment to this choice.

IT-architectects are an independent minded and busy bunch of people and they needed some urging to go this last step with us. Twelve architects responded to the questionnaire. Their responses varied from 'discard these 85 guidelines' to 'just keep them all and I'll gladly use them as a checklist'. In the end only five guidelines didn't receive 50% support to keep (they had a majority vote for 'discard'). We feel in this case it is appropriate to give more weight to the negative responses. If we draw the line on '70% support needed to stay in' another 27 guidelines fall from the table. In Table 1 this is summarized. The overall support column is calculated as % 'keep' of total votes for all guidelines for this visual attribute.

Table 1 Evaluation summary

Visual attribute	Number of guide-lines	Drop	Doubt-full	% Support
Hierarchy / focus	14	-	3	0,86
Form / size / width	8	1	1	0,86
Layout	29	-	4	0,86
Color	42	3	6	0,83
Connectors	16	-	3	0,86
Use of text	11	1	-	0,89
Graphics & icons	15	-	3	0,82
Context & design	45	-	5	0,88
Use in Report	10	-	2	0,88
Totals	*190*	*5*	*27*	

Some of the comments made by the respondents were: not a guideline but an explanation; not generally applicable (too specific); redundant (already in another guideline); too theoretical (mostly on perception). In sections 4 and 5 we give some examples of rejected or considered doubtful guidelines.

Our original document also contained a summary of 'Gestalt' rules, which were given as background in perception theory. Based on the voting these must also be considered doubtful. We expected (hoped to unleash) discussion from the participating architects and a rejection of a substantial number of guidelines. The discussion did not take place much, however, that was 'too far away' from their practical situation. The amount of rejected or doubted on guidelines supports the credibility of the (remaining) set of guidelines. Overall the architects appreciate the effort we have done to provide them with practical guidelines.

So the credibility of these guidelines is based on the quality of the sources, on our personal judgments and on taking votes from a small group of practicing architects. Our claim is not that these guidelines are proven correct. Our claim is that these guidelines are considered valuable by practicing architects.

3. RATIONALE OF THE READABILITY GUIDELINES

In this section we want to give some pointers to 'explain' the guidelines to IT-architects and to further motivate their use. In this we go only one step into the unknown. Further explanation for the interested reader can be found in the reference material.

The rationale behind the readability guidelines as a whole centers around four points: Human Perception, Appeal, Building the Mental Model and Support while Processing which are outlined in the next paragraphs.

3.1 Human Perception

First and foremost reading and looking are functions of human perception, which is in itself an amazing and very complex process. As Bertin [1] pointed out all the visual attributes play their role in helping the human brain in recognizing and grouping objects in the diagrams.

The guidelines also point to limits in human perception: the maximum numbers of objects, colors, forms, sizes, etc, a user can comfortably handle. The viewer needs a certain minimal differentiation between types of objects, a certain minimum space between objects, etc.

3.2 Appeal

Another major concern around the use of diagrams in IT-architecting is that diagrams must be appealing. Is the diagram attractive? Does the first impression stimulate the reader to dive into it and take in the information contained in the diagram? In modern day world most people are overloaded with information. If it doesn't look nice, it is easily put aside. The competition is on.

3.3 Building the Mental Model

When someone is reading the text and viewing the diagrams of an architecture description, he is building a mental model of the architecture. These elements of the mental model may be explicitly defined or tacitly implied. While viewing a diagram, a constant process of constructing possible objects, relations and attributes, evaluating the plausibility of the constructions, and affirming or rejecting them, is going on. This process is based on visual attributes present in the diagram. Information from already seen diagrams is taken into account, as is information from previous experiences (prior knowledge).

Our guidelines try to prevent as much as possible the construction of incorrect elements in the mental model and to promote as much as possible the construction of correct elements. Visual attributes of diagrams are what Norman [21] calls 'affordances' that provide strong clues as to their meaning. Creating diagrams without taking into account all the visual attributes that lead the viewer, results in meaningless variation and confusion. The value of diagrams in communication very often expressed by the saying 'A Diagram is Worth Ten Thousand Words'. Larkin and Simon [17], in their classical paper, have argued that the value of diagrams lies in the ease of recognizing complex relationships between many elements.

3.4 Support while Processing

Diagrams not only transfer information to build up a mental model, according to Narayanan [20], they also assist in processing the information. Viewing a diagram of an architecture helps keeping part of the model conscious in mind and it inspires and corrects thinking. It inspires because you can combine objects and/or attributes that you see in the diagram and construct alternative diagrams. It corrects because you 'see' more easily what is possible and what isn't.

For reasoning you need a starting point. In natural language, a starting point is created by expressions like "lets presume …, what consequences might that have?". The closer a diagram comes to your ideal starting point for a line of reasoning, the better you can think. If you have to impose additional mental constructs on a diagram to create your starting point, you have less brain resources for thinking and the starting point is weaker and less inspiring.

4. GUIDELINES CONCERNING VISUAL ATTRIBUTES

Our list of guidelines concerning visual attributes has the following sections: hierarchy/layers, forms/size/width, layout, color, connectors, text, and graphics & icons. For each section we give some examples of the guidelines and some of the rationale.

4.1 Hierarchy

A diagram may contain more elements than can be consciously viewed at in one glance. That need not be a problem, provided the viewer is given enough clues to easily perceive the main message, and separate this from any additional content. See Table 2 for examples of guidelines in this respect.

Table 2 Hierarchy Guidelines

Creating a clear hierarchy in complex diagrams
1. Design a complex diagram so it can be read in 30 seconds, in 3 minutes, and in 30 minutes. A diagram (graphics) must immediately and automatically make the most important point, then present secondary points, and with study reveal details. A diagram must organize information into a clear visual hierarchy.
2. Different visual attributes can play a role in indicating the hierarchy: size (bigger), colors (brighter), lines (thicker), pattern (emphasize the main path, shorter lines), etc.
3. Avoid more than three distinct visual levels.
4. Use primary colors only for to objects that need immediate action.

If you don't add hierarchy to a complex diagram, you require the viewer to derive the hierarchy (levels of importance) while viewing and to keep that information in his head, while processing the diagram further. This incurs extra work for the viewer, possibly irritation (up to quitting altogether) and possibly wrong conclusions. So a complex diagram without hierarchy is not appealing and has a high chance of not being properly understood. See Figure 2 for an example of a diagram with two visual hierarchical levels.

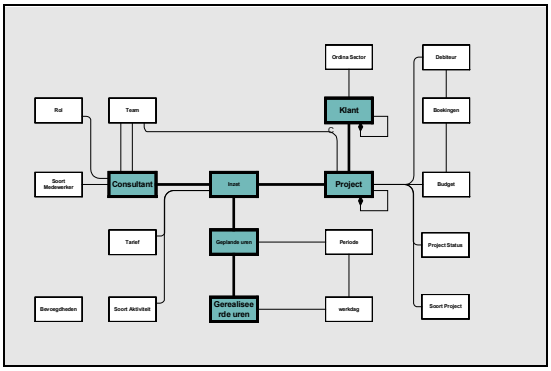

Figure 2 A diagram with two visual hierarchical levels

An example of a considered doubtful guideline in this category is "to create a sense of urgency in a diagram use simple graphics or 'unstable' graphics (like something almost falling to the ground)".

4.2 Forms of Objects, Size, Width

Forms used in IT-architecture diagrams generally fall into two categories: first there are the formal shapes like boxes, diamonds,

data stores. Second there are the freestyle small graphical images like icons and small clipart. The use of graphics and icons is rather new and is treated in more detail in section 4.7. See Table 3 for examples of guidelines in this respect.

Table 3 Forms of objects Guidelines

Forms of objects
1. Be clear about what your forms mean. Are all boxes equal? And do they then mean the same too? If not, provide annotation. Be consistent in the use of forms in a set of diagrams.
2. Try to match the outward forms of objects to the intrinsic properties of the objects.
3. Use more detailed, realistic, three-dimensional symbols for concrete and tangible objects and use simple, geometric shapes for abstract concepts.
4. Don't use more than 6 different forms in one diagram.

An example of a rejected guideline in this category is "high level -> low level: more shading in the shapes".

Size and width are important visual attributes. Many people underestimate how strong the size-signal is in real life. A taller person is easily seen as more important. A bigger car, house, etc. induce similar connotations. Different sizes in one diagram would 'normally' mean differences in importance. See Table 4 for examples of guidelines in this respect.

Table 4 Size and Width Guidelines

Size and Width
1. See to it that similar objects have equal sizes/widths. Only deviate in size and width if you want to signal something.
2. Avoid resizing objects because of short or long texts. For long texts the alternative is to put a short label in the object and the text in an annotation (insert in the diagram) or in a textbox in the margin of the diagram.
3. Avoid making objects smaller to have more information in one diagram, or making objects bigger to fill up a diagram that otherwise looks too empty.

Different sizes of the same object in different diagrams normally means the object has grown or shrunk. See Figure 3 and Figure 4 for an example.

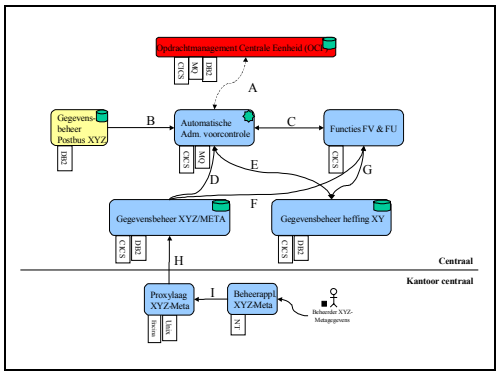

Figure 3 A diagram from a workshop participant

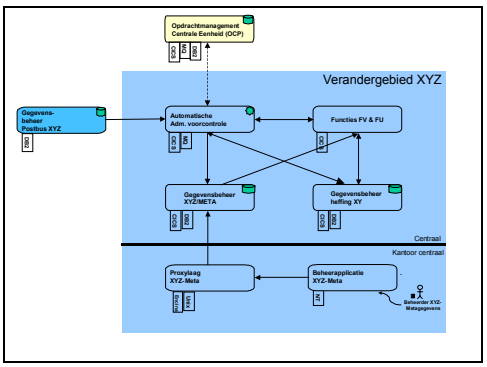

Figure 4 The same diagram as Figure 3 after applying guidelines about Size and Layout

In Figure 3 the objects have different sizes, but this has no meaning, so the 'size message' has to be suppressed in interpreting the diagram. The vertical position of the objects is meaningful, but the horizontal position is vague. This means the horizontal positions all have to be checked and interrelated to see whether there is some significance. In Figure 4 the sizes of the objects have been made equal and the objects have been positioned on a grid. This means much less parsing of the diagram has to be performed and one can start reasoning about the meaning immediately.

4.3 Layout

Putting the objects in a diagram in a pattern that is easily recognizable and fitting to the underlying message, is a great aid to the viewer of the diagram. It very much helps in discerning and remembering which objects there are and which relationships are relevant to consider. A clear pattern makes it easy for the eye to come back to objects that were already perceived, and thus supports processing.

Layout aspects of a diagram include: basic pattern, horizontal and vertical alignment, above/before positioning, symmetry, distance of objects from the center and from other objects, distribution of white space. A basic pattern makes clear to the viewer what strategy is being followed in positioning objects and what meaning can be derived from the position of an object. For instance: in a workflow diagram the activities might be positioned from left to right in the order of execution and having the same vertical position can mean being executed in the same stage of process.

Clearing up messy diagrams often starts with improving the layout, so it is one of the most important visual attributes. A good layout is perceived instantly and almost unconsciously. An unclear layout keeps nagging and hinders perceiving the more detailed information.

Providing enough, but not too much, white space makes diagrams elegant. White space gives room to envision alterations or additions, and in that way (again) supports reasoning about the diagram.

Figure 4 is an example of a diagram in which attention was paid to choosing a basic pattern and proper horizontal and vertical outlining of objects.

Table 5 Layout Guidelines

Layout
1. Choose a clear, recognizable positioning pattern for the objects in a diagram. Familiar layout patterns are: chain, grid, tree, web.
2. In positioning objects 'natural positions' should be preferred, e.g. central horizontal and vertical axis, secondary axis on ¼ and ¾ or 1/3 and 2/3, etc.
3. Objects should be positioned on horizontal and vertical lines, e.g. not positioning them so should be meaningful.
4. Take into account the 'natural flow' from left to right, and from top to bottom. Make other flow directions clearly recognizable.
5. Provide enough white space. A crowded diagram looks obtrusive.
6. Use white space to distinguish objects, borders, groups, etc.
7. In a set of diagrams similar objects should have a similar position.

Figure 5 is another example of a diagram with a clear layout pattern. The main page is divided in areas that convey the main structure of the architecture. In this case it shows how an application on a client workstation in the center of the left main area is serviced by applications on several local LAN servers and some remote midrange and mainframe machines. It is realistic, in that it deals with real machines and locations. It is conceptual in that some of the represented machines are actually groups of equivalent machines, and in that the whole left main area is in reality occurring three times at different locations. The operational division between LAN-hardware and midrange/mainframe is made clear by the extra white space area.

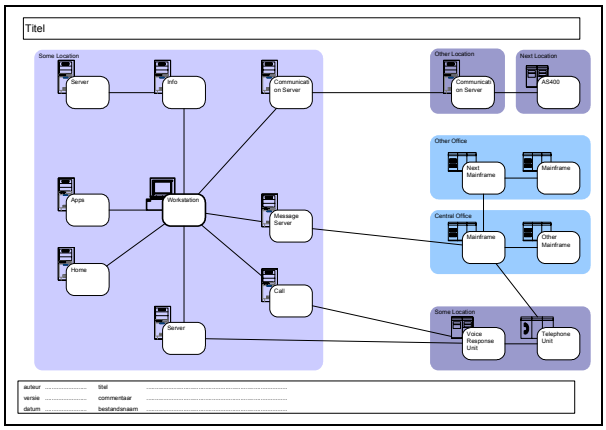

Figure 5 An example of a clear layout pattern

4.4 Color

Color is a very strong visual signal, so it is worthwhile paying attention to it. It is a visual attribute that is strongly influenced by 'prior knowledge', like cultural values, fashion colors in clothing or magazines, or company colors. Color is also rather new in computer documentation. Even nowadays not all practicing IT-architects have a color printer at hand. Due to the newness of the subject, the guidelines contain an element of 'color education', which may over time become less relevant.

Additional meanings can be easily (temporarily) attached to a certain color. Using a distinct color in a diagram for an object with a particular attribute, can program the meaning of that color for the rest of the documentation. Color can enlarge the appeal of the diagram.

Table 6 Color Guidelines

Color
1. Use color. The competition for the attention of the viewer is on!
2. Use color with restraint. Problems with (too much) color: wrong prior associations, distraction, tiresome, less legible, fuzzy, unreliable.
3. Don't use more than six colors in one diagram.
4. Color is especially useful to categorize objects, i.e. to group objects where other means (alignment, positioning) are not possible.
5. Use vivid colors only for strong signaling.
6. A safe rule is: choose light, non primary colors with hues from over the whole color wheel.
7. If possible, follow the company colors.
8. Western viewers tend to prefer colors in the following order: blue, red, green, purple, orange and yellow.
9. To be recognized on screen colors need to be further apart than to be recognized on paper.
10. To avoid problems with colorblindness or with printing in black and white: use colors with different levels of brightness/lightness.

11. Let proximity in color parallel proximity in meaning (this extends over diagram borders).

The 'fuzziness' in the meaning of colors is reflected in some of the guidelines, which state 'do something like …'. The combination of these guidelines can be used to setup a color scheme for a set of diagrams. Chijiiwa [6] gives many examples of color combinations and atmosphere.

In Figure 3 the color 'vivid red' was meant to indicate that the upper object is outside the scope of the system. The designer did not realize that the color red drew all attention to the object.

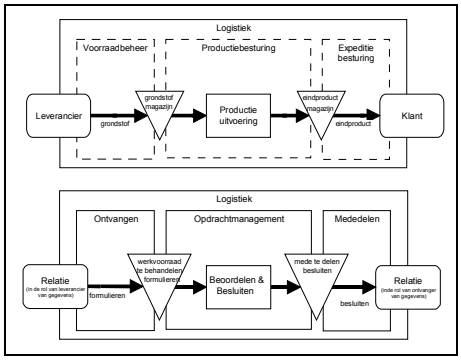

Figure 6 Another diagram from a workshop participant

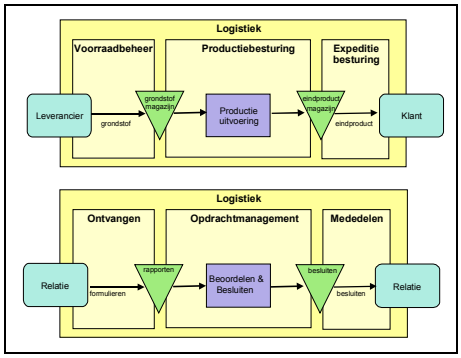

Figure 7 The same diagram as Figure 6 with colors

In comparing Figure 6 and Figure 7 you can see what a big difference adding colors can make. Figure 7 is much more appealing, the objects are more easily perceived against their background, and the semantic distance between the objects and the comparted areas is immediately clear.

An example of a rejected guideline in this category is "There are no ugly colors, or ugly color combinations. There are (combinations of) colors we are not used to … "

4.5 Connectors

Connectors are quite specific to IT architecture diagrams, which are all about objects and relations between objects. Not all architecture diagrams contain many connectors. A diagram showing how functionality is divided in three main areas may even do without any connectors. Process diagrams always contain connectors. It seems that the more precise the diagram is, the more design oriented, the more connectors come into play.

When connectors come into play, in number equaling or surpassing the number of other objects, they soon become problematic. Common problems are that connectors make the diagram look messy or overcrowded, and that connectors are not easy to follow. The usual solution is to try to find a better positioning of objects to make the connectors look better. This is not easy. Some designers give up and simply accept the mess, but this makes diagrams not appealing and more difficult to comprehend.

Some software tools optimize the flow of connectors. The problem with these tools is that the resulting layout totally ignores the already established, meaningful patterns in the diagram.

Table 7 Connector Guidelines

Connectors
1. Let the lines of connectors overlap, as long as this does not lead to unclearness or ambiguity.
2. Avoid unnecessary bends in connecting lines.
3. Avoid a too great emphasis on line-ends, like arrowheads. If possible, leave the arrowheads out all together.
4. Rounding of bends gives a more natural impression of flow.
5. Many close parallel lines are difficult to follow individually. A possible solution is to maintain different distances between (sets of) lines.
6. If parallel lines bend together, keep equal distances before and after bend.

Connectors that are easy to follow by the human eye give support to reasoning about the relationships between the objects. The guidelines about connectors give some ideas that make life with connectors more bearable. You can look at Figure 8 in section 4.7 and try to recognize some of the guidelines for connectors.

An example of a considered doubtful guideline in this category is "Give connectors a different line-width from other objects on the diagram. Smaller? Bigger?"

4.6 Use of Text

Text and graphics are friends and not enemies. Combine the power of 2. Text can be very strong in suggesting the proper interpretations and associations and in stimulating thinking. The guidelines on the use of text try to stimulate the architect to be diligent in adding proper titles, subscripts and annotations. They do matter. You can't expect people to remember everything you said about this diagram in your (wonderful) real-life presentation. Text is important to speed up the building up of the proper mental model and to create a good starting point for a line of reasoning.

Table 8 Text in diagrams Guidelines

Text in diagrams
1. Write in active voice, use action words, use examples, tell how things look/sound/feel/smell, use concrete words that can be memorized verbally and visually.
2. If space does not permit a complete label, place a

short label in or near the shape and use a footnote to provide more information. 3. Provide clear titles, subtitles and subscripts for a diagram. Possible uses: indicate position in whole set of diagrams, show importance, give reading clues, draw conclusion, and explain who/what/where/how/why/… 4. Annotations give answers to questions, focus the attention, and explain. Design labels and annotations so they stand out from the background but remain subordinate to the subject matter.

4.7 Graphics & Icons

Graphics & icons can be very useful to make diagrams appealing, especially for a non- technical audience. With graphics we mean freestyle artistic representations of objects or actions of modest size, for instance 1 square inch in print. Drawing packages for the PC usually come with libraries of this kind of graphics. With icons we mean the even smaller images of 32 by 32 pixels, which are today mostly known as recognition symbols for graphical user interfaces. Dreyfuss [7] and Modley [18] contain dated but still inspiring collections of icons.

The use of graphics and icons parallels the use of color. It's new. It has a strong visual impact. It is very dependent on cultural or company context. It's fuzzy. A difference is that it is a worthwhile option. You don't have to use it. Another difference is that is it more difficult to come up with a new set of icons. Not many people have the ability to draw new icons or graphics. There are possibilities here for creative contributions in an architecture design project, comparable to creating new icons for a web site.

Table 9 Graphics & Icons Guidelines

Graphics & Icons
1. Use icons and graphics modestly. 2. Use icons: to speed search, for immediate recognition, for better recall, to save space, for graphic or spatial concepts, for visual appeal. 3. Don't try to be funny. 4. Use graphics that are meaningful in the daily life of the viewers. 5. Adding icons to your boxes is useful for categorizing, like indicating all functions related to 'finance'.

Figure 8 is an simplified example of a technical diagram in which icons are used to speed up recognition. The idea is that the same icons are used in the whole set of diagrams, and that an icon represents an abstract concept. Of course this only works if the icons are self-explaining to the viewer, or 'easy' explainable. In Figure 8 there are four different green objects, of which the middle two have something in common, as indicated by the color and the icons. The green objects have a number of unspecified relations to three yellow objects, of equal type, that are fulfilling three roles/functions/…

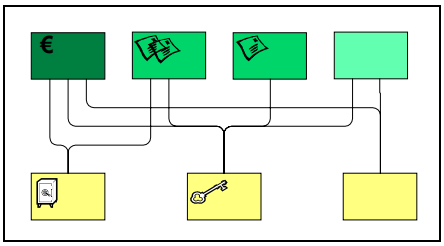

Figure 8 An example of a diagram with icons to speed up recognition.

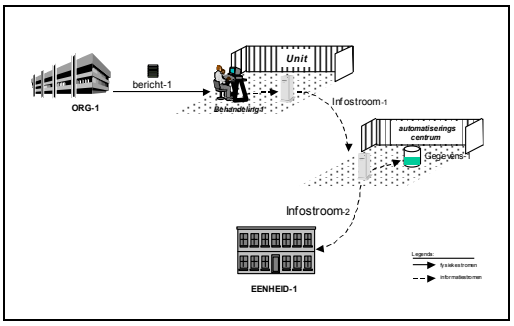

Figure 9 An example of a diagram with some freestyle graphics

Figure 9 is a simplified example of the use of free graphics. This is meant to be more appealing for non-technical viewers, but is less suitable for adding technical information. Of course this only works if the graphics are meaningful to the viewer. The details of the graphics should not distract from the meaning of the graphics.

5. CONTEXT AND DESIGN

Our focus in this paper is on the outward appearance of diagrams, as indicated in the various visual attributes. But in dealing with diagrams one soon comes to circumstantial factors, not specifically tied to one or more visual attributes. The 'Context and design' section of the guidelines tries to capture some of the circumstantial factors we encountered in our search for guidelines concerning visual attributes.

5.1 Design & Evaluation

The aim of this kind of guidelines is to support the architect who is planning or evaluating a set of diagrams from a general diagramming perspective. We do not take into account specific IT-architecture issues (types of views, certain concerns, stakeholders, etc). The essence is "making good diagrams is hard work".

Table 10 Designing a set of diagrams Guidelines

Designing a set of diagrams
1. The use of diagrams should be designed, just as the architecture itself. It should grow during the design project, not added afterwards. 2. From the start of the design project, be on the outlook for good graphics which are meaningful to the stakeholders. 3. Create a clear structure in the set of diagrams.

4. The use of visual attributes should be designed with the whole set of diagrams in mind.
5. Don't only prepare diagrams that show the good news. Create also diagrams which help the viewer to compare old and new, or to find possible blind spots in your report, to visualize for himself all your arguments…
6. Take into account the way of thinking of the organization. Adjust. Reach out.
7. Composing a diagram: start by identifying the most important information in the diagram. Allow no more than three to seven objects at this top level. Ideally, identify a single object to dominate the graphic.
8. Composing a diagram: formulate the specific questions a viewer could answer by looking at the diagram.
9. To evaluate a diagram the following list of general questions people ask while looking at diagrams can be useful: What is it? What is most important? How does it relate to other diagrams. How do I use it? Where am I in this diagram? Where does it start? What is the difference compared to the current situation?
10. A series of consecutive diagrams can be used to: build up a complex diagram, simulate processes, progressively reveal more detailed information.

An example of a considered doubtful guideline in this category is "create graphic organizer as preview for coming structured information. Concept structure rather than report structure ➔ additional insight in / organization of subject matter".

5.2 Diagrams in the Architecture Report

To make the subject complete, here are some guidelines for incorporating diagrams in an architecture design report. In the text of this paper we have given an example of positioning diagrams and referencing meaningful to diagrams.

Table 11 Diagrams in the architecture report Guidelines

Diagrams in the architecture report
1. The distance in the report (number of pages) between a diagram and a reference to it should not be too big. A possibility is to repeat the diagram (this should be signaled in the caption).
2. Idea: repeat a thumbnail in the margin of the text. Highlight on the thumbnail the part of the diagram that is being treated in the text.
3. Always number your diagrams (figures). Text references to diagrams are always by number (not: above/below, at your left/right). References to diagrams are preferably placed at the end of a paragraph. See to it that all diagrams are referred to in the text.
4. Simply and directly inform the reader of diagram content and purpose.
5. Encourage the reader to look at the diagrams. Ask thought-provoking questions about them. Point to peculiarities.

6. CONCLUSION AND FUTURE WORK

Looking at the various visual attributes is an effective means for finding ways to improve the readability of IT-architecture diagrams. Visual attributes we took into consideration are: visual hierarchical levels, layout aspects, coloring, forms and size, use of icons and graphics. Britton et al [4] have also expressed the importance of some of these classes of guidelines, in particular the use of a clear structure (hierarchy and layout) and motivating symbols (forms, icons and graphics).

A lightweight validation of our guidelines has been established in workshops with IT architects from various participating companies. This gave support for the practical usefulness of most of our guidelines.

Directions for further research include: maintaining and extending the guidelines (for instance the use of fonts), getting a better idea about the priorities of the various guidelines and about the way the visual attributes influence each other, differentiating different usages of diagrams, positioning the guidelines in a formal model of communication and perception, link the readability of diagrams to the readability of text. See Hargis [9] for developments concerning the readability of text.

Armed with an understanding of the most relevant visual attributes of IT-architecture diagrams we plan to start research into specific (new) types of diagrams that can be used to visualize specific aspects of an IT-architecture and into the way IT-architecture diagrams can be adjusted to better serve the interests of specific user groups (stakeholders).

ACKNOWLEDGEMENTS

This research is partly funded by grants from the Computer and Software Centre of the Dutch Tax and Customs Administration, ING Barings, Ordina Panfox, and Ordina Institute.

REFERENCES

[1] J. Bertin, *Semiology of graphics*, The University of Wisconsin Press, 1983.

[2] Bernhard. H. Boar, *Constructing Blueprints for Enterprise IT Architectures*, Wiley, 1998

[3] Jan Borchers, Oliver Deussen, Arnold Klingert, Clemens Knörzer, *Layout Rules for Graphical Web Documents*, Computers & Graphics, Vol. 20, No. 3, 1996

[4] Britton et al, *Evaluating the intelligibility of diagrammatic languages used in the specification of software*, First International Conference, Diagrams2000, Edingburgh, 2000

[5] A. Brown et al (Eds), *Describing Software Architecture with UML*, Workshop at the 23rd International Conference on Software Engineering, Toronto, ACM, 2001.

[6] Hideaki Chijiiwa, *Color Harmony*, Rockport Publishers, 1987

[7] Henry Dreyfuss, *Symbol Sourcebook: An Authoritative Guide to International Graphic Symbols*, McGraw-Hill, 1972

[8] M. Fowler, *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Addison-Wesley, Reading, Massachusetts, 1999.

[9] Gretchen Hargis, *Readability and computer documentation*, ACM Journal of Computer Documentation, August 2000, Vol.24, No. 3.

[10] Guido van der Harst, Rob Maijers, *Effectief GUI-ontwerp*, Academic Service, 1999 (in Dutch)

[11] C. Hofmeister, R. Nord and D. Soni, *Applied Software Architecture*, Addison-Wesley, Reading, Massachusetts, 1999.

[12] W. Horton, *Illustrating Computer Documentation*, Wiley, 1991.

[13] IEEE, *IEEE recommended practice for architecture description*, IEEE Standard 1471, 2000.

[14] Henk Koning, *Guidelines concerning readability of IT-architecture diagrams,* to be found on http://www.cs.vu.nl/~henk, 2001

[15] Henk Koning, *Guidelines readability: example diagrams from internet,* to bee found on http://www.cs.vu.nl/~henk, 2001

[16] P.B. Kruchten, *The 4+1 view model of architecture,* IEEE Software **12** (6) (1995) 42-50.

[17] Jill. H. Larkin, Herbert A. Simon, *Why a Diagram is (Sometimes) Worth Ten Thousand Words*, Cognitive Science 11, 65-99 (1987)

[18] Rudolf Modley, William R. Myers, Diana G. Comer, *Handbook of Pictorial Symbols: 3250 Examples from International Sources*, Dover Pubns; 1977

[19] James Martin, Carme McClure, *Diagramming Techniques for Analysts and Programmers,* Prentice-Hall, 1985

[20] Narayanan, *Diagrammatic communication: a taxonomic overview*, Technical Report CSE97-06, December 2, 1997

[21] D. A. Norman, *The Psychology of Every Day Things*, Basic Books, 1988.

[22] Software Engineering Institute, *Workshop on Software Architecture Representation,* 16-17 January 2001, http://www.sei.cmu.edu/publications/documents/01.reports/01sr010.html

[23] Dennis Smith, Bill Thomas, Scott Tilley, *Documentation for software engineers: what is needed to aid system understanding,* workshop at SIGDOC 2001, Annual ACM Conference on Systems Documentation, http://doi.acm.org/10.1145/501516.501570

[24] D. Soni, R.L. Nord and C. Hofmeister, *Software architecture in industrial applications*, Proceedings 17th International Conference on Software Engineering, ACM, 1995, pp 196-207.

[25] Robert Spence, *Information Visualization*, ACM, Addison-Wesley, 2001

[26] H. van Vliet, *Software Engineering: Principles and Practice*, John Wiley & Sons, Chichester, 2000.

[27] Colin Ware, *Information Visualization*, Morgan Kaufmann Publishers, 2000

[28] Martijn van Welie, *Task-based User Interface Design,* Master Thesis, SIKS Dissertation Series No. 2001-6, electronically available at http://www.cs.vu.nl/~martijn/gta/docs/Welie-PhD-thesis.pdf