Electronic Communications of the EASST
Volume 13 (2008)

Proceedings of the
Second International Workshop on
Layout of (Software) Engineering Diagrams
(LED 2008)

# 3D UML Heuristic Challenge

Paul McIntosh and Jens von Pilgrim

12 pages

Guest Editors: Andrew Fish, Harald Störrle
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

# 3D UML Heuristic Challenge

**Paul McIntosh**[1] **and Jens von Pilgrim**[2]

[1] School of Computer Science and Information Technology , RMIT University, Australia,
paul.mcintosh@internetscooter.com,
[2] Lehrgebiet Software Engineering, FernUniversitt in Hagen, Germany,
Jens.vonPilgrim@FernUni-Hagen.de

**Abstract:** Quality of diagram layout is not restricted to just the elements of one diagram and it is also not restricted to just two dimensions. Three dimensional spaces can be used to layout multiple related diagrams to give the user of the diagram a much more comprehensive view of a software system. The 3D UML Heuristic Challenge presents a workshop session that challenges the traditional methods for layout of UML diagrams and tests 10 information visualization heuristics as a means of evaluating the quality of 3D UML diagrams.

**Keywords:** Heuristic Evaluation, 3D Software Visualization, UML, Usability, 3D UML

## 1 Introduction

The goal of the 3D UML Heuristic Challenge is to workshop 10 well known heuristics using UML diagrams and 3D layout techniques. The challenge is to find the best performing heuristics and the best performing diagram layouts. This section presents the agenda for the 3D UML Heuristic Challenge and an overview of the contents of this paper, which provides supporting information for each part of the challenge.

### 1.1 Agenda

The agenda for the workshop session starts with participants forming layout and heuristic groups. The workshop is presented with background for the challenge and then groups select a number of heuristics to test in different ways and with different UML diagram types. It is assumed that the challenge will be undertaken by 4-5 groups of 4-5 people.

An overview of each section of the challenge and the estimated time is given below:

**Introduction, 15 minutes:** The workshop forms groups of layout or heuristic teams. The workshop participants are presented with an overview of Heuristic Evaluation and the list of 10 heuristics used for the challenge.

**Layout Problem, 10 minutes:** Two types of traditional complex UML diagrams, state machine and translated class diagrams, are presented. Also presented is the concept that sets of diagrams, within each type, are related and that 3D can be used to layout a set of diagrams to improve usability.

**Group Challenge, 45 minutes:** Each group is given their ”evaluation pack” described below. Layout teams select 2-3 unique guidelines to create diagram layouts against. Heuristics teams explore all guidelines applied to existing diagram layouts. The groups prioritise each heuristic and have a a fixed time to complete the challenge.

**Results, 20 minutes:** Each group then talks through their diagram layout and how the heuristics performed. The workshop as a whole discuss the best heuristics and best performing diagrams.

### 1.2 Evaluation Pack

The ”evaluation pack” contains material to enable groups to create a ”virtual virtual” 3D UML diagram layout. A virtual 3D UML environment does not exist to test diagram layouts, so groups will use pen, paper etc. to create representative layouts that could exist in a 3D environment. The ”evaluation pack” contains the following:

- Formatted Worksheets and cards (to write down heuristics and results):
  - Heuristics questionnaire
  - Heuristic evaluation worksheet (for heuristics team)
  - Task descriptions (for layout teams with state machine diagram and transformation chain)
- Pens
- Paper
- Post-it notes

## 2 Heuristic Evaluation

What makes a UML diagram ”good”? What makes one diagram layout better than another? What makes one diagram more useful than another? To answer these types of questions definitively, one approach is to define and test important variables through strictly controlled user (of the diagram) experiments on a sufficiently large population. In practice though, this approach is far too costly in time and money for most circumstances due to the number of variables to be tested and the need for user participation in experiments.

A less definitive but more accessible method for analysing usability, which avoids costly user testing, is Heuristic Evaluation. Heuristic Evaluation is a user interface design technique which uses guidelines or ”rule of thumb” analysis to allow experts to evaluate a user interface based on known issues [NM90].

As an example, one well known heuristic for diagram layout is to avoid edge crossings. Edge crossings are known to cause issues with users interpreting engineering diagrams and therefore the useability of diagram layout technique can be evaluated based how effectively edge crossings can be avoided. No user testing is required and empirical results can quickly be obtained through a few expert evaluators (or even automatically generating diagram metrics).

For new areas such as 3D UML, guidelines do not exist and heuristics such as edge crossing reduction, are not likely to be applicable in 3D space. Researchers are forced to use ”trial and error” and find out later through user experiments usability issues impacting on their visualization technique and the subsequent results.

Although very specific guidelines, such as edge crossing reduction, may not be applicable, well known higher level guidelines should be applicable. For the 3D UML Heuristic Challenge a number of high level useability heuristics have been reviewed by the authors and 10 heuristics have been chosen based on the experiences from implementing 3D UML tools (www.x3d-uml.org and www.gef3d.org). These guidelines are listed below:

- Speak the User’s Language
- Overview
- Be Consistent
- Zoom
- Relate
- Minimize the User’s Memory Load
- Filter
- Data-Ink Maximization, Data Density
- Details on Demand
- History

As these heuristics are generic to information visualization, participants are asked to interpret what they mean for UML diagrams and useability. The challenge is to take these refined heuristics test their validity for 3D UML diagram layout. Groups will either create diagram layouts that demonstrate the heuristics in action or apply the heuristics to evaluating existing diagram layouts.

A secondary goal of the challenge, is to challenge participants thinking on diagram layout. The 3D space enables layout and interaction techniques not normally associated with 2D interfaces. Without the 2D limitations and preconceptions, 3D layout presents a new way of thinking about what information a diagram should contain and how it should be presented.

## 3 Model Examples

This section presents examples of traditional complicated UML state machine and translated class diagrams which are to be tested in the 3D UML heuristic challenge. The two types of UML diagrams represent examples of sets of strongly related diagrams, where the combination of these diagrams in a single view provides valuable information to the user.

However to layout multiple diagrams, to show their relationships, presents a problem because of the limitations of traditional 2D representation. To remove the 2D limitations 3D can be used, which enables new and novel diagram layout techniques.

### 3.1 State Machine Diagram Example

In this example we present the UML state machine diagrams that are to be tested using 3D layout techniques against chosen heuristics. The state machine being tested consists of a hierarchal structure with a top state diagram and separate substate diagrams.

These diagrams are from RoseRT[1], which is a tool that generates code from state machines and is targeted at the real time embedded market. The “Traffic Lights” model is an example model provided with the toolset for training purposes. The model demonstrates the use of state machine inheritance to produce different variations of behaviour for different countries (e.g. North America and Austria). This model is the most complex provided in terms of hierarchy, with the Austrian variant containing 6 related state machine diagrams. Although it is considered complex in terms of the examples, actual models surveyed in industry are much more complex and are shown to have as many as 19 diagrams in a single state machine.

In Figure 1 we see how the structure is currently presented to the user. RoseRT presents these diagrams in a tabbed window approach, where substates are navigated to through double clicking on the diagram. With the use of 3D layout techniques, these diagrams can be presented in a more intuitive way and score better results in a heuristic evaluation. A view, called the navigator, shows the hierarchy of individual states (not state machine diagrams) and gives some clue of the layout of the state machines diagrams.

Visualization hints:

- In the evaluation pack, the name at the top of a state machine diagram is its substate name. This indicates how it relates to another diagram element, on another diagram. The name ”Top” indicates that it is the top level state machine diagram.

- Junction points (black dots on the state edge) which appear on the substate element (i.e. within a diagram), also appear on the edge of the diagram representing the complete substate. Note:These junction points generally appear in the same location at both levels of diagram, but not always.

- When considering the behaviour of a state machine, how events trigger transitions is very important. Events that are not handled at a substate level (i.e. don’t trigger a transition) are handled by the superstate (or its superstate) level or not by any state.

- When an event does trigger a transition, that transition could form part of a transition chain that spans multiple diagrams. To follow the behavior of one event may require the user to trace a transition to a junction point, find the related junction point in another diagram, then follow the transition connected to that.

- When considering a state machine, superstates and substates are closely related to each other. However only substates within the same diagram are closely related due to the direct transition links between them. Therefore state machines can be thought of has have a tree structure, where branches have no direct relationship to other branches.

[1] see http://www-306.ibm.com/software/awdtools/developer/technical

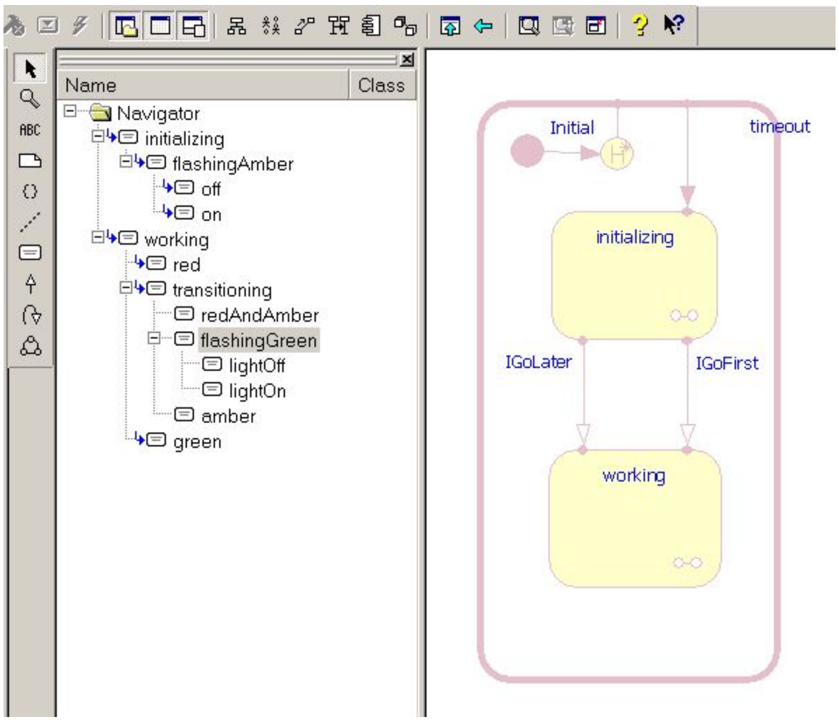

Figure 1: RoseRT "Traffic Lights" model. The tree on the left shows the "navigator" hierarchy of states within the state machine. The state machine diagram on the right shows the top level. Both "initializing" and "working" are composite states containing substate diagrams, as can be seen by the icon bottom right and the child states in the "navigator".

## 3.2 Transformation Chain Example

This example presents typical artifacts produced by a model-driven development process. An initial UML class model is transformed by several model transformations, resulting in a so called transformation chain. [2] The following artifacts are to be visualized:

1. Diagram 1: Initial UML class diagram representing a simple library model (Figure 2)

2. Transformation 1: The first transformation adds an interface to the model and an attribute ID to each class. The transformation consists of the following rules:

   (a) For each package, a new package with the same name is created and an interface "IUnique" is created.

[2] This example is taken from [PVSB08]. This paper includes a 3D visualization of the transformation chain, but we ask the participants of the challenge not to look at this paper before the workshop in order to avoid any "bad influence".

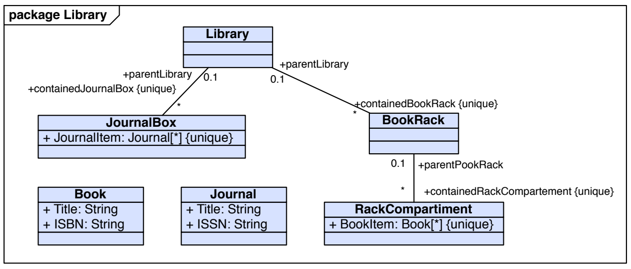

Figure 2: Initial UML class model/diagram

(b) For each class in the source model, a class in the target model in created with the same name. The newly created class implements the interface "IUnique" and an attribute "Id" is added.

(c) For each attribute/operation in the source class, an attribute/operation in the target class is created.

(d) For each association in the source model, an association in the target model is created.

3. Diagram 2: The second UML class diagram is the result of the first transformation applied on the first model (Figure 3).

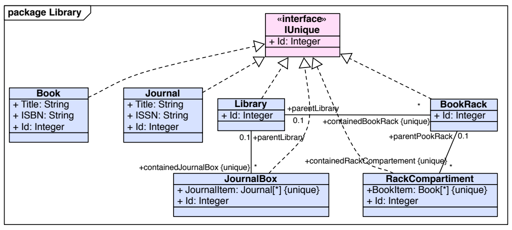

Figure 3: Second diagram, UML class model with IDs

4. Transformation 2: The second transformation adds getter and setter methods for each attribute and association. The transformation consists of the following rules:

(a) For each package, a new package with the same name is created.

(b) For each classifier in the source model, an appropriate classifier in the target model in created with the same name.

(c) For each attribute in the source classifier, an attribute in the target classifier is created. The visibility of the attribute is changed to "private" and appropriate getter and setter operations are created.

(d) For each operation in the source classifier, an operation in the target classifier is created.

   (e) For each association in the source model, an association in the target model is created. The properties of the classifier referencing the associations are made "private" and appropriate getter and setter operations are added.

5. Diagram 3: The third UML class diagram is the result of the second transformation applied on the second model (Figure 4).

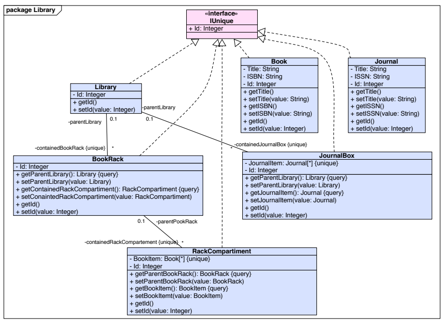
package Library
«interface»
IUnique
+ Id: Integer
Library
- Id: Integer
+ getId()
+ setId(value: Integer)
Book
- Title: String
- ISBN: String
- Id: Integer
+ getTitle()
+ setTitle(value: String)
+ getISBN()
+ setISBN(value: String)
+ getId()
+ setId(value: Integer)
Journal
- Title: String
- ISSN: String
- Id: Integer
+ getTitle()
+ setTitle(value: String)
+ getISSN()
+ setISSN(value: String)
+ getId()
+ setId(value: Integer)
-parentLibrary
0.1
-parentLibrary
0.1
-containedBookRack {unique}
*
-containedJournalBox {unique}
*
BookRack
- Id: Integer
+ getParentLibrary(): Library {query}
+ setParentLibrary(value: Library)
+ getContainedRackCompartiment(): RackCompartiment {query}
+ setConaintedRackCompartiment(value: RackCompartiment)
+ getId()
+ setId(value: Integer)
JournalBox
- JournalItem: Journal[*] {unique}
- Id: Integer
+ getParentLibrary(): Library {query}
+ setParentLibrary(value: Library)
+ getJournalItem(): Journal {query}
+ setJournalItem(value: Journal)
+ getId()
+ setId(value: Integer)
0.1
-parentPookRack
-containedRackCompartement {unique}
*
RackCompartiment
- BookItem: Book[*] {unique}
- Id: Integer
+ getParentBookRack(): BookRack {query}
+ setParentBookRack(value: BookRack)
+ getBookItem(): BookItem {query}
+ setBookItemt(value: BookItem)
+ getId()
+ setId(value: Integer)

Figure 4: Third diagram, UML class model with getters and setters

6. Transformation 3: The third transformation creates an Entity-Relatioships (ER) model. For each class, an entity is created. Associations are mapped to relationships, attributes are omitted here for simplicity. That is:

   (a) For each package, a new ER model.

   (b) For each class in the source model, an entity in the target model in created with the same name.

   (c) For each association in the source model, a relationship in the target model is created.

7. Diagram 4: The last diagram is an ER diagram. It is the result of the third transformation applied on the third model. Note that it might have been possible to apply this transformation to the preceding models as well, but it was applied to the third model here. In a more complex situation, the preceding transformation may have added other attributes or classes which have to be mapped, too (Figure 5).

Visualization hints:

- Transformations are to be visualized. They may be visualized by drawing traces for each applied rule. A trace connects all source elements with their target elements. Optionally a label is added indicating the name of the rule or other comments. Note that traces may connect *m* source elements with *n* target elements.

- Note that traces may be ordered hierarchically: A top level rule transforms the package, on the second level, classifiers are transformed, on the next level attributes and operations, and so forth.

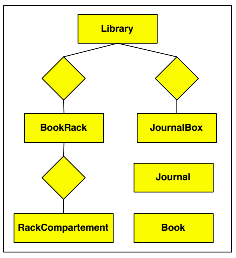

Figure 5: Last diagram, ER model

- The layout of the diagrams was created automatically here by some tool without taking preceding diagrams into account. This may be optimized, that is the layout of the 2D diagrams may be changed (if 2D diagrams are used).
- The transformations are rather simple here. This is usually not the case. That is: Do not rely on special properties of the transformations here. In MDD, a transformation may transform $m$ source models into $n$ target models, but usually, a single source model is transformed into a single target model (maybe using some additional information wich are not visualized). For the visualizing, you can assume simple 1 : 1 transformations.

## 3.3 Motivation and User Tasks

To be able to apply heuristics effectively we need to consider the user and what they are trying to achieve. For example, to evaluate compliance to the heuristic “Speak the User’s Language”, we need to know who the ”User” is to determine what their language might be. We also need to consider what information needs to be communicated to this user before we can evaluate if the way it has been communicated is appropriate. The following gives information on the user, their motivation and the task for each diagram type:

**State Machine Diagrams**

| | |
|---|---|
| **User (of the diagram):** | Embedded Software Engineer familiar with UML and RoseRT |
| **Motivation:** | Employed to develop and maintain a variety of controllers for different products. |
| **User Task:** | Develop and maintain controller firmware for traffic light behaviour. |

**Transformation Chain**

| | |
|---|---|
| **User (of the diagram):** | UML user who created the very first “domain model” |
| **Motivation:** | Understand the transformation chain, check for errors or bugs, understand the derived models. |
| **User Task:** | Use MDD techniques to speed up development, use prepared transformations (maybe third party transformations) |

## 4 3D UML Heuristic Challenge

This section presents an overview of the worksheets and cards given to teams and how they are to be used.

The Layout Teams select 2-3 heuristics, they then answer the questions about the heuristics (as per the example below) and then proceed demonstrating diagram layouts which represent the heuristics. Once each diagram is complete, the Layout Team complete the questions about the diagram layout.

The Heuristics Team answer all the question about the heuristics. Once complete they use the full evaluation sheet to test their heuristic definitions against 3D UML diagram layouts.

### 4.1 Questions about the Heuristic

Before the diagrams are evaluated, the participants are asked about the heuristics first. As the heuristics are generic they can be interpreted in many ways, we need to determine how the groups interpret them for 3D UML. For example, what does ”language” actually mean, in the heuristic below and having determined this, how do we quantify compliance?

The questions are presented on individual cards so that participants can easily treat heuristics one by one. A sample question looks like the following (ratings are from 1 – 5 with 1 = “Not applicable” and 5 = “Essential”):

**1. Speak the User’s Language** [MN90]
“The dialogue should be expressed clearly in words, phrases, and concepts familiar to the user rather than in system-oriented terms.” [MN90]

How do you think this heuristic description can be better stated for UML Diagrams?

---

What criteria would you expect to use in rating a diagram against this heuristic?

---

How important do you think this guideline is? ①②③④⑤

The following heuristics are used:

**1. Speak the User’s Language** [MN90]
“The dialogue should be expressed clearly in words, phrases, and concepts familiar to the user rather than in system-oriented terms.” [MN90]

**2. Overview** [Shn96]
“Gain an overview of the entire collection. Overview strategies include zoomed out views of each data type to see the entire collection plus an adjoining detail view.” [Shn96]

**3. Be Consistent** [MN90]
*Related:* “Show Data Variation, Not Design Variation” [Tuf01]

"Users should not have to wonder whether different words, situations, or actions mean the same thing."[MN90]

**4. Zoom** [Shn96]

" Zoom in on items of interest. [...] Smooth zooming helps users preserve their sense of position and context." [Shn96]

**5. Relate** [Shn96]

*Related:* "Rationale-Based Tasks" ("Concretize Relationships" and "Formulate Cause And Effect") [AS04]

"View relationships among items." [Shn96] "Users need to be able to relate data sets to the realms in which decisions are being made." [AS04] "[A] system can help bridge the Rationale Gap by clearly presenting what comprises the representation of a relationship, and present concrete outcomes where appropriate." [AS04]

**6. Minimize the User's Memory Load** [MN90]

"The user should not have to remember information from one part of the dialogue to another." [MN90]

**7. Filter** [Shn96]

"Filter out uninteresting items. Dynamic queries applied to the items in the collection is one of the key ideas in information visualization." [Shn96]

**8. Data-Ink Maximization, Data Density** [Tuf01]

*Related:* "Simple and Natural Dialogue" [MN90]

"Graphical excellence is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space." [Tuf01, p. 51]

"Dialogues should not contain irrelevant or rarely needed information. Every extraneous unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility." [MN90]

**9. Details on Demand** [Shn96]

"Select an item or group and get details when needed." [Shn96]

**10. History** [Shn96]

"Keep a history of actions to support undo, replay, and progressive refinement. It is rare that a single user action produces the desired outcome." [Shn96]

## 4.2 Questions about the Diagram Layout

In this section we layout the diagram with a particular motivation/user task in mind. The aim of the challenge is to create a diagram that will do well in all heuristics, but specifically be best in class for the heuristics the group has chosen (ratings are from 1 – 5 with 1 = "fails to comply" and 5 = "fully complies"):.

| |
|---|
| What is the heuristic your diagram demonstrates? ________ |
| How well does your diagram comply with the heuristic? ①②③④⑤ |
| Issues: ________ ________ |

## 4.3 Full Evaluation

For the Heuristics group the following questionnaire is used with the heuristics listed above (ratings are from 1 – 5 with 1 = "fails to comply" and 5 = "fully complies"):.

| Diagram | | |
|---|---|---|
| Heuristic | Rating | Comments/Criteria |
| Speak the User's Language | ①②③④⑤ | ________ ________ |
| Overview | ①②③④⑤ | ________ ________ |
| | ... | |

## Bibliography

[AS04] R. Amar, J. Stasko. A Knowledge Task-Based Framework for Design and Evaluation of Information Visualizations. In *IEEE Symposium on Information Visualization (INFOVIS'04)*. Pp. 143–150. IEEE Computer Society, Los Alamitos, CA, USA, 2004. http://doi.ieeecomputersociety.org/10.1109/INFVIS.2004.10

[MN90] R. Molich, J. Nielsen. Improving a human-computer dialogue. *Commun. ACM* 33(3):338–348, 1990. http://doi.acm.org/10.1145/77481.77486

[NM90] J. Nielsen, R. Molich. Heuristic Evaluation of User Interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*. Pp. 249–256.

ACM, New York, NY, USA, 1990.
http://doi.acm.org/10.1145/97243.97281

[PVSB08] J. von Pilgrim, B. Vanhooff, I. Schulz-Gerlach, Y. Berbers. Constructing and Visualizing Transformation Chains. Pp. 17–32 in [SH08].
http://dx.doi.org/10.1007/978-3-540-69100-6_2

[SH08] I. Schieferdecker, A. Hartman (eds.). *Model Driven Architecture — Foundations and Applications. 4th European Conference, ECMDA-FA 2008, Berlin, Germany, June 9-13, Proceedings*. Lecture Notes in Computer Science 5095. Springer-Verlag, June 2008.

[Shn96] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *1996 IEEE Symposium on Visual Languages*. P. 336. IEEE Computer Society, Los Alamitos, CA, USA, 1996.
http://doi.ieeecomputersociety.org/10.1109/VL.1996.545307

[Tuf01] E. R. Tufte. *The Visual Display of Quantitative Information* . Graphics Press, Chesire, CT, US, 2nd ed. edition, 2001.