Redesigning Design/CPN: Integrating Interaction and Petri Nets In Use

Paul Janecek, Anne V. Ratzer, Wendy E. Mackay
Department of Computer Science, Aarhus University
Aabogade 34
8200 Aarhus N., Denmark
{pjanecek,avratzer,mackay}@daimi.au.dk

Abstract

This paper describes the redesign of the interface for Design/CPN, a graphical editor for building, simulating, and analyzing Coloured Petri Nets. One of our goals in the redesign is to explore how recent advances in graphical interfaces and interaction techniques can improve the editor's support for CP-net designers. This requires an understanding of Petri Nets In Use, our term for the collection of guidelines, work styles, and interaction patterns that influence the way designers build CP-nets. This concept is central to our design framework, and has guided our design process.

In this paper we describe our design framework and give an overview of our design process, a series of empirical studies, brainstorming sessions, and design exercises. We then follow two key issues (setting graphical attributes and alignment) through our design process, and describe how the Petri Nets In Use perspective influenced the evolution of toolglasses and magnetic guidelines as solutions for these issues in the new tool.

Keywords

Participatory Design, Coloured Petri Nets, Design Process, Toolglasses, Magnetic Guidelines

1 INTRODUCTION

A Coloured Petri Net is not simply a mathematical formalism; it uses a graphical representation because it is easier for people to understand. The Design/CPN tool, designed in the 1980's, recognized this and provided a state-of-the-art graphical editor to interactively create, simulate and analyze CP-nets.

The CPN2000 project is a complete redesign of the Design/CPN tool developed at the University of Aarhus. The project draws from four areas of expertise within the computer science department: Coloured Petri Nets, object-oriented languages, human-computer interaction and advanced interaction techniques. The project, funded by Hewlett-Packard, began in February, 1999, and involves participants from each of these research areas, with a core group of 11 people involved in the design.

As we begin the user interface design of a new CPN tool, code-named CPN 2000, we have shifted our focus from Coloured Petri Nets to "Petri Nets In Use": How people borrow from existing Petri Nets, rearrange objects for practical and aesthetic reasons and iteratively test and modify their designs. This paper describes our attempts to identify the critical elements of Petri Nets In Use and incorporate them into the design of CPN 2000. Our research approach includes observation of CPN users at work, various design activities that capture processes associated with designing Coloured Petri Nets and the integration of recent advances in graphical user interfaces and interaction techniques. Our goal is to provide CP-net designers with a significantly more effective design tool.

The structure of the paper is as follows: First, we explain our design framework, including an techniques. Second, we give an overview of the design process. Finally, we follow the evolution of two design issues (setting and managing graphical attributes, and alignment) through the design process. We look at how these ideas developed from a problem in the current tool, or an idea about some new functionality, over isolated ideas of smaller interaction patterns, into a more refined design.

2 DESIGN FRAMEWORK

Our design framework is composed of both technology and use domains. Factors that influence the architecture of the tool, such as its functionality, the graphical interface and interaction techniques, are part of the technology domain. Factors related to the use of the tool, including design guidelines, overall work styles and interaction patterns, are part of the use domain. Our design framework is based on developing an in-depth understanding of these two domains and the interaction between them to guide the design of the new CPN editor.

2.1 Design principles

There are three key principles that we use to guide and evaluate the design of the overall graphical interface:

1. **Reification:** The process of turning interaction patterns into first class objects. Thus, commands can be made accessible as instruments, combinations of properties can be turned into styles and the selection of multiple objects can be tagged and accessed as groups.
2. **Polymorphism:** Similar operations may be applied to different objects. Thus, various objects can be cut, copied or pasted, any operation can be undone, and operations that apply to a single object can be applied to groups of objects.
3. **Reuse**: Previous commands and the responses by the system can be reused by the user. Thus, input may be reused as the "redo" command and macros, output may be reused as input to other commands, and new commands may be created out of existing commands and a partial list of pre-defined arguments.

The reification principle has strongly influenced the design of the new tool. For example, in the current tool a set of objects are aligned by applying a “vertically align center” command, but additional objects cannot be added without reselecting the aligned objects. In the new tool, this alignment command is reified into a **magnetic guideline**, a visible first-class object which is continuously accessible and modifiable. New objects can be attached to the guideline, and moving the guideline also moves all the objects attached to it.

2.2 Interaction Techniques

The interaction techniques of most traditional graphical interfaces use some combination of Windows, Icons, Menus, and Pointing (WIMP). Although these have a number of strengths, e.g., when well designed they are self-revealing to a novice user, their interaction is often limited to indirect manipulation techniques. This type of interaction forces users to divert their focus from the objects they are working with to commands embedded in menus and dialog boxes. In the context of a graphical editor, this separation between object and action is inefficient and slow. Contextual menus and floating palettes are improvements, but are still indirect manipulation techniques.

In contrast, direct manipulation techniques (Shneiderman, 1998) follow three principles: continuous representation of the objects and actions of interest with meaningful visual metaphors, physical actions or presses of labeled buttons, instead of complex syntax, and rapid incremental reversible operations whose effect on the object of interest is visible immediately.

Toolglassesä (Bier et al., 1993), for example, are floating, semi-transparent, “two-handed” direct manipulation tools that are positioned with the non-dominant hand and applied with the dominant hand. A toolglass similar to a color palette would allow a user to apply a color to or absorb a color from an underlying object directly. The non-dominant hand moves the desired color over the object of interest, and then applies the color by clicking through the toolglass on the underlying object with the dominant hand. This allows the user to specify both the object and the action with a single mouse click, in context. An extension of this idea is a **Magic Lensä**, a transparent tool that shows a modified version of underlying objects. For example, a Magic Lensä version of a color palette would show the portion of the underlying objects within each cell in that cell's color. Each cell in the palette would effectively become a graphical preview.

Another advanced interface technique is **layers** (Fekete, 1996), which creates graphical sets of objects whose visibility and depth can be controlled like overlapping layers of transparencies. Layers are an effective way of separating structural objects from informational objects. This allows the user to manage the complexity of the view by adjusting the visibility of a layer based on its relevance to the current activity. For example, comments could be placed into one layer and simulation feedback in another, allowing the user to fade or hide items when they are not the focus of attention.

Standard architectures for graphical interfaces do not support these “Post-WIMP” interaction techniques and they are not trivial to add to existing interface toolkits. We have, therefore, completely redesigned the user interface architecture to support these new types of interaction.

2.3 Petri Nets In Use

Petri Nets In Use is our term for the collection of factors that influence the way a designer works with Coloured Petri Nets. We divide these factors into three groups: visual representation, work style, and interaction patterns.

The graphical characteristics of the **visual representation** are essential in communicating the underlying meaning of the CP-net to others. We look at this communication role both within smaller workgroups, in larger communities of designers in a certain domain area, and in general between designers familiar with the language of CP-nets.

CP-net designers have developed a number of guidelines for building readable nets. Some of these guidelines are applicable to graphs in general, some are specific to CP-nets, and some are used only within certain workgroups. Jensen's (1992) readability guidelines, for example, are general and applicable to most CP-nets. They include suggestions on the use of structure and graphical attributes to emphasize the flow of data and the semantic relationships between objects. Information visualization techniques (Noik, 1994) and guidelines from graph drawing research (Di Battista, 1999) are additional sources for techniques designers may employ to communicate and emphasize the meaning of their net. Understanding these guidelines and how designers may employ them is essential in the process of designing an editor to support them.

Interaction patterns describe the designer’s low level interactions with the design tool during the process of building a CP-net. For example, does the designer use keyboard shortcuts instead of menus? Does the designer create new objects or cut, paste and edit existing objects? These patterns are a product of the design tool. Understanding these patterns in the current tool helps us determine if they should be supported in the future tool, improved or replaced with different interaction patterns.

The **work styles** describes the designer’s overall process of creating a net. Several of the modeling guidelines described in Jensen (1992) suggest a top-down framework: the designer first models the core functionality and then progressively adds details, switching between editing (building the iterative process of “seeing” and “moving”, “reflection-in-action”. Feedback from our user group and examples from industrial use (Christensen et al, 1997) show how these approaches are employed in CP-net design. During this process, the designer may create a net in isolation, or use parts of existing nets, design guidelines, and group conventions. Jensen, (1992) discusses additional guidelines for modeling CP-nets involving the use of abstraction and structure. How individual designers employ these guidelines is their work style.

The components of Petri Nets In Use are intertwined: the guidelines that a workgroup use for the visual aspects of a net will affect the way that members of that workgroup create a net; the factors that influence the work process will also affect the result. Understanding these relationships helps us design a tool to support them.

2.4 Redesigning Petri Nets In Use

Figure 2 illustrates how the components of the design framework combines in the process of creating the new tool. Through the different design process activities, the users have worked on redesigning their Petri Nets In Use, and we have used this input to guide our design of the new tool.

Title:
framework3.eps
Creator:
fig2dev Version 3.2 Patchlevel 0-beta3
Preview:
This EPS picture was not saved
with a preview included in it.
Comment:
This EPS picture will print to a
PostScript printer, but not to
other types of printers.

Figure 2. Petri Nets In Use and the Design Framework

The **functionality specifications** of the design tool are created through a requirements analysis and define what the new tool must support both in terms of the visual representation, and the simulation/analysis of the underlying model. Our analysis of the current tool, research in graphical interfaces, and the participation of CPN experts define the functionality of the future tool.

In the use domain, the **interaction patterns** and **use scenarios** describe how the users of the current tool create and work with Coloured Petri Nets. We rely on observation of users and interviews of experts to identify and understand these components. Through analysis of these observations and interviews, and feedback from users, we identify areas where work styles can be improved and integrated in the new tool.

The **interaction techniques** we have introduced and the **interaction patterns** from the current tool together help define the interaction patterns of the new tool. This area of the design is entirely reinvented through the design activities described in this paper.

3 DESIGN PROCESS

The project uses a participatory design process, in which we rely on the active participation of expert CP-net designers throughout all phases of the project. Our design process involves four key activities: studying Petri nets as they are actually used in real-world settings, generating ideas for new ways of interacting with the CPN/2000 tool, designing the tool itself, and evaluating these design ideas (Figure 1).

Studying Use	**Generating Ideas**	**Designing**	**Evaluating designs**
observation	text brainstorming	design workshops	experiments
interviews	video brainstorming	software prototypes	prototype evaluation
analysis	scenario brainstorming	design scenarios	design walkthroughs

Figure 1. Design activities

3.1 Studying use

We began by studying how novice and expert users build and edit CP-nets. We conducted two types of user studies: video observations and interviews. For the video observations, we visited a range of users, including expert users at a small company working on their current project, an academic expert working through the solutions to student exercises as well as his own project, and a group of students trying out the same set of exercises. In each case, we visited people in their offices (or in the computer lab) and asked them to work as they would normally rather than give us a demo. We interviewed each person afterwards and asked them to explain their actions and identify key problems. We also interviewed several local CP-net experts about their ways of using the tool and what they felt were problems with using the tool.

We performed several types of data analysis. The most basic involved coding activities and problems from the videotapes and making transcripts of the interviews. From these, we created a video summary of the most common problems, contrasting the experiences of novice and expert users. We also created "use scenarios", comprised of typical activities and problems observed in the field studies. These scenarios enabled us to create concrete representations of "Petri nets in use", including current work styles and interaction patterns, and helped us focus on areas where they could be improved. We were particularly struck by the time spent adjusting layout and how users were forced to constantly shift their attention as they searched for commands and other information. Subsequent design activities were specifically organized to address the problems we identified during the field studies.

3.2 Generating ideas

Identifying problems is not the same as finding solutions. To kick off our design activities, we engaged in several different types of brainstorming sessions, with the goal of generating, rather than evaluating, new ideas. We began with traditional brainstorming sessions, with both CPN experts and other researchers. Everyone was asked to generate a list of problems with the new tool and ideas for possible solutions. We provided inspiration the week before by showing video clips of new interaction techniques that might be relevant to the new CPN tool.

Subsequent brainstorming sessions involved video, which forced participants to demonstrate, rather than just talk about, new ideas. Participants were given paper, scissors, transparencies, pens, post-it notes and other supplies and asked to simulate ideas for new ways of interacting with the new tool. We found that having participants perform the interaction in front of the camera forced them to work out the details of the interaction and created a video record that was used later to create software prototypes. Early video brainstorming sessions were open-ended, with participants offering design solutions to a range of problems within a specified area, such as layout. Later video brainstorming sessions were organized around use scenarios, in which participants were asked to generate multiple ways of supporting the work activities shown in the video.

3.3 Designing prototypes

Our design process involves a combination of activities, including a systematic review of the technical functionality required in the tool, a comparison of the interaction techniques possible for each function, and an analysis of how the choice of particular interaction techniques affects users' patterns of interaction with the tool. Whereas the brainstorming sessions helped expand the range of possibilities being considered, the design sessions involved the difficult task of balancing tradeoffs and actually making design decisions. We used a set of design principles to help us generate new solutions to design problems, ensure uniformity within the interface and resolve design conflicts. Most design sessions generated new issues and some design decisions were deferred, requiring either additional user studies or new brainstorming sessions.

Our design activities included a series of design workshops, with a mix of software designers, human-computer interaction researchers and coloured Petri net experts. Each workshop examined a different design issue. We also developed video design scenarios to help us examine how CP-net developers would use the tool under different work conditions. For example, designing a new net is different from editing an existing net or modifying it to make it conform to local guidelines. We had to ensure that the new design could accommodate developers working in each context.

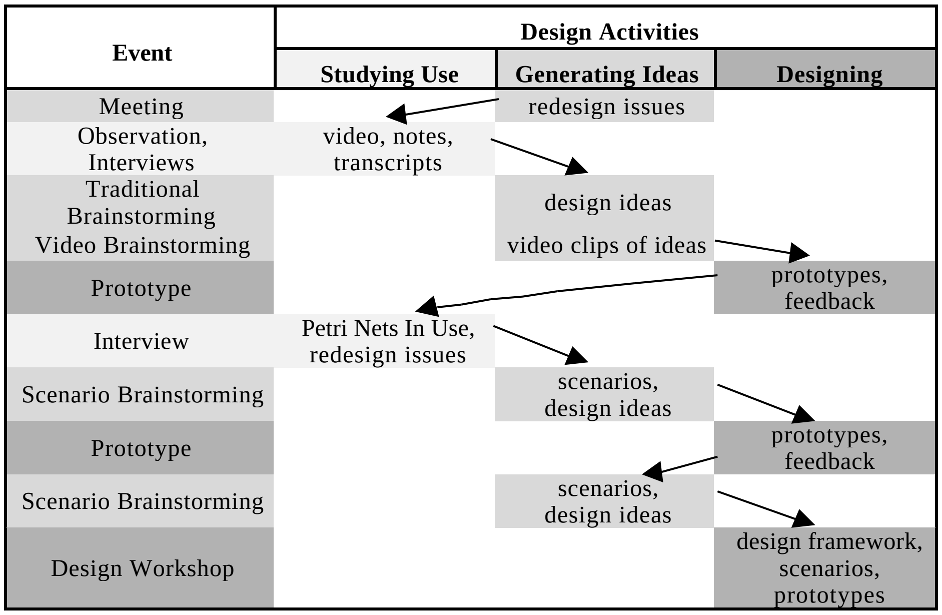
We created software prototypes, derived from the video brainstorming sessions and other design activities, to explore the consequences of particular design decisions. We also developed a completed demonstration of the current state of the design, in the form of a video prototype scenario. We used this video to communicate the current look and feel of the design to each other, new members of the team and to the programmers developing the code.

3.4 Evaluating designs

Our design process involves evaluation of ideas throughout, since we obtain feedback on a regular basis from experienced CP-net users. In addition, we have begun a series of more formal evaluation activities, including experiments to test specific design decisions and design walkthroughs to give us qualitative feedback about the look and feel of the tool. These activities are currently underway but are beyond the scope of this paper.

4 DESIGN PROBLEMS AND SOLUTIONS

We explored a variety of issues as part of our design process. Here, we focus on two specific problems and how we arrived at solutions as a result of our design process. The two problems



Event	Design Activities		
	Studying Use	Generating Ideas	Designing
Meeting		redesign issues	
Observation, Interviews	video, notes, transcripts		
Traditional Brainstorming Video Brainstorming		design ideas video clips of ideas	
Prototype			prototypes, feedback
Interview	Petri Nets In Use, redesign issues		
Scenario Brainstorming		scenarios, design ideas	
Prototype			prototypes, feedback
Scenario Brainstorming		scenarios, design ideas	
Design Workshop			design framework, scenarios, prototypes

Figure 2. The flow of events and their relationship to design artifacts

Before the project's official start, preliminary meetings between the designers of the current tool and the project manager established some extensions in functionality and interaction for the new editor. Among these issues were improved support for setting graphical attributes and positioning objects. The new editor supports these issues with toolglasses and magnetic guidelines. The following sections will trace the evolution of these ideas through the design process, and how participatory design and our focus on Petri Nets In Use has helped us to refine and improve these ideas.

4.1 Problem 1: Managing graphical attributes

The preliminary meeting raised two issues concerning graphical attributes: support for styles, and a better way of copying attributes from one object to another. These issues are important components of Petri Nets In Use; sharing a style among objects often emphasizes a semantic connection between the two, or adherence to specific guidelines. Both of these issues highlight a need for better support for managing and reusing object attributes.

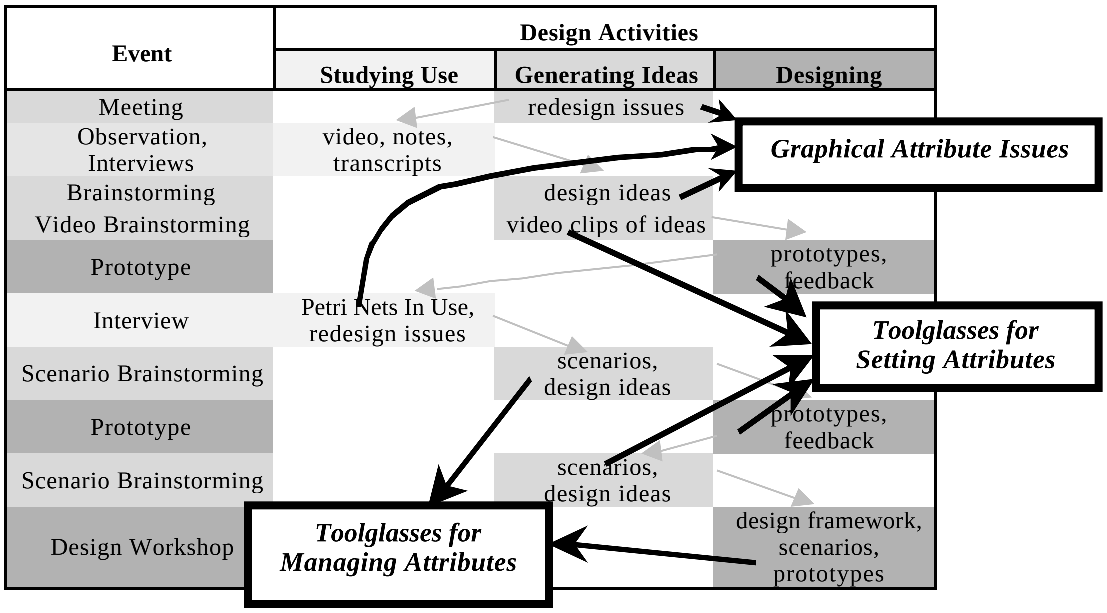


Figure 3. Sources for graphical attribute issues and toolglass ideas

Figure 3 gives an overview of the evolution of toolglasses in our design process as a way to solve this problem. Toolglasses appeared to be an obvious choice for working with attributes. As explained earlier, toolglasses bring the action to the object; a feature that is not supported with traditional menus and dialog boxes. The toolglasses we have worked with focus on different ways of setting and managing graphical attributes.

4.2 Solution 1: Toolglasses

In the first brainstorming session, people raised several issues on styles and setting attributes: style sheets, personal configuration of styles, selection by graphical property (e.g., all red transitions). At this point, the ideas were general; they described ways the users wanted to be able to work but did not contain details on interaction. Style sheets, the first idea, make it easier to formalize standard guidelines used for related groups of objects or types of objects. The second idea, personal configuration, supports switching between the guidelines developed inside different work groups, e.g. one group may use thick and thin lines, while another group might use the colors blue and green. The third idea, selection by property, overlaps with the style idea, but is more informal. For example, a user can select all red transitions and change them to thick lines directly, without changing a style.

Figure 4 shows an example of a styleglass produced in the **video brainstorming** workshop. The idea involved applying styles from an implicitly-defined style sheet; the video clip was used directly as a model for the first prototype. The clip shows how a toolglass displays objects in the net in different styles: One style for the client side of the network, one for the server side, etc. The toolglass is moved with the non-dominant hand, using a trackball, and the dominant hand can use the mouse to click on the toolglass frame with the desired style to apply that style to the object. Participants at the workshop developed several variations on this idea: One-handed toolglasses, toolglasses that can be resized, and toolglasses for port assignment.

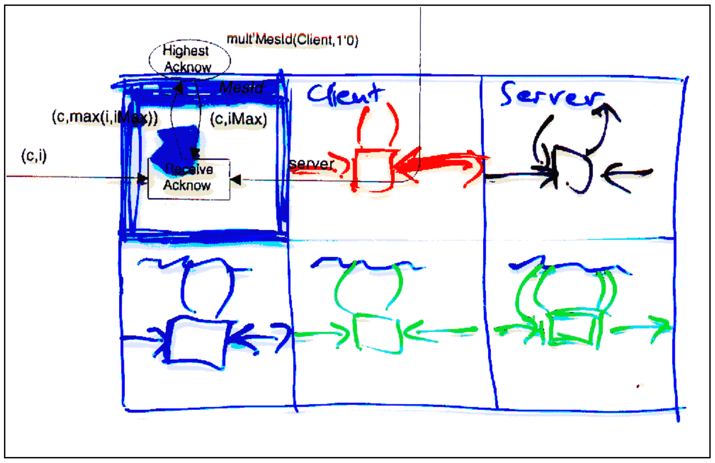


Figure 4. Style preview toolglass

The styleglass has one transparent frame where the object of interest in the underlying net can be seen unmodified. Surrounding this transparent frame are eight opaque frames for displaying the object in different styles. A variation of this is a toolglass where all frames are semi-transparent, but each displays the underlying net in a different style. Both toolglasses apply the style to the object of interest by clicking on the style. There is a trade-off between the two concerning the visual complexity of the screen image: With the opaque frames one object is shown in several styles making comparison easy, but they cover parts of the underlying net. With the transparent frames, it is more difficult to compare the different styles for one object, but more of the net structure is visible. These are some of the tradeoffs we investigated in the first prototype.

1.1.1 First use scenario workshop

Given the input from the brainstorming sessions, we wanted to look at a more continuous example of working with graphical attributes. We included this task because the idea from the first brainstorming on selection by property seemed to highlight important issues, and we wanted to see the users' ideas on how the task could be performed in the new tool. We constructed two scenarios illustrating realistic use situations, and recorded them on video tape. These two clips showed simple editing of graphical attributes on a single page, selecting all items with some particular property and changing them to a new style.

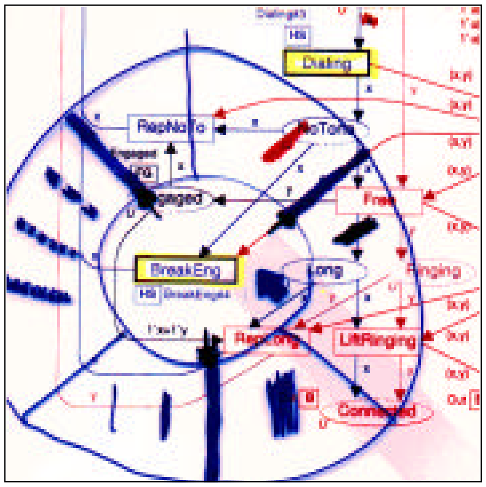


Figure 5. Graphical search and replace toolglass

The workshop resulted in a number of different versions of a search and replace toolglass: A toolglass that makes it easy to find and change items with a certain property or combination of properties *(see figure 5)*. The user specifies the values of attributes to search for and the attributes to replace them with. It is possible, for instance, to set the toolglass to find all objects that are red and make their outlines dashed. After this workshop, one CPN expert commented that if graphical search and replace was very easy, then formal support for styles may not be necessary.

In the second use scenario workshop, one group developed a toolglass to pick up and apply attributes. Its appearance was very similar to the search-and-replace toolglass, showing the values picked up from the objects in the fields of the toolglass. Clicking through the middle of the toolglass on an object then applied these attributes to the object. It was also possible to set the desired values of attributes directly in the toolglass.

1.1.2 Prototypes

We wanted to test how the toolglass with styles would actually look and feel so that we could compare the different versions. We built several prototypes of the toolglasses (e.g., changing the number and color of the frames, switching between one and two handed input) in Tcl/Tk and presented them to the project group for evaluation.

The general feedback was positive. People were surprised at how easy and natural two-handed input felt. The advantages of two-handed input (smooth movement and positioning of the toolglass, easy interaction with the underlying net) clearly made up for the small amount of desk space the extra input device occupies. Actually trying the toolglasses convinced the group of their potential, and they became central to the new design as a result of this prototype.

Given the simplicity of these first prototypes, the feedback was mostly on the idea level and not really an in-depth evaluation. Some of this feedback (how many frames should we have, where do you click to apply changes to an object, etc.) was used in the design phase and for the subsequent prototypes.

We based the second prototype on the evaluation of the first and the design ideas from the use scenario workshop. All prototypes in this phase were programmed in Beta and connected to the underlying user interface tool kit.

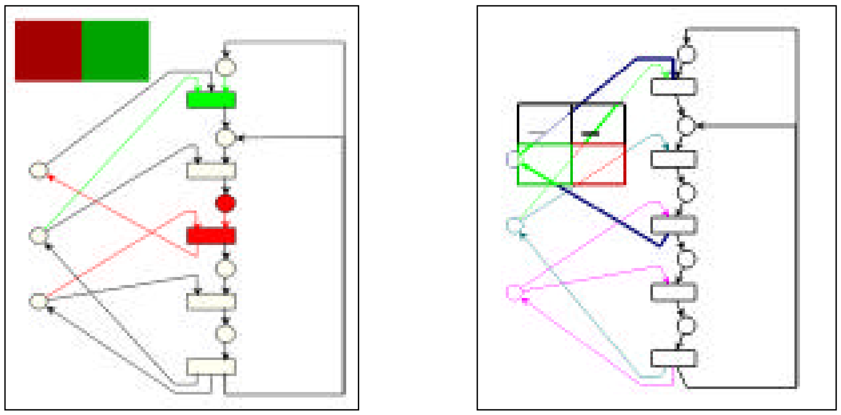


Figure 6. Toolglass prototypes. (a) Click through Toolglass. (b) Magic Lens & Toolglass

We built two simple versions of toolglasses for changing the graphical attributes of objects in the net. Figure 6a shows a toolglass prototype that a user clicks through to apply an attribute to an underlying object. Figure 6b shows a slightly modified version of the earlier prototype with transparent frames showing the net underneath in different styles, with click-through for applying the styles.

1.1.3 Designing

In the design workshops, we continued to define and evaluate the low-level interaction with toolglasses through scenarios of use. We have also begun looking into how to integrate and manage toolglasses in a workspace.

The adoption of toolglasses as a solution for setting attributes was not clear until the first working prototypes where users could actually experience them. Since then, they have become increasingly sophisticated. Originally they were a way of managing styles, and more recently they have become tools for inspecting attributes, and search and replace.

1.3 Problem 2: Positioning objects

The user studies highlighted the problem of positioning objects, and especially rerouting arcs. For example, we analyzed a video clip of an expert from industry engaged in the relatively simple task of creating a new place, redirecting arcs to that place from another, and connecting other objects in new ways. This task takes approximately three minutes, over 60% of which involve tedious and repetitive rerouting operations. Clearly much time and effort could be saved if a better way of interacting with arcs was invented. Positioning of arcs should be more highly automated, according to the positions of the objects the arcs are attached to, based on simple guidelines for how arcs should be routed through a net. Figure 7 shows an overview of the events that influenced our design of **magnetic guidelines** as a solution to the problem of positioning objects.

The first input on the issue of positioning of objects came from the preliminary meetings, and was about having constraints as a supplement to ordinary alignments. From the first brainstorming session there were only a few ideas on constraints - one was to have constraints between bendpoints on arcs, an idea that later became essential for ideas on manipulating arcs.

The issues of manipulating arcs and aligning objects might appear to be separate, but both have to do with how to position objects easily, and arcs are essential to this. When an object is moved, the arcs must follow. The users have attempted to integrate these two areas of work throughout the design process.

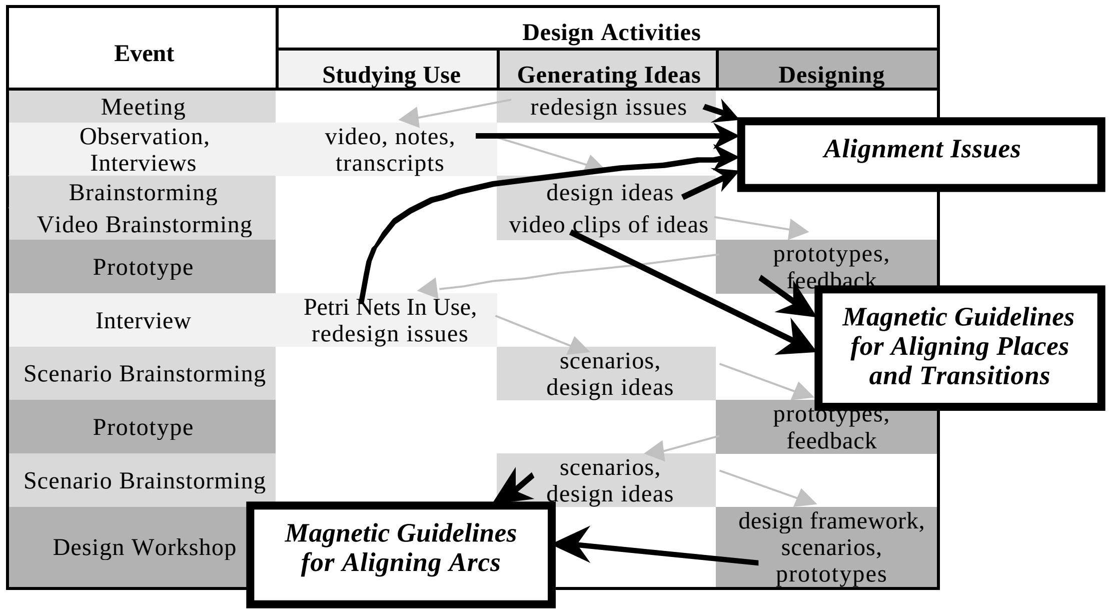


Figure 7. Sources for alignment issues and magnetic guideline ideas

1.4 Solution 2: Magnetic Guidelines

Figure 8a shows a video clip from the video brainstorming workshop illustrating the idea of magnetic guidelines. In this clip, when an object is moved over another object, guidelines will appear for both of them, and the object moved will “snap-to” the guideline of the other object, so that they are aligned. The guidelines can be more or less advanced, appearing only for the center of the object or for center and sides, so that it is possible to align to different points of the objects this way.

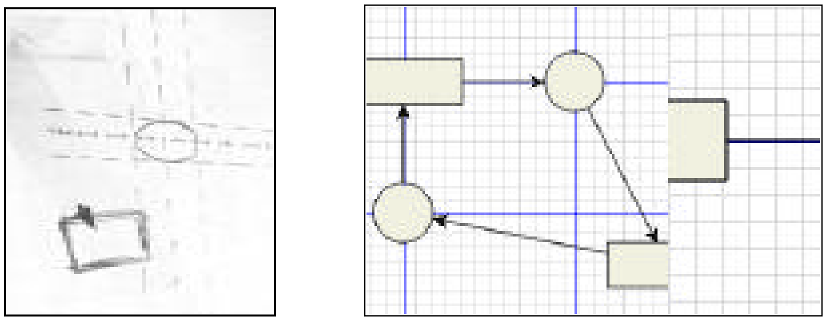


Figure 8. Magnetic Guidelines: (a) video clip of an idea from video brainstorming (dashed lines are guidelines); (b) software prototype showing three objects attached to two horizontal and two vertical guidelines. Objects dragged near a guideline snap onto it. Moving a guideline moves the objects attached to it.

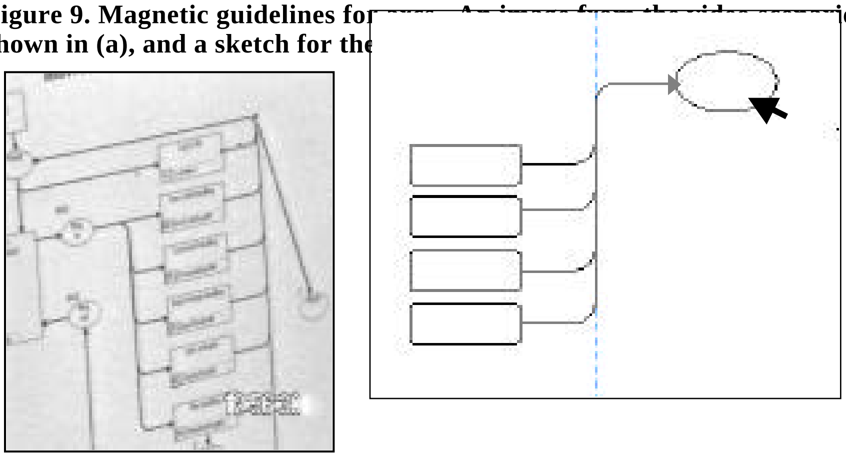
1.4.1 Prototypes

The idea about magnetic guidelines was used directly for a prototype *(see figure 8b).* In the prototype, it is possible to insert objects (places and transitions) on a canvas, connect them with arcs and attach guidelines to the different objects by clicking on the edge of the canvas. When moved over the guidelines, the objects will snap to the guidelines as demonstrated in the video brainstorming clip. The guidelines can be toggled to show or not show, but objects will still snap to when near a (possibly invisible) guideline. An important feature demonstrated in the prototype is that guidelines can be selected and moved, which also moves any attached objects. This is an example of a **reification** of the alignment constraint - the constraint between the aligned objects has been turned into an object that can itself be moved and manipulated.

1.4.2 Use scenario workshop

For the use scenario workshops we constructed use scenarios built over clips or general observations from the user study tapes. The above mentioned clip on rerouting arcs was turned into a use scenario with a storyboard, and in the workshops we worked through the scenario to redesign the interaction in a new design context. Normally, the output from the use scenario workshops would have been filmed on video, but for the magnetic guideline ideas the ideas were written down and sketched instead.

Figure 9. Magnetic guidelines fo io is shown in (a), and a sketch for th



Inspired by the prototype on magnetic guidelines and the use scenario, one of the groups in the workshop developed the idea of guidelines for arcs. These guidelines control the routing of an arc by holding the bendpoints of an arc in a fixed position relative to the objects the arc is connected to. The alignment of an arc segment is adjusted automatically when the guideline or one of the objects is moved, as shown in figure 9. This is a continuation of the reification of constraints - the constraints between the arc segments and bendpoints (originally mentioned in the first brainstorming) can now be manipulated.

1.4.3 Designing

In the designing phase we have unified the concepts of aligning and moving arcs, arc segments, nodes, etc. through the use of guidelines. We have recently begun integrating spreading constraints into guidelines as well. These guidelines evenly spread objects that are attached to them. More complex layouts can then be achieved by combining these two types of guidelines. For example, attaching several alignment guidelines to a spreading guideline creates equally spaced parallel rows of objects.

Turning all constraints into guidelines of the same overall nature makes it possible to create polymorphic interaction techniques for manipulating different kinds of constraint guidelines. These guidelines can then be combined or split according to a “guideline algebra,” making constraints a powerful and flexible functionality in the new tool.

The evolution of design ideas to handle graphical attributes and object alignment demonstrate the influence of Petri Nets In Use in our design process. Design issues were highlighted during observation, interviews and feedback from users. Design ideas were generated and progressively refined by CP-net experts, who were creating their future work styles. The participation of CP-net users has been essential throughout the design process.

5 CONCLUSIONS

Our goal in this paper is to share our current understanding of Petri Nets In Use with members of the CP-net community; both to get feedback about our observations, and input into the design of the new tool. We hope that increased understanding of Petri Nets In Use will increase the effectiveness of the new CPN tool and provide even greater support for CP-net designers.

6 ACKNOWLEDGMENTS

We would like to thank the CPN users, students and experts, who participated throughout the design process, and the members of the Beta, HCI and CPN groups for their support. We would also like to thank Susanne Bødker for her comments on an earlier draft of this paper.

7 REFERENCES

Beaudouin-Lafon, M. (1997) Interaction instrumentale : de la manipulation directe à la réalité augmentée. In Actes Neuvièmes journées francophones sur l'Interaction Homme Machine (IHM'97), Futuroscope, September.

Bier, E., Stone, M., Pier, K., Buxton, W., and DeRose, T. (1993) Toolglass and magic lenses: The see-through interface. In *Proceedings of the 20th annual conference on Computer graphics*, pp. 73-80.

Christensen, S., Jørgensen, J., Madsen, K. (1997) Design as Interaction with Computer Based Materials. In *Proc. ACM Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, DIS'97*, Amsterdam, The Netherlands, pp. 65-71.

Di Battista, G., Eades, P., Tamassia, R., & Tollis, I. (1999) Graph Drawing -- Algorithms for the Visualization of Graphs. Prentice Hall, New Jersey.

Fekete, J. (1996) Using the Multi-Layer Model for Building Interactive Graphical Applications. In *Proc. ACM Symposium on User Interface Software and Technology, UIST'96*, Seattle, USA, pp. 109-118, November.

Jensen, K. (1992) Coloured Petri Nets -- Basic Concepts, Analysis Methods and Practical Use. Volume 1, Basic Concepts. EATCS Monographs on Theoretical Computer Science, Springer-Verlag.

Noik, E. G. (1994) A Space of Presentation Emphasis Techniques for Visualizing Graphs. In *Proc. Graphics Interface '94*, pp. 225-233.

Schön, D. (1983) The Reflective Practitioner. New York: Basic Books.

Shneiderman, B. (1998) Designing the User Interface. Addison-Wesley.