Software architecture reconstruction
Krikhaar, R.

Citation for published version (APA):
Krikhaar, R. (1999). Software architecture reconstruction

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
Appendix C

Presentation Tools

In this appendix we briefly discuss, in a chronological order, a number of proprietary presentation tools we have developed over the years.

C.1 Teddy-Classic

*Teddy-Classic* [Omm93, Roo94] displays components and relations between components. Components are represented by boxes and relations are represented by lines. *Teddy-Classic* requires as input a relation file and optionally a view file and a component file. The user can layout the components on the screen (using a mouse). The layout can be saved in a so-called view file. Later on, the view file can be used again, even in combination with another relation file (having the same components as carrier). So, in *Teddy-Classic*, the relation and view are separate concepts.

Components are clickable, meaning that the user can ‘click’ on a box, which results in a text viewer with component information. This ‘click’ information is described in a component file that describes the relation between components and information files. *Teddy-Classic* has various types of boxes, which represent different types of components. The lines also have different representations, dictated by the types of components to which they are connected.

Figure C.1 shows an example of the output of *Teddy-Classic*. It shows the same information as presented in the diagram of Figure 4.11 (page 83). A thick line between two components means that there is a relation between the graphically lower component and the higher component. A thin line
Figure C.1: Teddy-Classic

represents a relation between the higher component and the lower component. From these rules we infer that bi-directional relations are always represented as thick lines.

Teddy-Classic was written, in 1992, in the programming language C using X windows. The separation of relations, click information and views is a powerful concept: various relations can be displayed with the same view information. A drawback of Teddy-Classic is its graphical appearance. Bi-directional relations are not explicitly handled and all the boxes are of the same size; lines start (end) in the middle of a box, which reduces the possibilities of creating an appealing layout. There are now more tools available that provide similar functionalities, e.g. Rigi [SWM97], developed at the University of Victoria, Canada.
In 1996, the functionality of Teddy-Classic was also implemented in Visual Basic in combination with Visio\textsuperscript{TM} [Vis]. The Visio tool displays 1-D and 2-D objects, say for clarity arrows and boxes. Each box has various connection points to which an arrow can be connected. The connected arrows automatically re-size when a box is moved (using a mouse). Visio calculates the best connection points for arrows (they call it ‘dynamic glueing’) and it automatically layouts boxes. Furthermore, arrows may be straight, but the tool can also bend lines to beautify the layout. An example of the output of Teddy-Visio is given in Figure C.2.
C.3 Teddy-PS

The aim of Teddy-PS was to present architectural information in forms resembling as closely as possible the diagrams already used in the architecture documentation concerned. So, in 1996, Teddy-PS was developed to resolve drawbacks of Teddy-Classic: boxes of the same sizes and arrows that start (end) at predefined positions at the border of these boxes.

Teddy-PS requires as input a view file and one or more relation files. The view file is a prepared postscript file that contains the layout of all the components (boxes) and possible relations between components (arrows). The view file is manually created (by adapting a copy of a template view file). Per relation, Teddy-PS filters, from the view file, the corresponding arrows and gives them a colour. Teddy-PS also calculates the sizes of arrows in the case of multi-relations. An example of the output of Teddy-PS is given in Figure C.3.

C.4 Teddy-ArchView

All of the Teddy tools discussed above use two dimensions to present information. Teddy-ArchView, developed in 1997, presents architectural information in a three-dimensional picture [FJ98]. The tool’s input consists of various relations and part-of relations. From this information, Teddy-ArchView generates a VRML [VRM] description. The result is presented in a standard Web browser (using a plug-in, a VRML viewer) or any other VRML viewer. The viewer gives the user the opportunity to walk through the information, in a virtual-reality world. An example of such a world is given in Figure C.4.

C.5 TabView

TabView (developed in 1998) presents relations and multi-relations in a tabular (or matrix) form in a Web browser. The input for this tool is a use relation, e.g. imports Files, Files, and a number of partof relations. For the sake of discussion, we will call the use relation $U_{1,1}$ and the chain of partof relations $P_{1,2}, P_{2,3}, \ldots$. For example, decomposition level 1 refers to Files, level 2 to Comps, level 3 to Subs and level 4 to Systems, so partof $P_{1,2}$ describes which Files belong to which Comps.
Figure C.3: Teddy-PS
Figure C.4: Teddy-ArchView
TabView shows, in a Web browser, a matrix that belongs to a tuple of focus points: a focus point in the domain $x$ at level $d$ plus a focus point in the range $y$ at decomposition level $r$. In the first column the domain's focus point $x$ is presented, in the second column the constituents of $x$ are listed. In Figure C.5 the domain's focus point is $Comm$. The constituents of $x$ can be calculated with RPA: $partof_{d-1,d,x}$. In the given example the constituents of $x$ are, amongst others, $CIL$, $CUL$, and $Std$. Analogously, in the first row the range's focus point $y$ is presented, and in the second row its constituents ($partof_{r-1,r,y}$) are given. The cells in the matrix show whether a tuple exists in the relation $U_{d-1,r-1}$ or, in the case of a multi-relation$^1$, they also show the corresponding weight in RPA:

$$U_{d-1,r-1} = P_{r-2,r-1} \circ \ldots \circ P_{1,2} \circ U_{1,1} \circ P_{d-2,d-1}^{-1} \circ \ldots \circ P_{1,2}^{-1}$$

The user can navigate through the information by clicking on hyperlinks. Zooming-in can be achieved by clicking on elements in the second column or row. A new matrix is then calculated and presented with the clicked element as a new focus point (preserving the other focus point). Zooming-out can be performed by clicking on the element in the first column (row). The parent of this element becomes the new focus point (again preserving the other focus point). When the user clicks on the cells, the corresponding tuples (lowered to decomposition level 1) are presented in a table. All the calculations are performed on request (i.e. after a user's click); a Perl script accessed via cgi [SQ96] calculates a new matrix or table.

Since the development of this prototype, a more elaborate version of TabView has been implemented by the Switch development team [Gla98, BGKW99]. It contains a more dedicated user interface containing, amongst other things, more zooming and hiding actions.

$^1$In case of multi-relations, the part-of relation $P$ of the formula should be interpreted as a multi-relation.
Figure C.5: TabView