# Mahler, Mondriaan, and Bauhaus: using artistic ideas to improve application usability

**Jonathan Seth Arnowitz**
Informaat, BV
Jacob van Lenneplaan 57
NL-3740 AT Baarn
+31 35 543 1222
jonathan_arnowitz@informaat.nl

**Eric Willems**
Compuware
Hoogoordreef 5
NL-1100 AX Amsterdam
+31 20 311 6222
eric.willems@nl.compuware.com

**Laura Faber**
Informaat, BV
Jacob van Lenneplaan 57
NL-3740 AT Baarn
+31 35 543 1222
laura_faber@informaat.nl

**Ruurd Priester**
Informaat, BV
Jacob van Lenneplaan 57
NL-3740 AT Baarn
+31 35 543 1222
ruurd_priester@informaat.nl

**ABSTRACT**

This paper addresses a strategy designed to handle the increasing and broadening interactivity demands in software. This paper specifically looks into using other interdisciplinary areas of art and music as an inspiration material for creating new forms of user/computer communications. The projects looked at are a project for the Dutch Social Security System, a work-flow driven administrative application and ending with an in-depth look at the Uniface 7 4GL interface which uses the Bauhaus as the jumping point for creating a new image-language.

**KEYWORDS**

GUI, Art, Design, Iteration, Interface Design, Bauhaus, Strategy, Methods

**INTRODUCTION**

Software and software development is becoming more complex and difficult. Yet in this day and age of both user-unfriendly bloatware products and stripped down java applets, there is an increasing demand for interfaces that can clearly and easily communicate with a user. Java applets are indeed pointing the way to one solution: breaking down applications in to smaller tasks and offer only key functionality needed to support those tasks. However with a broad range of software this strategy is not possible. These software packages can cover a broad terrain: programming applications for non-programmers, document registration systems for governments, complex work-flow administered business administration packages, to name but a few. These products because of their very complex requirements prohibit the use of traditional user interfaces.


DIS '97 Amsterdam, The Netherlands


There is indeed a perceived need to overthrow the traditional user interfaces. One article which appeared in a recent issue of ACM *Interactions*, [1] "The AntiMac Interface" addresses this very phenomenon. The article called for a virtual reality to replace the desktop or WIMP interface platform. The WIMP interfaces being defined as Windows, Icons, Menus and Pointers. There is also a perceived need for a new look and feel to interfaces. (For the rest of the paper we will call WIMP interfaces traditional interface, alleviating us of the derogatory WIMP anagram.)

Without traditional interfaces where can the user interface designer turn for guidance when charting new or different interaction territories? Besides standards, there are other alternative tried and true methods of communications, methods developed not by Apple, Microsoft or Xerox but rather by Michaelangelo, Klimt and Mondriaan to name but a few.

If a designer can take an expression or an idea that has proven to work in another media, they have a proven communication method. They only need to know how to translate the idea from the one medium to the next.

In usability testing, we have found that alternative forms of communications are successful when trying to communicate ideas or interactions that are not in the standard realm of basic computer applications such as word processors or spreadsheets where objects or documents are more tightly defined.

Before this paper comes across as anti-GUI standards, we should also add here that designers sometimes are forced to create non-traditional interfaces. Often, more often than we care to report, we must work within interface limitations that are dictated by product management who do not always have a user's best interests in mind. For example, when a product manager demands the inclusion of senseless

features that do not belong in a product, which nevertheless for marketing purposes must be included. In this situation a user interface designer is in the awkward position of either standing up for what they believe in or putting food on the table. If the designer chooses the latter, the designer must still find a design solution that still delivers an ergonomic product or as ergonomic a product as possible. This paper is also meant to give these designers other places to turn for help when traditional channels are purposely closed to them. There is also a lot of exciting literature on art and cognition that gives us valuable insights into this process, works by Robert Solso is an example of this work (6). These works however valuable, are out of the scope of this paper which is more focused on practice than theorey.

This paper will briefly discuss the idea behind the use of artistic ideas in user interface design. The paper will begin with two short illustrations based on previous projects performed by Informaat. Then the paper will go over a case study of this idea in the development process: an implementation of a Bauhaus metaphor to improve the usability of Compuware's Uniface 4GL application.

## GUI STANDARDS VS. CREATIVITY

The main limitation with applications that follow the traditional interface guidelines is that they all employ a standard group of objects and in a limited set of contexts: usually file, document or object contexts. In most applications, these limitations are actually an advantage because it defines how most users have already experienced graphical user interface software. For example, most users know check boxes because everyone uses them in GUI applications. How checkboxes behave differently than radio buttons needs no explanation because users have expectations from previous usage how a radio button will behave and how a checkbox will behave. Restricting user attention to tangible objects creates a more user friendly application by inviting the user to directly manipulate objects using real-world metaphors. Furthermore by restricting a users focus to only the essential elements, a user can achieve their task easier.

The problem comes when an application introduces ideas or follows product demands or guidelines that cannot be easily or clearly depicted in traditional interfaces.

When this problem happens, communication problems occur, and with these problems comes a lack in clarity and consequently a user unfriendly product. To solve this problem the designer must look elsewhere than standard interface design for the solution. One source of solutions is to look where someone else had solved a similar communications problem. Then try to adapt their solution to the design problem.

## THE USE OF ART IN INTERFACE DESIGN

To achieve clarity of communication, a User Interface Designer can look to other people who have used media in order to communicate. Specifically, when confronted with a problem not addressed in traditional interfaces, it is a logical step to look to someone who has addressed that problem in another media. People who have used media to communicate can include industrial designers, commercial artists, painters, sculptors, musicians etc. If you identify an artist who has addressed a problem that you are confronted with, then that artist is presenting a proven method of communication. A method you can implement in your own design. The result will be a non-traditional interface but one with a proven communicability. This artist can come from your own experience, but it would be better if the artist could come out of part of a user study.

The idea is not to recreate the Sistine chapel in an VGA monitor, nor is it to include the soundtrack of Beethoven's Third Symphony with your software. The practice we try to employ is look at a particular artists and see why his employment of communication works and how can that be developed for an appropriate visual clue in a piece of software.

## PAST EXPERIENCES: MAHLER AND MONDRIAAN

In previous projects this method was employed to solve communication problems in interfaces for software for the Dutch Social Security, as well as a workflow administrative application software. These solutions employed ideas from a variety of sources such as music, sculpture, and artistic movements. Quickly, I will give these two example we had already implemented before describing the Uniface example.

### Music encompassing the world

At the end of the Nineteenth Century many composers were struggling with where to take the symphonic structure. It seemed as if the romantic era had exhausted the possibilities of the orchestra: everything that could be done had been done. Beethoven, Berlioz, and Wagner could be thought of as the Apple, Microsoft and Xerox's of romantic music. These musicians, Wagner explicitly, claimed orchestral music had been taken as far as it could possibly go. This would appear to leave little room for maneuvering for the late Romantics. Wagner's music overloaded as it sometimes is with features no listener will ever use, could be associated with the phenomenon in software called bloatware. This feeling that it's all been done, that all the discoveries had been made lead many developers, such as Brahms and Richard Strauss, to resort to simply reusing the forms their predecessors had passed down.

Gustav Mahler's solution to this problem was the expansion

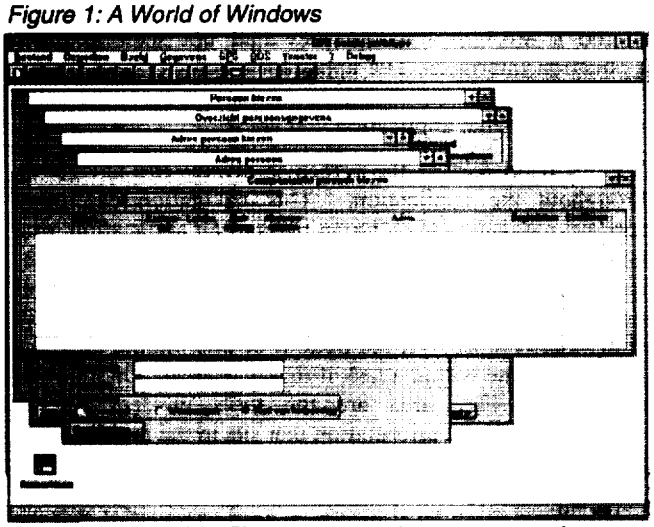
of the orchestra and the expansion of musical expression. In the orchestra, Mahler increased the number of players as well as introduced a number of new instruments and instrumentation to create new sounds and produce new musical effects. According to Mahler, "The symphony must embrace everything, the symphony is the World!" Mahler took the over-expanded romantic symphony and dared to expand it further. The music of Mahler's reflects this world by using real-world sounds such as bird calls, forest sounds and transcribing them to music. Mahler added many new instruments to the orchestra ranging from a glockenspiel to a large wooden hammer. His symphonies are large in scope. Yet they all maintain concise communication due to very inventive extraordinarily sparing use of the expanded orchestra. Mahler would call on a wide variety of instrumental combinations to develop themes passed from one to the next. Thus he used the instruments only when he needed them, resulting in an unprecedented clarity of sound with a gigantic orchestral force.

### Software encompassing the world

In a seemingly unrelated event at the end of Twentieth Century, a software company was struggling with the development of a software package that could handle the extra-ordinary demands of data accessibility for the Dutch Social Security system. The workers of the Dutch Social security system had to access an enormous amount of data very quickly for a variety of procedures that looked similar but were never identical. Furthermore, a product requirement (even though it was discredited in user interviews) was that all information for a record had to be available from the same screen. This meant forcing the interface to confront the user with an enormous and unreadable amount of information.

In order to tackle this problem of accessing this incredible amount of data, this software company was struggling with GUI standards to come up with a solution that looked like Figure 1.

*Figure 1: A World of Windows*

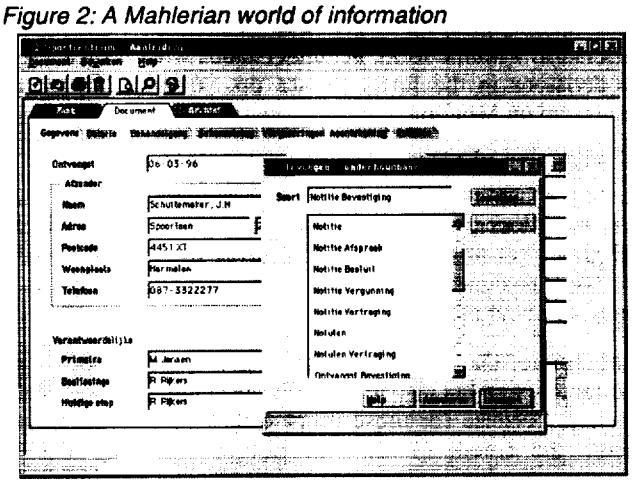


The problem with Figure 1 was the movement between windows was not all that easy. As soon as a user clicked on a window title in the middle of the screen they could no longer read the other menu titles, or even lost site of smaller windows altogether. Furthermore all sorts of window navigation techniques such as a Window menu etc. did not improve upon the usability of the application. Tested users, still wanted to use their memorized three code short cuts to jump between screens instead of the mouse or any other standard GUI solution.

### Finding a solution in a strange place

Coincidentally, one of the user interface designers was studying the Mahler's aforementioned solution to the crisis in music in the late 19th Century. The designer likened this problem to the design team's problem. Mahler's solution to expanding the symphonic structure was then applied to the graphical user interface of the Dutch Social Security software. Mahler's ingenious combinations of instruments, including inventing new instruments as he saw fit seemed an appropriate source of inspiration. The implementation meant that instead of widening the orchestra, the designers increased the amount of data and both the number and types of widgets that would be displayed on one screen. The amount of information, however, would be selectively grouped together making the data easy to read. An example of this practice is the tab within tab widget pictured in Figure 2. A hierarchical tab set was used. The first level organized data according to general information or information specific to a particular law or request form. Then within each grouping data was further grouped into smaller logical groupings. These logical groupings when possible followed the same flow as the application forms that people would have filled out by hand.

*Figure 2: A Mahlerian world of information*



The Mahlerian solution for this seemingly unwieldy administration software was to expand organizational ideas as far as they could go. In our implementation a Mahlerian interface looked like Figure 2. In Figure 2 the interface offers a single click access to 10 of the most used screens for a related task and double click access to up to 80 other related screens. After low-fidelity prototype testing this solution on users, users disregarded their three key shortcuts 70% of the time and more significantly, users found information 75% quicker.

### GUI AND *DE STIJL*

Another project where artistic help was sought was in an interface design project for administration packages by a

software house in the Netherlands called Multihouse. Multihouse wanted not only an interface that could be used for all their current and future administrative application packages, but they also wanted their own look and feel. Furthermore, they wanted a design that reflected their Dutch heritage without resorting to windmills, clogs or cheese. A further requirement was that the design solution had to fit in both in stand alone usage as well as in a work-flow implementation.

Again, the problem was coming up with a solution that could house both simple as well as complicated applications. Easy access had to be made to an unpredictable variety of information.

### De Stijl philosophy

De Stijl was a school of art that flourished from c. 1920-1930, primarily in the Netherlands. De Stijl was developed as an answer to the challenge of how to clearly communicate in an abstract form which is a complete disassociation from the natural world. Their use of color was limited to the primary colors red, blue yellow, and white, black and grey, since these colors are the most unnaturalistic colors. As stated in [8] "De Stijl is a world of color and form, rectangles and lines." The idea was a form of communication that was pure abstraction totally freed from any naturalistic expression. This sounded suspiciously like a rudimentary adjunct rule to User Interface Design.

Indeed, what is interesting for user interface designer is their use of abstract forms for communication. Abstract forms that allow the user to react abstractly but not in a literal communication. The user makes no naturalistic association with the form, yet understands what is meant by the divisions in the form. (6, see pp. 261-69 for a good discussion on Mondriaan in this regard.)

### De Stijl's use of color

De Stijl's use of color is judicious and limited: just the primary colors plus our favorite standbys of white, black and grey. Yet this very limitation gives the graphical user interface designer the freedom to use color without worrying about creating an odious combination of colors.

Furthermore, De Stijl use of colors is not only modest for interface design but also technically speaking very practical. Practical because for those users with VGA monitors can handle the colors of De Stijl just as easily as SVGA monitors. So that modest clients with 16 color monitors can still work with an interface based on the color and form philosophy of De Stijl.

### De Stijl's use of Form

De Stijl's use of form, the rectangle, is it's most recognizable symbol. Graphical User Interfaces are almost 90 percent lines and rectangles. Windows are just rectangles. Fields are rectangles. Buttons are mostly rectangles. Icons all fit in a rectangle shape. Lists are rectangles within rectangles. Spreadsheets are almost a virtual Mondriaan design in themselves. All these rectangles and lines, invite an expanded use of these forms into a user interface design.

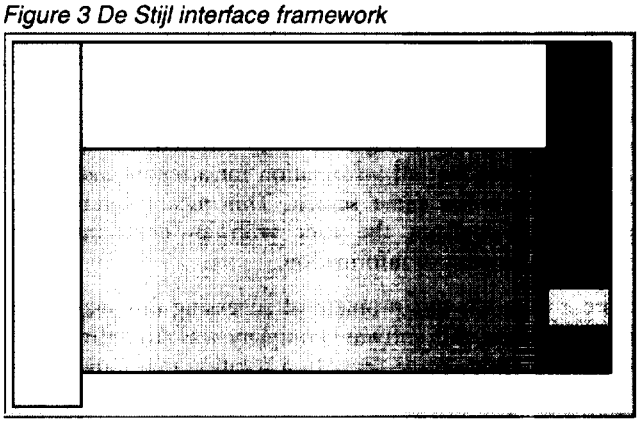
The use of color can be used to communicate the characteristics of a given form. For example give a red shade to the form of destructive commands. Make a yellow shade to the form of a non-destructive command structure.

In this use of color and form, a user always has a clue as to what the relationship and the characteristics of a part of the interface, even if they have never seen the interface structure before. With color and form the recognizable metaphor a designer has chosen has a new aid. There is more room for extra-ordinary functions without the fear of miscommunication because there is a new layer of communication added to the interface namely De Still's use of both color and form.

### The New application: De Stijl Interface

Figure 3 actually is not a Mondriaan painting but the interface's background. Figures 4-5 are resulting applications that have been implemented by Multihouse Automatisering in the Netherlands in their database application generator. Users, as space and taste permits, can click and drag to change the proportions to see more or less of different parts of the screen. A user could also change to color settings from De Stijl to grayscale or 2 more traditional windows color settings.

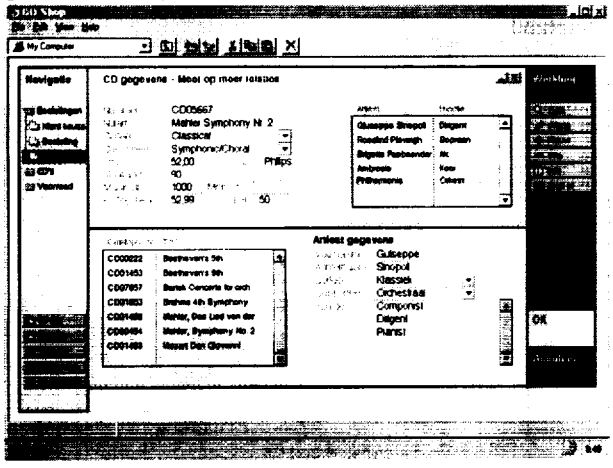
*Figure 3 De Stijl interface framework*



Interestingly enough, users also though this interface offered a level of freedom they had not seen in other application interfaces. The application actually uses a very strict division in form that does not vary, it stays the same regardless of what application you are in.

The left hand part of the screen is always the navigation throughout the application. This navigation can be a standard navigation to different parts of an application, or a list of work a user must perform that is generated by a work-flow module. The middle part is always the information or manipulatable objects, always with the recognizable grey or white backgrounds. At the bottom of this middle area is a way to navigate within a task if that is needed. On the right side in the blue area are the tools needed for the manipulation of the information that sits in the middle white area. But always, regardless of what type of screen as you see in the example, Figure 4, the focus is always on the information. High level prototype testing revealed that the yellow and blue areas only commanded the user's attention when they need access to them. De Stijl colors and forms, the layer is completely invisible to the user.

*Figure 4: Cd shop from the Multihouse prototype*



**CASE STUDY: PROJECT: BAUHAUS & UNIFACE**

Uniface is a very powerful platform/database independent application generator. With Uniface, a developer can write an application that can be deployed on almost any platform using almost any databases and in any combination. The problem with this powerful 4GL was with the conversion was made from character based terminal interface to a Graphical User Interface. The original developers at Uniface, at the time, had no User Interface Designers. Consequently, they preceded to simply port their current character application. This resulting application caused numerous customer complaints, strengthening of the position of the competition, etc. Then a new company, Compuware, took over the Uniface product. As a result, the next version of Uniface, 7.0, did involve a User Interface Design team. The team's first step in the complete restructuring of the application along traditional cognitive ergonomic principals as well as creating an attractive graphical design for the product.

The project began with a brief survey of the application. This included consultation with Developers, End Users, Compuware's Management and Marketing departments. The survey also included technical and functional evaluation of different possibilities for changes, i.e. what changes in the interface could be accomplished in the current development cycle. After the survey, the scope of the project was established.

For the scope of this project, we concentrated on how design and other use of graphics could improve the user experience. The identified problems were:

1. Users didn't understand the underlying philosophy and therefore had trouble understanding the product
2. Users used the poorly designed interface of Uniface 6.1 as an example of how to build applications, which resulted in many unhappy end-users of Uniface products.
3. Although the Uniface 6.1 release featured standard Windows interface widgets, users could not figure out how to properly implement them.
4. Users worked in an unclear mental model.
5. Users found interaction style inconsistent and sometimes arbitrary.

Our task was to, as much as possible, solve these problems through better communication as the total re-evaluation of the interface would have to wait for the next release of Uniface. For now, given the current situation of Uniface and the time limitations in which we had to work, we set as our goal to: Improve the user experience.

To improve the user experience, we set out as our tasks to:

1. Make the interface more engaging
2. Make the interface more pleasant to look at
3. Use the opening screen or workspace of the application as a means of communicating to the user the abstract hierarchical structure of Uniface
4. Improve the use of icons and use them as a communication tool for the user to help them understand how the application works
5. Improve on-line help and screen layouts
6. Test all changeable parts of the interface for usability and change accordingly.
7. Use results from usability test to improve interface, correct visual hints, and improve documentation both on-line and in print to help users in areas we could not fix.

**The Team and the Method**

The team included three designers and three developers. The idea was that the designers would propose solutions and the developers would implement them. However, the actual approach taken was more a team effort, with the entire team involved in all phases. The developers helped with a design scheme that was both accurate for the end users and technically implementable. The designers helped to make sure the implementation of designs were as user friendly as possible, especially when problems were created that needed both new ergonomic and technically feasible solutions. Whenever the two were at odds, the technically feasible answer won short term but the ergonomic solution was kept for future development.

The team members were: User Interface Designer: Jonathan Arnowitz. Technical Designers/Illustrators: Ruurd Priester and Laura Faber. Lead developer: Eric Willems. Editors developer: Eric van Hinte, Workspace and Tree widget developer: Niels Schnieder.

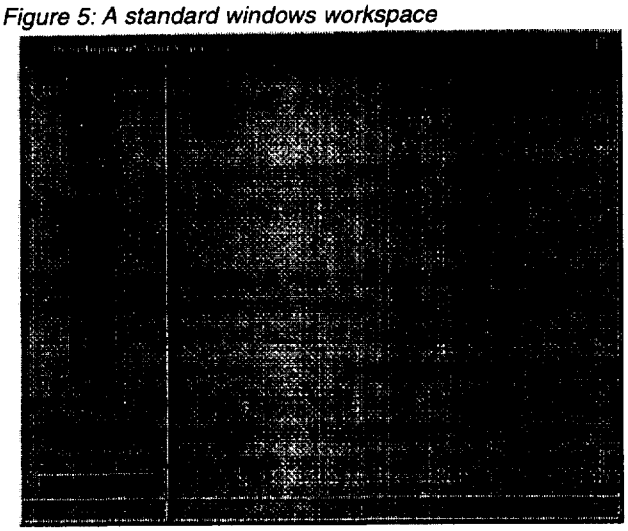
Many other people from development, documentation, help desk, sales, marketing, and the administration were also consulted so that everyone was informed not just of the work but also the development process. With this inclusive style of development, everyone not only understood what was being done but why. This communication empowered these parties to not only question the why, but also participate in a solution along the philosophical lines of what were trying to develop.

We also tried as much as possible to interact with the actual users of Uniface and get their feedback on the application. Through personal interviews we tried to assess how they understood the product, how they used it, and what kinds of tasks they were primarily involved in. This way we could

establish where the communication failure was in Uniface and what were the most important tasks to offer users in the opening workspace screen.

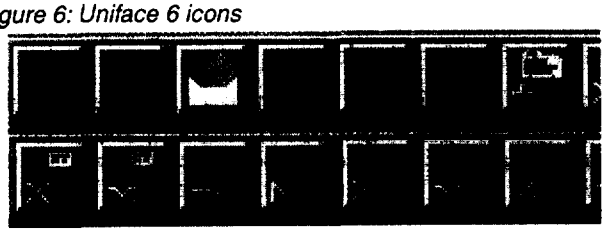
The challenge was first to find a concept, a style, that could help both communicate to the user how Uniface worked and also give the user a basis for a good mental model of the application. We had originally played with many different kinds of windows solutions. Figure 5 being an example of a possible Windows95/Macintosh like solution that was considered. However what was missing was the communication of the underlying philosophy and structure of Uniface.

*Figure 5: A standard windows workspace*



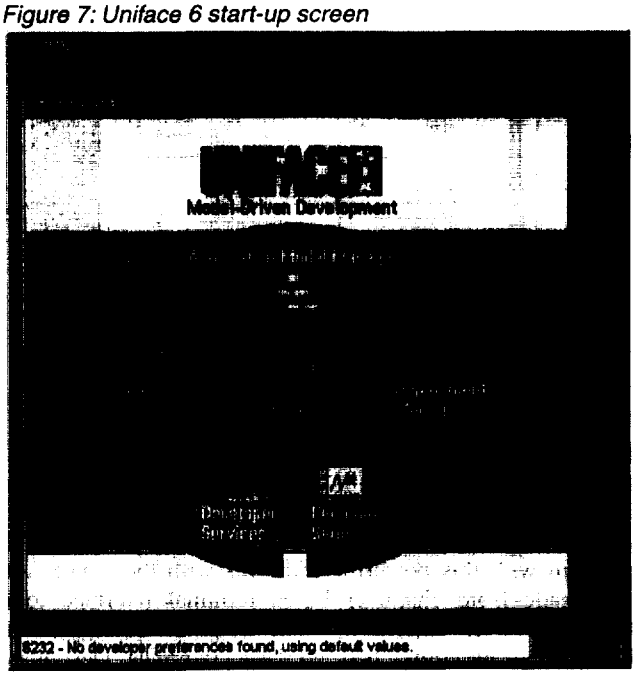
In order to allow the Uniface user to understand this structure and philosophy, the team would develop a conceptual model based on 'building.' Building so that users would look at building an application in Uniface in the same way as building a structure.

*Figure 6: Uniface 6 icons*



This metaphor was to be implemented in a new family of Uniface icons as well as for a new workspace. You see the old versions of Uniface icons in Figure 6 and the original start up screen is Figure 7. The Uniface 6 icon meanings were not always clear nor the functionality they offered. The workspace was abstract, in fact it was too abstract and did not offer the user a clear picture of what they were doing. For example, even though every section around the red button was clickable, the red button itself could not be clicked. This caused many users to complain that their "Repository button was broken." The search for ideas to change these icons and the workspace promptly sent the design team into their art books, software text books, other sources, for example museums, the web, etc. in search of something that communicated in a way that suggested building and could make an abstract structure easy to understand.

*Figure 7: Uniface 6 start-up screen*



We assembled images that we found spoke to the idea of communicating building with the following images.

One example, Figure 8, came from a set from the Dutch Opera's recent production of Der Fliegende Hollander. It was an example of how with simple stark lines it was possible to suggest a three dimensional space, to suggest a construction. This set also made use of small pictures on the wall which mirrored the frame structure of the rest of the set. We thought with this use of space we could actually play with the room or space itself in the representation of the workspace.

*Figure 8: Act 1 set of Der Fliegende Hollander*

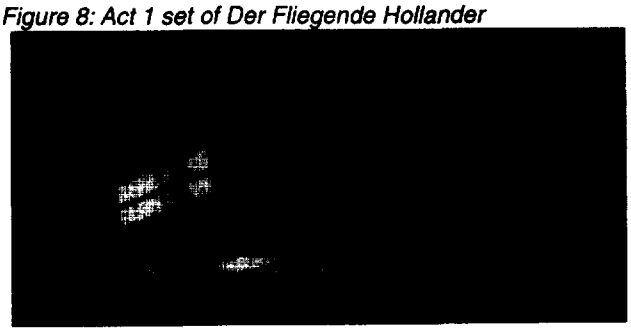
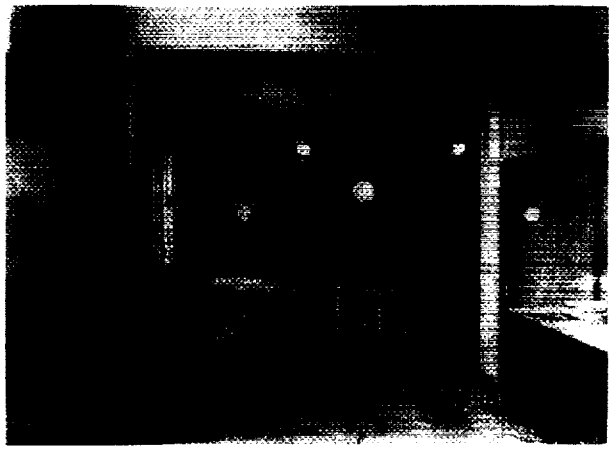


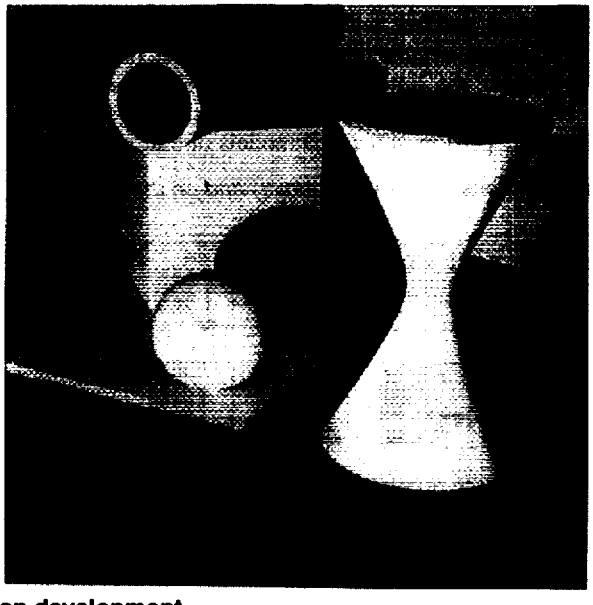
Figure 9 was a room taken from a book on Bauhaus as was a Zeitung kiosk (not shown here). The office gave a nice restful feeling, but at the same time the strong lighting suggested intense work. In Figure 10 we thought we found the ideal model to follow for our abstract building blocks. This picture used simple shapes yet they could also communicate structure.

*Figure 9: Bauhaus office*



These images became our basis for the development of the Compuware Bauhaus icons and workspace.

*Figure 10: Bauhaus objects*



**Icon development**

The design team had decided to go with the Image Language of Figure 11: the use of small simple objects that can be arranged hierarchically. The object hierarchy consisted of five main object. The highest level was called the *Model*, followed by the *Entity*, *Component*, *Field*, and then lowest level: *Trigger*. Understanding the hierarchy was very difficult for users because items lower in the hierarchy could be defined by any object above it. For example, the lowest level object, a *trigger*, could be defined by any higher level object. By selecting simple objects and combining these relations could be made clearly simply by using the icon. Consequently, when a *trigger* icon is pre-defined by a parent object, the *trigger* icon is showed with it's parent object. The resulting icon would clearly communicate to the user what the object relationships were.

What we further liked was that the use of simple abstract objects had the advantage that there was less chance of miscommunication with the user. With abstract objects, we didn't have to worry that the user would incorrectly interpret an image. Had we used a literal metaphor we would have had to use a metaphor that was known to every Uniface user, something very difficult when dealing with an international audience of 4GL developers. Furthermore, in our usability testing, we found that users easily attached their own meanings to the simple abstract images. Because the objects themselves were abstract, these meanings were never 'incorrect' interpretations as Solso relates in (6) p.268.

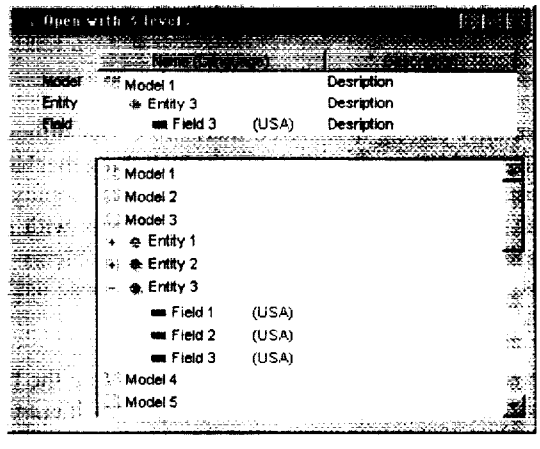
*Figure 11: The basic objects and combinations.*

Figure 11 pictured right to left the icons for: Model, Entity, Component, Field, Trigger.

Another requirement of the icons were that they be 14x16 pixels, therefore any complex imaging was also out of the question. The reason they had to be 14x16 pixels is that they were always used in object lists, which for technological and anthropological reasons dictated that they be 14x16 pixels. One example of reuse was that users had a new capability in Uniface to make aliases, and they wanted to be able to make aliases in the "desktop of Uniface. The type of alias would have to be determined by creating a 32x32 icon alias shell, where the type of object could be pasted into that shell so the user would recognize what object the alias pointed to.

Furthermore, the use of simple objects would help in the communication of the Uniface structure. Because Uniface had a very complex hierarchical structure, by using simple objects these objects could be easily combined. These combinations of objects could indicate special flavors or properties in the icons without creating a complex or confusing image.

*Figure 12: Combinations showing complex relations in a simple manner*




The basic objects Ruurd Priester developed are in Figure 11. The combination of these icons to convey the hierarchical relationships in Uniface is demonstrated in Figure 12. The

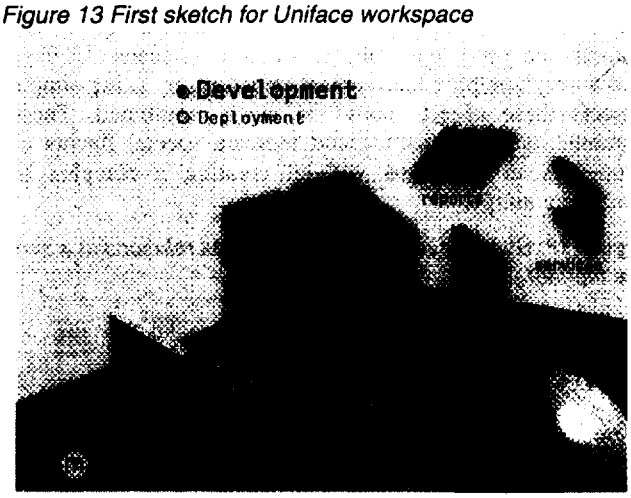
difficulty in communicating the hierarchy was that elements in different levels of the hierarchy could share the same lower level objects. This resulted in user confusion. However with the following combination icons, the complex relations were reduced to understanding very simple relations that were echoed elsewhere in the application as well.

### Workspace

The workspace became a very important keystone to understanding the icons. The simple shapes had to be three dimensionally rendered and rendered in such a way that the user would immediately see a building block type metaphor. From the workspace design, users should also see an arrangement that also reinforces the communication of Uniface's hierarchical structure. For this reason the Windows version of the workspace shown in Figure 5 was not considered.

Instead, we asked Laura Faber, the team's workspace designer, to implement a workspace where the ideas set forth in Ruurd's icons would be released in a sort of three--dimensional space as in the image of Figure 9 and 10. The first workspace which we used in our usability testing is pictured in Figure 13.

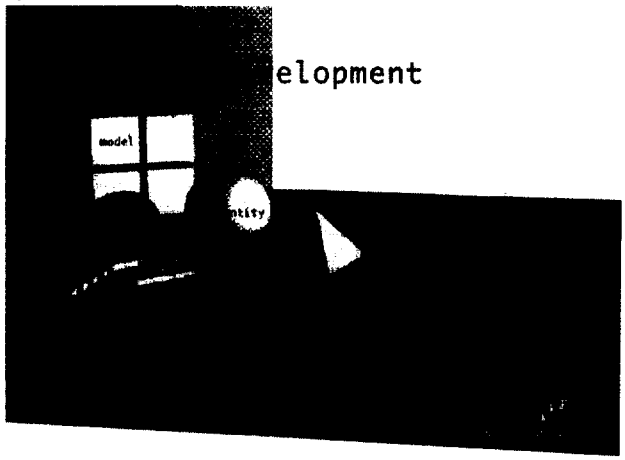
*Figure 13 First sketch for Uniface workspace*




The usability testing discovered there were user difficulties with this workspace. For example the switching mechanism between two workspaces, Development and Deployment was not very clear. Also there were problems with some terminology used. However, when these problems were addressed in the redesign of the workspace, these usability issues disappeared.

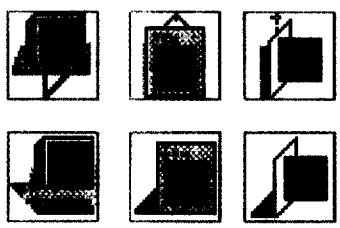
Furthermore this workspace really didn't capture the heavy building ideas, nor accurately convey the hierarchical structure as we wanted. Our Usability testing also showed that since we split the application between two workspaces that we had to make the switching mechanism between the workspaces clearer.

*Figure 14: Final workspace design*




The resulting workspace design is depicted in Figure 14. This workspace accurately communicates the strict hierarchical structure of the Model, entity and Trigger, and the more loose position of Forms, Services and Reports which share entities and can share or have their own triggers. Furthermore they have no direct relation to the Model, unlike the Entity. In the lower right hand corner is the switching mechanism to the other workspace, a miniature depiction of the 3-D models of that workspace.

*Figure 15 Icons on the wall (above) and on the workspace*

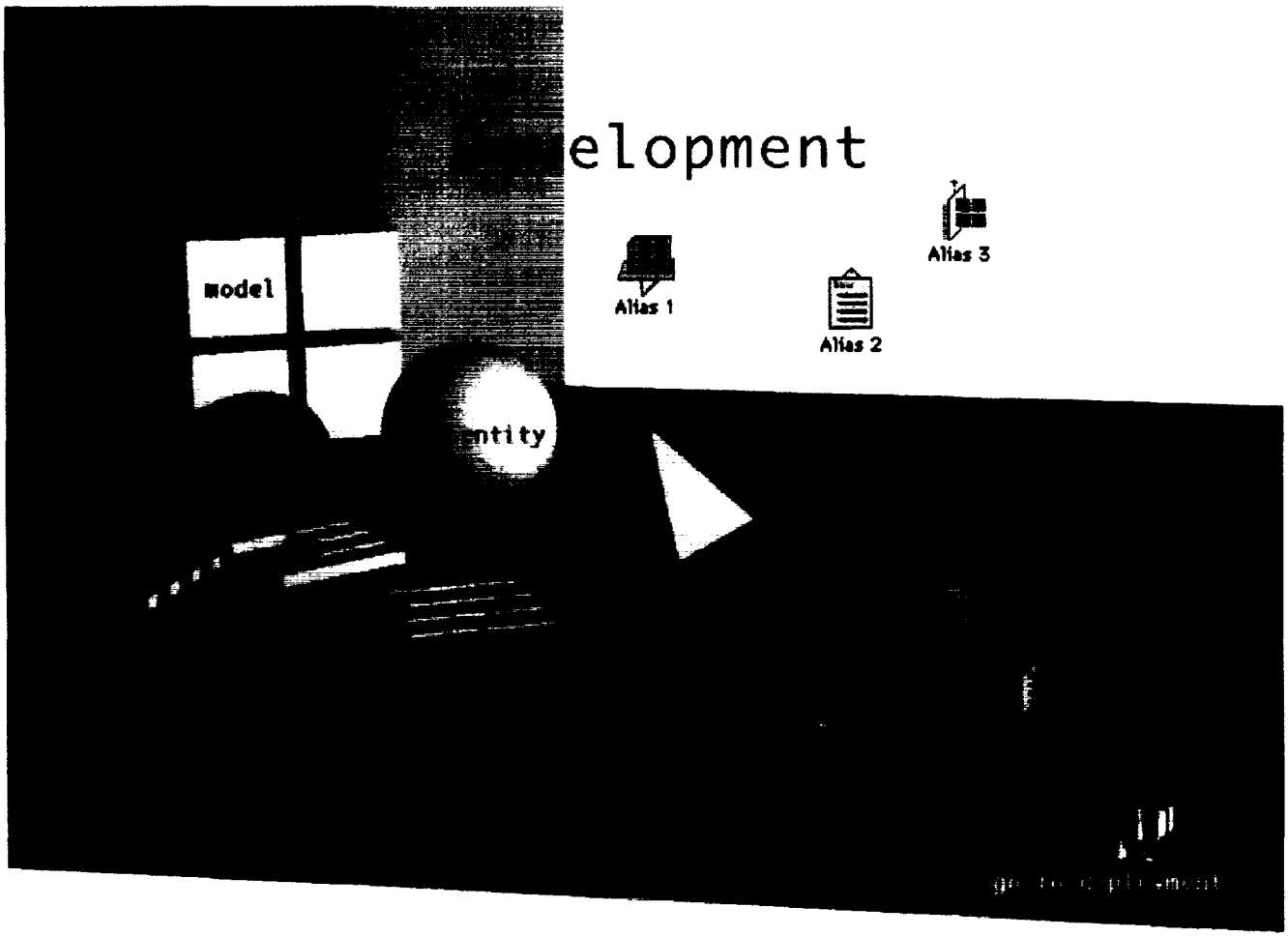


This left still the problem of getting the user to play with the three dimensional space. The user can make three different kinds of aliases or shortcuts in Uniface: to a specific object, an object editor, and a part of a specific object. The aliases were developed so they looked differently when a user dragged the alias on the desktop, or dragged the alias to wall. The different aliases are shown in Figure 15.

Figure 16 shows the workspace with aliases both resting on the desktop and sitting on the wall.

This resulting workspace manages to suggest the hierarchical structure of Uniface by showing the Model, Entity, and Trigger hierarchy in a stark three dimensional alignment. That there are three objects, Report, Forms and Services, that share objects from the hierarchical structure, but are not part of that structure, is suggested by their laying in front of the hierarchy, and grouped together. When these objects are viewed by users, they understand immediately that the flat 14x16 icons are their two dimensional representation. It is interesting to note, that when asked after the test, many users said all the icons, even the two dimensional objects were three dimensional. We feel this is because the user could easily adapt their mental model to fit in the three dimensional simple abstract model

*Figure 16: Uniface 7 workspace with icons*




we present in the workspace.

## SUMMARY

Compuware had a challenge for user interface communication that could not be easily addressed in a standard GUI interface. We have found that it is useful in seeking solutions from a variety of sources, whether interface designers, artists or musicians—anyone who has struggled with the same communications problem can teach us their solutions. We have likewise found that by adapting their solutions to interface design, we can create an application interface that communicates the user in ways that are not possible by just using Windows Icons Menus and Pointers.

## REFERENCES

1. Genter, Don and Nielson, Jakob. "The Anti-Mac Interface" in Communications of the ACM, vol. 39 No. 8, Pages 70-82

2. De La Grange, Henry-Louis. Mahler, Vol. II: the Years of Challenge, Oxford University Press, New York, 1995

3. Hoffmeester, GH, Kemp, J.A.M., and . Blankendaal, A.C.M.. Sensuality in Product Design: a Structured Approach. *Proceedings of ACM CHI'96*

4. Laurel, Brenda. Computers as Theater Addison Wesley, Menlo Park, 1993.

5. Mullet, Kevin and Sano, Darrell. *Designing Visual Interfaces: Communication Oriented Techniques.* Sun-Soft Press, Mountain View, 1995.

6. Solso, Robert. *Cognition and the Visual Arts.* MIT Press, Cambridge, Mass., 1994.

7. De Vrienden van de Nederlandse Opera. *Opera Jaarboek 1996.* the Netherlands, 1996.

8. Warncke, Carsten-Peter. *De Stijl 1917-1931.* Taschen-Librero Nederland, Hedel, the Netherlands, 1990.

9. Droste, Magdalene. *Bauhaus, bauhaus-archief,* Benedikt Taschen Verlag GmbH, Cologne, Germany, 1994