Beyond Grids: Interactive Graphical Substrates to Structure Digital Layout

Nolwenn Maudet **Ghita Jalal** **Philip Tchernavskij**
Michel Beaudouin-Lafon **Wendy E. Mackay**

LRI, Univ. Paris-Sud, CNRS,
Inria, Université Paris-Saclay
F-91400 Orsay, France
{nolwenn.maudet, ghita.jalal, philip.tchernavskij, mbl, mackay}@lri.fr

ABSTRACT

Traditional graphic design tools emphasize the grid for structuring layout. Interviews with professional graphic designers revealed that they use surprisingly sophisticated structures that go beyond the grid, which we call *graphical substrates*. We present a framework to describe how designers establish graphical substrates based on properties extracted from concepts, content and context, and use them to compose layouts in both space and time. We developed two technology probes to explore how to embed graphical substrates into tools. *Contextify* lets designers tailor layouts according to each reader's intention and context; while *Linkify* lets designers create dynamic layouts based on relationships among content properties. We tested the probes with professional graphic designers, who all identified novel uses in their current projects. We incorporated their suggestions into, *StyleBlocks*, a prototype that reifies CSS declarations into interactive graphical substrates. Graphical substrates offer an untapped design space for tools that can help graphic designers generate personal layout structures.

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): User-centered design

Author Keywords

Layout; Graphic Design; Creativity Support Tools;

INTRODUCTION

One of the main tasks of professional graphic designers is to organize visual content to communicate it efficiently and appealingly to the reader. When creating magazines, books and advertisements, professional graphic designers traditionally use structures called grids: intersecting lines that partition the page to lay out content. The grid is designed to organize print content when the graphic designer knows, in advance, all of the characteristics of the final design, including content length, page size, binding, etc.

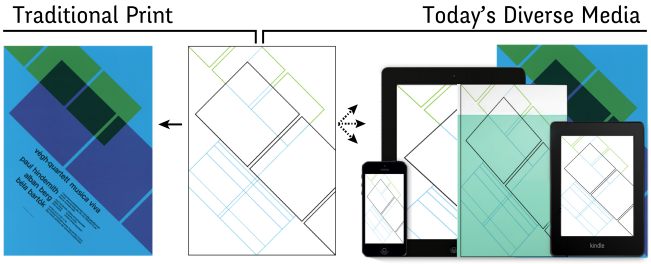

Figure 1. Josef Müller-Brockmann's grid for the Végh Quartet Poster in 1958 is based on a fixed print poster format. But is the grid enough to support today's diverse media and formats?

Traditional desktop publishing applications base their structuring tools on the grid, e.g. guides, rulers, and masters. Embedded in these tools is the assumption that output is fixed and static. However, as interactive devices proliferate, so do the demands for layouts that handle variable formats (Fig. 1). In addition, new types of media have appeared, such as websites, blogs and online magazines. Continuous information streams and media diversity add new constraints and opportunities for structuring visual content.

Graphic design pioneer Muriel Cooper foresaw the impact of this paradigm shift on graphic design practice: "*Designers will simply be unable to produce the number of individual solutions required for the vast number of variables implicit in real-time interaction. Design will of necessity become the art of designing processes.*" [6]

We are interested in investigating how designers have responded to this paradigm shift. How do they create layout structures and processes that solve the problems of the contemporary graphic design landscape? Do they go beyond the grid? Our goal is to design tools that support the constraints and possibilities of contemporary graphic design.

After reviewing related work, we describe a study of current graphic design layout practices and analyze how designers' structures and processes go beyond the grid. We introduce a descriptive framework, *graphical substrates*, to describe these structures. We designed two probes, *Linkify* and *Contextify*,

CHI 2017, May 06-11, 2017, Denver, CO, USA

 ISBN 978-1-4503-4655-9/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3025453.3025718

to explore how graphical substrates can inform new layout tools. We report the results of a second study, analyzing how designers appropriate the probes, and describe *StyleBlocks*, a prototype that reifies CSS declarations into interactive graphical substrates. We conclude with directions for future work.

RELATED WORK

Creating and structuring layout is a skilled activity, supported by a number of dedicated tools that either guide novices or assist expert graphic designers. Researchers seek to enhance our understanding of graphic designers and their practices, as well as create tools that better support them.

Supporting layout creation by novices

With the advent of desktop computing, creating layout is no longer the exclusive domain of experts, but is performed by novices as well. A common strategy for helping novices is to offer suggestions as they create layouts. For example, DesignScape [26] makes suggestions during both the brainstorming and refinement phases of layout creation, R-ADoMC [18] makes recommendations for magazine covers and Sketchplorer [34] provides real-time optimization of layout sketching. These recommendations take multiple factors into account, including color themes and visual balance.

For more focused tasks, Edge et al. [10] propose automatic alignment and systematic restyling of related objects to maintain consistency across slides. Piccoli et al. [27] propose an interactive system based on principles from physics to guide content organization on the page. These systems support novices' needs by hiding complexity from the user and focusing directly on an efficient final layout, but are less helpful for expert graphic designers who go beyond established principles to define their own rules and constraints.

An alternative strategy is to build systems that automatically design and generate layouts. Several models have been proposed, such as automated responsive design [4], semi-automated document designs inspired by magazine layouts that adapt to device screen sizes and content selections [29, 20], and adaptive grid-based document layouts [17] that automatically choose and fill in existing templates. Sukale et al. [32] also automate adaptation of layouts based on users' proximity to the screen. Yet, as Hurst et al. [15] point out, "*It is still unrealistic to expect automatic layout systems to rival the creativity of a good graphic designer.*"

Enhancing graphic designers' practices

Instead of replacing graphic designers, some systems provide tools that support specific needs, from solving localized "micro-problems", to facilitating workflow and providing programming support. At the local level, automatic tools handle many small, but crucial design challenges. For example, Moulder and Marriott [23] offer a machine learning approach to solve line-breaking issues. Expert designers can also benefit from interaction techniques that support specific and recurrent tasks, especially alignment. The GACA [36] group-aware alignment technique helps professionals deal with complex alignments. NEAT [12] and Grids-and-Guides [11] provide sets of multi-touch gestures for creating guides, as well as aligning and distributing elements on interactive surfaces.

Other tools address the designer's need to control their overall workflow. Adobe Comp [1] lets designers quickly draw initial layout ideas before moving to expert software. Gem-ni [37] and Parallel Paths [33] support parallel editing and active comparison of multiple divergent visual elements. Adaptive Ideas [21] helps designers create website layouts by sampling example elements. DÉCOR [30] supports web design workflow by providing recommendations for adapting layouts to accommodate a variety of screen sizes; and Adobe Edge Reflow [2] uses media query breakpoints to help designers envision their layouts on various devices.

Both local and workflow tools can significantly enhance the work practices of expert graphic designers, but only for well-defined tasks. However, graphic design, like many other creative activities, is undergoing a paradigm shift toward more programmatic approaches. For example, Processing [5] is a language and interactive development environment (IDE) designed to make programming more accessible for visual creators. Personal Information Management (PIM) [22] reduces the scripting learning curve for designers. Gliimpse [9] uses animation to visualize mappings between source markup and final result. These approaches empower graphic designers, but require them to think in programming rather than visual terms, which is particularly challenging for designers with no software development training.

Understanding graphic designers' practices

Design is a deceptively complex activity. Schön [28] highlights the reflective nature of design, which is difficult for automated systems to imitate, and the corresponding need to better understand real-world graphic design practice.

A few studies focus specifically on graphic designers, and provide important insights about their artifacts and processes. Murray [24] sheds light on social aspects of design practice, such as the importance of shared feedback among team members. Newman and Landay [25] focus on practical aspects of the web design process and analyze the role of several intermediate artifacts used by web designers, such as sitemaps and mock-ups. Herring et al. [14] demonstrate the importance of using examples both as inspiration and as starting blocks in creative design. These studies highlight the social and material aspects of graphic design, whereas we are more interested in practices surrounding the earliest phase of laying out content.

Danis et al. [8] show that designers begin by broadly exploring multiple alternatives. Cross [7] points out the importance of correctly framing the problem in the early design phase in order to define a set of "*first principles*". For multimedia designers in particular, Bailey et al. [1] state that, "*the early design process begins with the exploration of content structure*". These studies demonstrate the critical role that structuring plays in the early phases of design, but offer few grounded examples of how designers actually accomplish this.

Our goal is to improve how graphic designers create layouts for digital media. This requires a deeper exploration and

[1] https://www.adobe.com/products/comp.html

[2] https://www.adobe.com/products/edge-reflow.html

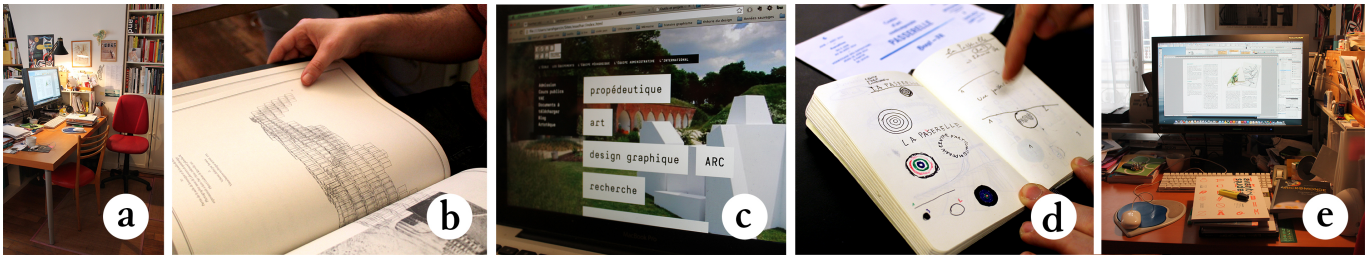

Figure 2. We interviewed 12 graphic designers in their studios (a). They demonstrated how they created layouts for both print (b) and digital media (c). They also showed us the physical (d) and digital (e) artifacts they used to create these layouts.

characterization of a crucial part of the layout design process: the creation and use of graphic design structures that guide the designer's subsequent work.

STUDY ONE: LAYOUT STRUCTURING PRACTICES

We are interested in the strategies, tools, and techniques used by professional graphic designers to create and structure layouts for both print and digital media.

Participants

We interviewed 12 graphic designers (5 male, 7 female), age 24-50, with 4-25 years of experience (mean=10,5) who work in various environments (freelance, studio, agency) and create layout for digital media (2), print media (2) or both (8).

Procedure

We interviewed participants in their studio or office for about two hours (Fig. 2a). We asked them to show us recent projects where they had to create a layout (Fig. 2b-c) and the different artifacts used to develop it (Fig. 2d-e). We asked them to tell us the story of how they made layout decisions for each project and how they obtained the final results. We probed for situations when they felt that creating the desired structure was straightforward, but also when it was challenging.

Data collection

At each interview, we recorded audio and video of the participants' interactions with the documents they created, and we photographed each artifact and any related layout creation or manipulation tools.

Analysis

We analyzed the stories and depicted them as StoryPortraits [19]: each includes a photograph of the artifact, as well as quotes and drawings that describe key steps in the process of designing a particular layout (Fig. 3). We later showed the StoryPortraits to the participants to verify the details. Next, we used a grounded theory approach [31] to categorize the keywords extracted from the stories, and organized them into a descriptive framework.

RESULTS

We collected 52 specific layout creation stories from twelve participants (3-5 stories per participant).[3] We found that seven participants use grids to structure their layouts. For example $P1_d$ defined her website structure using guides to create the grid. Some guides establish the margins that she takes into account, while others act as markers to guide content composition and alignment. When the first page is complete, she duplicates the file to reuse the guides with other pages.

We also found that many designers go beyond grids to structure their layouts, establishing rules that describe how print or digital content should be laid out. $P5_b$ described a typical example: She decided to use only multiples of 42 to create the layout of the novel *the Hitchhiker's Guide to the Galaxy*. Her layout clearly extended beyond a basic grid structure, since it required her to incorporate higher-level rules to manipulate these numbers and map them to the parameters that control the book's layout, including the CMYK color values, font sizes, line widths and grid dimensions.

Graphical substrates

We found that all participants begin by establishing what they call "*systems*", "*principles*", "*architectures*", "*structures*", "*rules*" or "*constraints*", or what we call *graphical substrates*. They share a common characteristic: they guide the layout, but rarely appear in the final result. By analogy with the substrates on which some living organisms grow, graphical substrates are the underlying structures onto which the designer "grows" a layout. As with living organisms, changing the substrate usually affects the layout as well.

The term *substrate* has also been used in another creative context to describe how music composers represent their musical ideas [13]. Although a five-line musical score provides a standard structure comparable to a grid, many composers invent their own, innovative musical representations: "*Although musical notation was important for all four composers, each composer designed his own personal musical substrate*" [13].

We developed a simple descriptive framework that identifies the types of inputs and outputs used by participants to create and interact with graphical substrates (Fig. 4). Participants based their graphical substrates – or substrates for short – on three main types of inputs: *concepts*, *content* properties and *context* constraints, such as page dimensions. They then map these inputs to *spatial* and *temporal* output properties.

Inputs based on concepts

Almost all participants (11/12) created substrates that used *concepts* as input, like $P5_b$'s use of the number 42. Some inputs are specific, such as numbers, others are more abstract,

[3] All 52 stories are available as supplementary material. In the text, the notation $P1_d$ refers to the fourth story of participant 1.

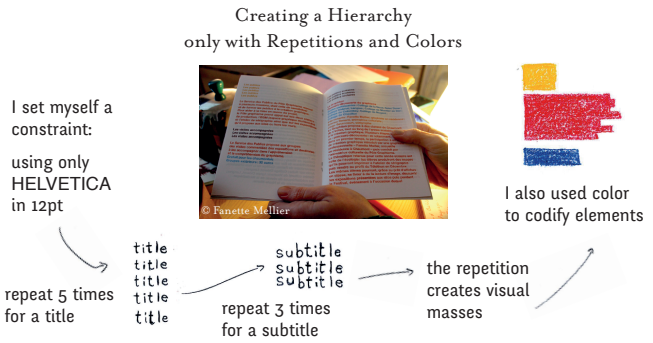

Figure 3. Each StoryPortrait includes a photograph of the artifact, quotes and drawings of each step in the process.

such as "*ambiance*". For example, $P4_b$ created typographic landscapes by preserving only one letter, "c", from a text. She erased all the other letters with a drawing application and preserved the positioning of the "c"s, creating an abstract landscape of letters for each cover.

Inputs based on content

Ten participants created substrates based on *content* properties, e.g. title, subtitle, images, or on relationships among content elements. $P7_a$ explained that "*information of the same nature must have the same style*". $P4_c$ wanted to see if it was possible "*to lay out content without any typographic hierarchy.*" for her book design (Fig. 3). She assigned different numbers of repetitions and colors to the different semantic types of content. For example, a title would be repeated five times, but a subtitle only three times. Similarly, $P7_b$ created a substrate to visually distinguish the multiple semantic elements of a grammar book. In order to communicate its subtleties, she established a substrate at the letter level: "*Every case needs to have its own style.*"

Five participants mentioned projects that used semantic relationships among content elements to establish their substrates. $P11_a$ wanted to lay out a history of text editing tools and based her substrate on parent-child relationships. She began with the two main tools and then defined a rule to dictate the layout: Place the "*children*" below and the "*parents*" above.

Inputs based on context

Ten participants used properties that they extracted from the *context*, including page and screen dimensions and properties generated by the printing process. They treated these contextual constraints as a source of creativity: $P8_c$ programmed a grid system based on relative proportions that made it easy to adapt to very different screen sizes, allowing different reading contexts. $P1_d$ used a similar approach for a website: she created a grid based on the smallest physical screen dimension (900px) to accommodate all possible readers' screen dimensions, and used it to influence all of her subsequent grid choices.

Participants also used production constraints to create substrates. For example, $P2_e$ created a book using sheets folded in two, which were nested and stapled. She used physical properties of the binding process to establish her substrate.

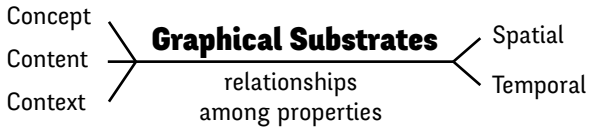

Figure 4. Participants establish graphical substrates based on properties extracted from *concepts*, *content* and *context*; mapping them to spatial and temporal properties.

She began by creating images that spanned full sheets of paper. Once folded, the left part of the image became separated from the right part, and was juxtaposed with the right part of another image, "*creating an interesting confrontation*". $P2_d$ had another project that required folding a poster. She used the fold marks as a layout constraint to ensure that text would not be printed on the folds.

Once designers select the properties they want to use as inputs to their substrates, they map them to output properties. We identified two main types: *spatial* and *temporal*.

Mapping to spatial properties

Layout is most often viewed as the organization of the *spatial* properties of the content. Designers may focus on composition, e.g. by playing with the relative positions of elements on the page or on visual weight, e.g. by playing with relative proportions of content over white space (Fig. 5).

Eight participants used the positions of elements as output to their substrates. For example, $P1_b$ created an initial substrate for the four master pages of a website where she defined the positions of the elements that would appear: "*All master pages will work the same way. They should have the same look. It's a global positioning*". In this case, she started by drawing and positioning elements on paper before moving to Adobe Photoshop. $P9_d$ defined the precise location of a recurring caption that appears on all pages of his book: "*Then we can move the images around without losing the reader*".

Nine participants used the visual weight and sizes of graphical elements from the content as output to their substrates. For example, $P7_e$ first played with the relative weights of different semantic elements: "*The content creates visual masses, I use them when defining the principle of my book design*". $P8_c$ created a relative column system, then adapted it for all his website layouts. He calls it the "*grosso-modo grid*" because it uses approximate proportions ("*tiny, little, big and huge*") that are extracted dynamically from the reader's screen size.

Mapping to temporal properties

Layout is affected by *temporal* as well as spatial properties. Designers must often create a coherent series of layouts, such as the pages of a book or a series of posters.

Ten participants created substrates that explicitly address either temporal evolution or rhythm across a series or collection. For example, $P7_d$ created a grid-system based on two or three columns for a cookbook. Depending on page type, she applied one of two grids: "*It creates a rhythm thanks to the modular repetition of this system*". She used this temporal rhythm to guide the reader through the different content types. Similarly, $P6_e$ described a temporal pattern he created for a series of

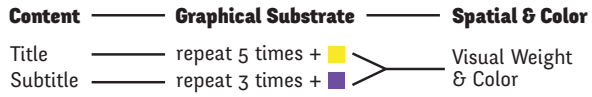

Figure 5. Representation of Fig. 3's layout structure using the descriptive framework. The substrate is based on content types (titles and subtitles) and shapes spatial properties (visual masses) as well as colors.

posters published every six months. Graphical components vary differently from one poster to the next. For example, the factual information and textured line at the center of the poster never change: "*It is a backbone*". By contrast, the client's logo partially evolves with each new poster, and the dividers between the content elements are always different.

Participants mainly used substrates to shape spatial and temporal properties of their layout. We also identified other common types of outputs, including color (Fig. 5) and font. This suggests that substrates can potentially shape a wide variety of layout properties. Fig. 6 maps the different stories map to the categories of the Graphical Substrates Framework.

Manipulating graphical substrates

Designers not only use graphical substrates as tools to shape particular layouts, but also to manipulate them as dynamic objects in their own right. We identified two main manipulation patterns: reusing and adapting.

Reusing

Most participants (11/12) reused existing substrates across projects, most often by modifying them (16 stories) or combining them with existing substrates (8 stories). For example, $P6_d$ created an evolving planet logo for a posters series. Each time, he manually reused the previous version and slightly modified one of its characteristics, such as changing the color or adding a ring. "*I first need to establish my principles over several specimens before I can override them.*". Similarly, $P9_c$ reused a substrate he created for the print identity of a company to apply it to the corresponding website. He kept some parameters, such as typography, but modified other rules to add interactivity to the website. Participants also combined multiple substrates or parts of existing substrates together. For example, $P12_c$ created a substrate for developing a coherent yet diverse set of characters for a short clip. Each feature, such as hairstyle or clothes, is based on a substrate meant to be mixed easily with the others. Creating a new character involved a simple recombination of elements from each feature's substrate.

Adapting

Participants created substrates that accommodate different levels of flexibility to cope with different levels of constraints within projects. While some rules and constraints may never be broken, such as the page dimensions of a book, a great part of the graphic designer's work is "*to find a solution for each case*" ($P7_b$). Eight participants, in 12 projects, created very flexible substrates to adapt to diverse and new constraints. For example, $P1_b$ created a "*master page*" on paper to structure the positioning of the elements of several web pages. She explained that "*Everything is flexible, even though I plan as much as I can*". Similarly, $P10_b$ created an initial structure for a book layout where all the images had the same vertical size. When he tried it with images of extreme sizes, it created too much white space on the page. So he broke his substrate for these extreme cases and adapted it with new rules to accommodate the smallest and largest images, such as spreading the content onto a second column.

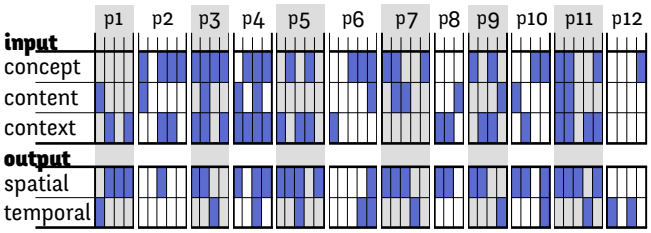

Figure 6. All five categories of the Graphical Substrates Framework appear in multiple participants' stories. Participants' stories listed in alphabetical order.

Eight participants also created 15 "hackable" substrates. In each case, the substrate guides the layout but can also be tweaked or overridden if necessary. For example, $P5_c$ created a substrate that represents the visual blocks of a book. In a few cases, some of the dialog had to overlap vertically. This led her to manually override her general rule, in order to maintain the overall grid. $P10_d$ established a precisely defined substrate for a magazine cover with variants and invariants. With each new issue, he changed the color and illustration, but retained the same grid. However, for the final issue, he decided to break the grid with an overlapping illustration.

REIFYING GRAPHICAL SUBSTRATES

All 12 participants developed and could easily describe details of the graphical substrates they created for each project. However, most of these substrates were strictly mental constructs, ideas in the designer's heads. Only participants who program could fully manipulate their substrates in existing tools. We are interested in helping designers turn their conceptual substrates into actual software tools that they can manipulate, using a process called reification [2]. All 12 participants created reifiable substrates, with clearly identified, well-defined "*rules*" or "*constraints*" to manage layout. If reified, these rules could be executed by the system or by another graphic designer. For example, $P5_b$'s book design based on multiples of 42 required her to manually set all the parameters of the book, including CMYK color values, font sizes, line widths and grid dimensions. If she had a tool that let her treat these parameters as variables, instead of being hard-coded, she could easily change the number to 54 and change the whole layout accordingly.

Even so, not all substrates are reifiable, at least not easily. Half the participants reported stories (8) where they created part or an entire ad-hoc substrate using principles such as "*ambiance*" and "*style*". These substrates could not be executed by a system or by another graphic designer unless they were defined more formally. For example, $P7_c$ inserted a set of pages into her cookbook as interludes between the recipe pages. She said, "*For these pages, nothing is aligned, it is organised using spread ambiance.*", making it difficult to systematize.

Current tools offer limited support

Participants relied heavily on a limited set of traditional tools to express their substrates: guides, master pages, paragraph and character styles. However, these tools only support a fraction of the substrates that they actually used for layout.

A first consequence of this lack of support is that designers must manage their substrates manually. For example, $P12_d$ created an animation principle for a crane appearing in a short video and decided to reuse it for all of her objects. However, she had to adapt it and apply it to each object manually, because she could not express the animation in the tool directly.

Another important consequence is that designers cannot easily share substrates with each other. We found only two cases where designers reified their substrates using traditional tools and shared the result with a colleague. $P5_a$ created a report layout in Microsoft Excel, because she knew that a non-graphic designer would be limited to Excel when creating the layout for the next issue. She based her substrate on the possibilities offered by an Excel master sheet and set as many parameters as she could to help her colleagues reuse the same layout. $P7_b$ created the substrates for a grammar book so that another designer could apply its content when creating the final layout. $P7_b$ first explored different layout principles with a one page example and later abstracted her substrate by creating a document with all the possible cases. She explained that "*The person doing the layout must be confident about which rules to apply to each content type*".

Using code to reify graphical substrates

Half the participants created projects fully or partially implemented in code (17 projects), six of which resulted in printed artifacts. On further investigation, we noticed that designers explicitly reified their substrates in code. We identified three recurring approaches that are not supported by traditional design tools: supporting more diverse inputs, automatic application of substrates, and collaboration with the reader.

Generating more diverse inputs

Reifying substrates in code lets designers manipulate additional input properties as well as create new substrates that rely on complex relationships. For example, $P8_b$ created a website layout for visualizing other websites. "*I had to design without having the content, and for all the web variability*". He created a responsive grid based on different screen sizes, to make his layout support this diversity. Whereas traditional graphic design software would fix the format choice, $P8_b$ could use this input to better tailor the layout to each reader.

Participants who wrote code created substrates that changed according to content properties. For example, $P11_d$ produced hundreds of different posters during a one-night event. With her team, she created an installation with scanners that live-streamed images into a pre-established dynamic grid. The grid reacted to the image width so that wide images spread over two squares. The team also used a mixing console that allowed a designer to choose images in the stream to produce unique posters.

Finally, participants created relationships among the layout's content elements and applied different substrates to the same content at different times. For example, $P3_b$ established a rule that dynamically creates header images for a blog layout based on text length and creation dates as inputs. He also added rules to display fewer and fewer elements of the blog post according to their publication date, which enabled him to display all posts on a single page. He coded these rules which were then applied automatically by the system.

Automatically applying graphical substrates

Reifying their substrates in code lets designers choose how the system applies them to content. This partnership helps designers focus on the early exploration and creation of substrates rather than the time-consuming task of manually applying them to each content element. For example, $P10_a$ used a system of styles and grids to automatically lay out book content from a database. He greatly appreciated this workflow: "*I could focus on the most interesting part: choosing pictures, making sure that every detail was correct and creating a cover page*". Similarly, $P2_a$ used Markdown to semantically tag the content of her book and then played with CSS properties to quickly explore alternative layouts. "*I didn't have to manually select all the images to see the change*".

Automatically applying substrates to content also meant that designers could generate an infinite number of unique layouts. For example, $P9_a$ created a generative website layout based on shifting and rotating arrows between content elements. He created a set of arrows and gave a few simple rules to the system. The system then randomly chose the arrows, which dictated a unique, potentially infinite reading path for each visitor. Similarly, $P3_c$ created a series of generated images by trying to find "*the shortest function that produces the greatest graphical diversity*". He focused on creating the substrate while the machine executed the code to create hundreds of different images for his series.

Involving readers and other designers in layout creation

Existing graphic design tools do not usually let users modify the final layout, except with respect to window size. However, if substrates are reified in code, designers can let readers provide inputs or manipulate the substrate to generate layouts dynamically. $P2_c$ based her layout on an active partnership with the reader. She created a book by hacking the possibilities of CSS Print. Each reader has to go to a website and provide a page size for their book before printing it. $P2_c$ designed the book layout to depend entirely on the book format, by using CSS rules such as relative positioning and width. She pointed out that "*There is not one final object but infinite possibilities.*" and added "*I will never see the final object*".

By embedding their substrates in code, participants could also create interactive layouts that directly react to the user's actions. For example, $P9_b$ created an interactive substrate for a website layout. He programmed two circles that reveal the background image according to the movements of the mouse. The reader directly interacts and modifies the layout by revealing the different parts of the screen with cursor movements.

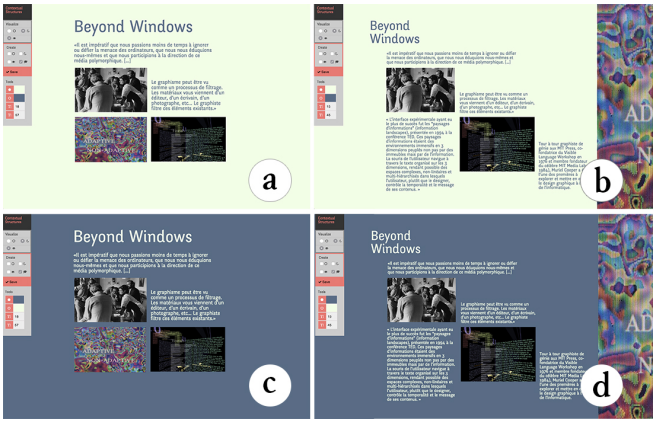

Figure 7. ***Contextify*** **accepts inputs for tailoring layout under four reading conditions: day (a-b) vs. night (c-d); preview (a-c) vs. full detail (b-d).**

Figure 8. ***Linkify*** **lets designers link content properties to create generative layouts. Here, the length of titles affects the position of images and subtitles.**

Our findings suggest that adding code provides many possibilities for reifying graphical substrates, for traditional print layout as well as interactive content. However not all designers can or want to program, and current tools only reify simple substrates such as guides. We argue that we need higher level tools that support the reification of graphical substrates.

PROBING GRAPHICAL SUBSTRATES

In study 1, we observed multiple designers explicitly reify graphical substrates into code to support (1) collaboration with the reader, (2) more diverse input, and (3) automatic application of substrates, none of which are supported by traditional tools. Inspired by these stories, we created two software probes [16] that reify graphical substrates into novel layout creation tools. The goal is not to support all possible substrates, but rather to open new opportunities for graphic designers and inspire new directions for designing tools that generate innovative layouts. *Contextify* reifies aspects of the reader's context, modifying the layout according the reader's preferences and the current reading environment (1). *Linkify* reifies spatial relationships, providing new inputs for modifying the layout according to dynamic properties of the content (2). Both are implemented as web applications to facilitate deployment and use by graphic designers.

Contextify - Richer inputs for structuring layout

Although Responsive Design lets designers adapt layouts to the width of the reader's display, designers must write code to accommodate other sources of input. We were inspired by P2's book layout, which lets readers choose the page dimension, thus taking an active role in the creation of the layout.

Contextify provides a visual interface for defining sub-layouts based on inputs from the readers' intentions and context. The system combines sub-layouts according to these inputs and generates the layout displayed to the reader. We implemented a contextual condition, *night* or *day*, and a condition specified by the reader when entering the page or web site: *preview* or *detailed reading*. *Contextify* redefines the relationship between the reader and the designer: combining these conditions results in a layout that can be tailored to four reading contexts.

We illustrate this probe with a simple scenario. Ron is designing an article on Muriel Cooper to be published in an online magazine. For the preview condition, Ron wants to show an overview of the article, so he emphasizes the title size to attract attention and includes a selection of texts and images (Fig. 7a). He saves this first sub-layout. To guide the reader through the content in the detailed reading condition, he creates a diagonal flow (Fig. 7b) and saves the second sub-layout. Ron decides to play with colors for both the night and day conditions and chooses a set of two colors (Fig. 7c-d). The reader can now access the same article from four different perspectives.

Linkify - Relationships among content parameters

A major challenge is how to design a layout without knowing the content beforehand. Instead of creating a fixed layout, some designers base the layout on dynamic relationships among content properties. We were especially inspired by P3's design for generating images based on title length. His layout evolves continuously according to the characteristics of each new content element.

Linkify lets designers visually connect content properties to establish relationships that are then reapplied when the content changes. The user selects two content elements and the system captures the visual ratios among their properties. Once the relationship is created, it is automatically applied when the content elements change, just as a spreadsheet recomputes formulas when a cell changes. We implemented five properties: the width of an element, its horizontal and vertical positions, its font size and number of characters (for text elements).

We illustrate this probe with a simple scenario. Alice is designing a blog layout and already has the first three articles. She decides to set the title position but to position the other elements according to the title length of each blog post. She first creates an interesting composition and then draws a link between the title length and the vertical positions of the subtitle and other images (Fig. 8a). Longer titles push the content downwards. Linkify automatically calculates this ratio based on the linked parameters. Alice checks that the relationships produce interesting results for longer titles (Fig. 8b).

EXPLORING THE PROBES WITH GRAPHIC DESIGNERS

Using the tools as probes helps us assess how users explore, interpret and appropriate them in the context of their current work. We are also interested in how they would use, hack and/or improve these tools in future projects.

Participants

We interviewed 12 graphic designers (6 men, 6 women), age 23-57, with 4-27 years of experience (mean=11), who work in various environments (freelance, studio, agency). 11 create layouts for print and digital media, one for digital media only. Seven had at at least some experience with a programming language and three already participated in Study 1.

Procedure

Each session lasted approximately one hour and a half. We gave a scripted presentation of the functions of each tool and asked the participant to perform a short task based on the above scenarios, after which they could experiment with each tool for 10-15 minutes. The *Linkify* task consisted of using three sample articles to create a layout for a blog that varied according to the title length. The *Contextify* task consisted of creating a layout for an article in an online design magazine that adapts to daytime vs. nighttime, and to the reader's choice of whether to read a preview or the full article. We used a think-aloud protocol and counterbalanced the order of tools across participants. After each task, we asked participants to describe to a colleague what they think the purpose of the tool was and how to use it. We asked if they had a recent layout project for which they thought the tool could have been useful and to describe in detail how they would have used it. We also asked them to suggest improvements to the tool.

Data collection

We collected audio recordings of each session and screen captures of their interactions with each tool. We also took notes based on participants' answers to our questions.

RESULTS

Participants suggested many different ways of using the tools for developing projects that they cannot currently create.

Interacting with Contextify

Nine participants described concrete examples of how they wanted to use *Contextify*. For example, P20 thought that the tool gave designers a new form of editorial power. She wanted to use the tool to design school content on tablets. Providing multiple layouts would allow students to adapt the content to their learning method: "*Some need more images, some need more words, others need to see all the content at a glance. It would also be very interesting to add some types of content only at home, such as sound for example in the case of an English workbook*". Similarly, P16 explained that for her project on Danish police data, *Contextify* "*could adapt the analysis of the data to the different jobs in the police, because they have very different needs. You could also select a global view of the information or a very precise one*".

P18 is working on an editorial web project with both detailed and summary views of the same content. He would like to use *Contextify* to simplify the creation of these views by interacting with the tool visually rather than programming everything. Finally, P19 wants to use *Contextify* as a teaching tool for students to "*understand and explore the challenges of adaptability beyond responsive design*". He wants the students to "*design according to other factors and not only the viewport*".

Interacting with Linkify

Eleven participants described concrete examples of how they wanted to use *Linkify*. For example, P24, a graphic designer working in close relationship with a developer, wanted to use the tool as an "*interface between designers and developers*". Instead of waiting for her colleague to implement the layouts in order to see how they render on each page, she could try her substrate directly, with multiple content examples, and make adjustments before handing it off to the developer.

P22 wanted to link content parameters to create the layout of an archive. She would use the "*organic nature of the tool*" to generate a layout that prompts new encounters and relationships among images because: "*It is not very interesting to have traditional linear layout for archives*". P13 was curating a webpage on which, every week, he published four animated images with a text that analyzes them. He would like to use *Linkify* "*to create a completely different layout without having to redesign everything each time, because by modifying the last page, all the others are going to evolve.*" He thought this would add a sense of "*temporal evolution*" and would offer a new perspective on his old content every week.

Extending Graphical Substrates

Participants quickly understood the power of reifying graphical substrates in tools such as *Contextify* and *Linkify*. They also gave feedback and suggested improvements, including new inputs they would like to use and new ways to turn substrates into fully interactive objects. We categorized the suggestions in two classes: generalization and reification.

Generalization - Extending inputs

Contextify supported only two sets of inputs that designers could experiment with. Ten participants suggested other inputs that would support and extend the dynamic nature of their projects. Some participants wanted to control environmental and contextual parameters in order to adapt the reading experience to the current context, such as the reader's current location (P20); current weather; ambient sound level (P19); or even a continuous time parameter to facilitate fluid changes in the layout (P13, P21). Other participants proposed letting readers specify their needs, thus creating a line of communication between the reader and the graphic designer. Suggestions included: age, handicap (P16), memory type (P20) and reading urgency (P17). Participants also wanted to create layouts according to different types of readers including author or client (P18, P23) or even job types (P16, P15).

Linkify was limited to five content properties to define relationships. Eight participants suggested additional properties such as opacity (P19) or white space between elements (P15, P17, P23). They also wanted to access non-content properties, such as viewport (P18), margins and visual reference points (P17).

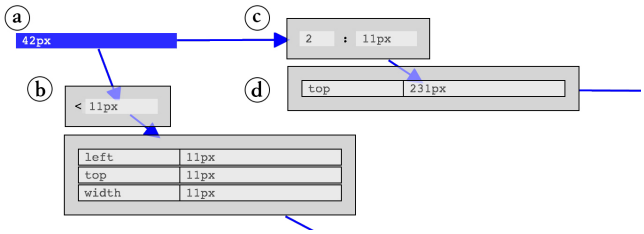

Figure 9. *StyleBlocks* supports arbitrary values (a), operators (limits (b) and ratios (c)) as well as CSS declarations (d).

Reification - Graphical Substrates as interactive objects

Overall, participants wanted more ways to interact with the substrates they were creating. First, participants wanted the substrates to become persistent, so that they could easily reuse them across projects. P17 wanted to use *Linkify* across several projects: "*To remember the common parameters I usually use in all my layouts. The system could directly reuse these parameters at the beginning of a new project*". He was currently working on a knowledge management system and wanted the system to "*directly understand some of my links and patterns and suggest new possibilities based on them*".

Second, participants wanted more control over the links provided by *Linkify*. For example, they wanted to manipulate the ratios, e.g. to invert them or specify them as absolute or relative values (P19). They also wanted to set the values (P14) and define the bounds (P16, P13) of some parameters.

Linkify and *Contextify* demonstrate that we can reify the concept of graphical substrates into tools that address previously unmet needs of professional graphic designers. These tools should support a wide variety of inputs and outputs and be flexible to let designers break rules.

STYLEBLOCKS

To further explore the power of graphical substrates for web layout, we created a prototype, *StyleBlocks*, that combines and extends *Linkify* and *Contextify*. The goal is to rethink CSS stylesheets as interactive graphical substrates. CSS (Cascading Style Sheets) is a declarative language to specify web content layout. CSS supports a large set of properties but is a very static language. Designers have to use Javascript to implement any non-trivial dynamic behavior. Preprocessors such as SASS[4] support higher-level constructs, including variables and expressions, but still generate static style sheets.

StyleBlocks (Fig. 10) reifies CSS declarations into interactive *blocks* that can be attached to content with *pipes*. In addition to CSS declarations, blocks can also represent operators, which perform functions on style declarations, e.g. limiting or scaling values. Designers create blocks from scratch or extract them directly from content by clicking on it. They connect blocks with pipes to map the output from one block to the input of another. The resulting substrates can express relationships among any numerical CSS properties (Fig. 9). Several substrates specifying sub-layouts can be applied in sequence or in parallel depending on their connections.

[4] https://sass-lang.com/

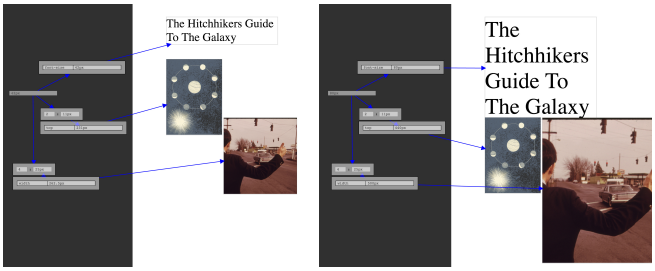

Figure 10. *StyleBlocks*: Designers create substrates that link the CSS properties of a web page. Here, a substrate generates the layout from a single value, 42 for the left page and 80 for the right one.

StyleBlocks builds on both *Linkify* and *Contextify* and incorporates feedback from Study 2. Participants wanted to create substrates from a wider variety of inputs in both probes. *StyleBlocks* supports the creation of relationships among numerical CSS properties and readers can provide their own inputs by specifying CSS values. Participants also wanted to interact with the relationships they created in *Linkify*. In *StyleBlocks*, designers can interact with relationships by adding operators such as ratios and limits.

Reifying CSS declarations is similar in spirit to Attribute Objects [35]. *StyleBlocks* extends this idea with interactive relationships among attributes. We were also inspired by visual languages such as PureData[5], which are widely used by artists to prototype and create interactive digital audio pieces.

Scenario

We illustrate *StyleBlocks* with a simple scenario inspired by P5 from Study 1. Alice is a graphic designer who wants to create a layout for the novel *the Hitchhiker's Guide to the Galaxy*. Alice's idea is to use the number 42 for many different aspects of the layout. She begins by creating a block with the value 42. She then draws relationships between that number and the horizontal position blocks (`left`) of images as well as the font size block of the title. If she decides to change the number, she can modify it and instantly see the results on the layout. She can also interact with the substrate to modify the nature and bounds of the relationships. For example, she wants two images to respond differently to the number, so she adds a ratio block between the number and the `left` block of one image and sets the value of the ratio to obtain a result she likes. Now, the number controls the left position of the two images, but in different proportions. However the two images can still go beyond the borders of the page. Alice positions the image in its desired extreme position. She clicks on it to reveal its properties and extracts the `left` block. She then feeds that value to a limit block by drawing a pipe, and adds the limit to the relationship. Now the relationship is bound by an extreme position and the image cannot go off the page.

Discussion

StyleBlocks illustrates the generative power of fully reifying graphical substrates for graphic design tools. Below, we reflect on how *StyleBlocks* addresses the needs of designers identified

[5] http://puredata.info/

in studies 1 and 2, as well as the challenges of investigating this new design space.

Extending the vocabulary of inputs and outputs
Study 1 demonstrated that graphic designers create substrates based on a wide variety of inputs and outputs that are seldom supported by existing tools. We chose to reify CSS declarations as they describe layout properties. This opens up new possibilities in terms of inputs and outputs. For example, designers can now access opacity (P19), viewport (P18) and margins (P17). Designers and even readers can also provide their own inputs, by specifying a CSS value. This supports P2's story where readers specified the page dimension to influence the final layout. However, using CSS currently limits the scope of properties to those available in the language. Designers can provide their own inputs, but only by manually specifying a CSS value. Going beyond this limitation requires opening up the system to external inputs, i.e. implementing a protocol for connecting external data to layouts.

Providing greater flexibility through Reification
Study 1 showed that graphic designers establish design principles that guide, but do not strictly define the layout. *StyleBlocks* supports flexibility by supporting the tweaking of relationships using ratios (P14) and limits (P13, P16). For example, P10 had to break his substrate to fit extreme images in the layout. *StyleBlocks* lets him limit the height of the image. Substrates are independent from content. They can be reused and combined by detaching and reattaching them to other modules. This supports P12's strategy of combining existing graphical substrates to produce a diverse set of coherent layouts. However, adding more complex conditions to *StyleBlocks* is a challenge. Constraint systems such as Apple Auto Layout support spatial constraints, but they generally do not let designers simultaneously keep and tweak a relationship.

Back and forth between layout and Graphical Substrates
Study 1 showed that designers use both top-down and bottom-up strategies for creating and manipulating substrates. *StyleBlocks* lets designers manipulate both the final layout and the substrate within the same workspace, using the same interactions, which enables a constant back-and-forth. Designers can start by building an example layout, then extract interesting properties and reify them into an independent structure that can then be reused and manipulated. This would facilitate P7's workflow, so she would not have to manually create the substrate for her grammar book before handing it off to another designer. Designers can also build a substrate from scratch and iterate quickly, since it is instantly reapplied to the whole layout. Designers currently write code to produce feedback loops. For example, P3 experimented with algorithms that produce simple yet expressive rules for his images. *StyleBlocks* would let him modify the substrate and see the resulting images immediately without the indirection of changing and re-executing code. However, whereas *StyleBlocks* lets designers change the content while keeping existing blocks, it does not automatically apply these blocks to the new content. Supporting CSS classes would address this problem, letting designers further automate substrates.

Future Work
Building *StyleBlocks* helped us better understand the potential of reifying graphical substrates, as well as the requirements for implementing them. Here we present two areas for future work based on our experience with the tool.

Diversifying Representations of Graphical Properties
StyleBlocks currently borrows CSS's representation of graphical properties as text. This means that colors, positions, and transformations can all be connected together in a consistent way. However, designers would benefit from more expressive representations of graphical properties, such as representing positions as points and lines or ratios as rectangles. This would let designers apply existing graphical techniques and workflow to work with substrates.

Exploring other types of Graphical Substrates
We plan to explore which abstractions offer an optimal balance between power and simplicity within the design space of graphical substrates. We chose to model graphical relationships as networks of blocks and pipes. This approach is closer to programing than visual design. In the future, we plan to continue working with graphic designers to develop and evaluate a more extensive vocabulary of graphical substrates.

CONCLUSION
This paper explores how professional graphic designers structure layouts beyond the use of grids. Based on interviews with 12 professional graphic designers, we discovered that they all use sophisticated ways to structure layout. We call these structures *graphical substrates* and show that they consist of mapping a variety of inputs, including conceptual, content and contextual inputs, onto outputs, most notably spatial and temporal properties. However, current layout tools provide very limited support for graphical substrates and graphic designers currently either manage them by hand or rely on code to explicitly represent them in their designs.

To explore how graphical substrates can be reified into tools, we created and tested two software probes: *Contextify* reifies context inputs and lets designers tailor layouts according to the reader's intention and context; *Linkify* reifies spatial relationships to let designers create dynamic properties based on content. 12 professional graphic designers experimented with these tools and explained how they would enrich their current projects. They also suggested improvements such as extending the set of possible inputs and outputs as well as making graphical substrates persistent and manipulable. We incorporated their suggestions into a new prototype, *StyleBlocks*, that reifies CSS declarations into interactive graphical substrates. We argue that graphical substrates offer a general framework for generating new forms of layout creation tools that meet the evolving needs of professional graphic designers.

ACKNOWLEDGMENTS
We thank our participants for their time and their inspiring insights into graphic design practice. This work was partially supported by European Research Council (ERC) grants n° 321135 CREATIV: Creating Co-Adaptive Human-Computer Partnerships, and n° 695464 ONE: Unified Principles of Interaction.

REFERENCES

1. Brian P. Bailey, Joseph A. Konstan, John, and John V. Carlis. 2001. Supporting Multimedia Designers: Towards More Effective Design Tools. In *In Proc. Multimedia Modeling: Modeling Mutlimedia Information and Systems (MMM2001)*. 267–286.
2. Michel Beaudouin-Lafon and Wendy E. Mackay. 2000. Reification, Polymorphism and Reuse: Three Principles for Designing Visual Interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '00)*. ACM, New York, NY, USA, 102–109. DOI: http://dx.doi.org/10.1145/345513.345267
3. Marianela Ciolfi Felice, Nolwenn Maudet, Wendy E. Mackay, and Michel Beaudouin-Lafon. 2016. Beyond Snapping: Persistent, Tweakable Alignment and Distribution with StickyLines. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 133–144. DOI: http://dx.doi.org/10.1145/2984511.2984577
4. Grace Colby. 1992. Maintaining legibility, structure, and style of information layout in dynamic display environments. In *Posters and short talks of the 1992 SIGCHI conference on Human factors in computing systems*. ACM, 73–74. http://dl.acm.org/citation.cfm?id=1125089
5. Andres Colubri and Ben Fry. 2012. Introducing Processing 2.0. In *ACM SIGGRAPH 2012 Talks*. ACM, ACM Press, 12. DOI: http://dx.doi.org/10.1145/2343045.2343061
6. Muriel Cooper. 1989. "Computer and Design". *Design Quarterly n°142* (1989), 23.
7. Nigel Cross. 2002. Creative cognition in design: processes of exceptional designers. In *Proceedings of the 4th conference on Creativity & cognition*. ACM, ACM Press, 14–19. DOI: http://dx.doi.org/10.1145/581710.581714
8. Catalina Danis and Stephen Boies. 2000. Using a technique from graphic designers to develop innovative system designs. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM, ACM Press, 20–26. DOI: http://dx.doi.org/10.1145/347642.347657
9. Pierre Dragicevic, Stéphane Huot, and Fanny Chevalier. 2011. Gliimpse: Animating from markup code to rendered documents and vice versa. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 257–262. http://dl.acm.org/citation.cfm?id=2047229
10. Darren Edge, Sumit Gulwani, Natasa Milic-Frayling, Mohammad Raza, Reza Adhitya Saputra, Chao Wang, and Koji Yatani. 2015. Mixed-Initiative Approaches to Global Editing in Slideware. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM Press, 3503–3512. DOI:http://dx.doi.org/10.1145/2702123.2702551
11. Mathias Frisch, Sebastian Kleinau, Ricardo Langner, and Raimund Dachselt. 2011a. Grids & guides: multi-touch layout and alignment tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1615–1618. http://dl.acm.org/citation.cfm?id=1979177
12. Mathias Frisch, Ricardo Langner, and Raimund Dachselt. 2011b. Neat: A Set of Flexible Tools and Gestures for Layout Tasks on Interactive Displays. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*. ACM, New York, NY, USA, 1–10. DOI: http://dx.doi.org/10.1145/2076354.2076356
13. Jérémie Garcia, Theophanis Tsandilas, Carlos Agon, and Wendy Mackay. 2012. Interactive Paper Substrates to Support Musical Creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 1825–1828. DOI: http://dx.doi.org/10.1145/2207676.2208316
14. Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting inspired!: understanding how and why examples are used in creative design practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 87–96. http://dl.acm.org/citation.cfm?id=1518717
15. Nathan Hurst, Wilmot Li, and Kim Marriott. 2009. Review of automatic document formatting. In *Proceedings of the 9th ACM symposium on Document engineering*. ACM, 99–108. http://dl.acm.org/citation.cfm?id=1600217
16. Hilary Hutchinson, Wendy Mackay, Bo Westerlund, Benjamin B. Bederson, Allison Druin, Catherine Plaisant, Michel Beaudouin-Lafon, Stéphane Conversy, Helen Evans, Heiko Hansen, Nicolas Roussel, and Björn Eiderbäck. 2003. Technology Probes: Inspiring Design for and with Families. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 17–24. DOI: http://dx.doi.org/10.1145/642611.642616
17. Charles Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin. 2003. Adaptive grid-based document layout. In *ACM SIGGRAPH 2003 Papers (SIGGRAPH '03)*. ACM Press, 838. DOI: http://dx.doi.org/10.1145/1201775.882353
18. Ali Jahanian, Jerry Liu, Qian Lin, Daniel Tretter, Eamonn O'Brien-Strain, Seungyon Claire Lee, Nic Lyons, and Jan Allebach. 2013. Recommendation system for automatic design of magazine covers. In *Proceedings of the 2013 international conference on Intelligent user interfaces*. ACM, 95–106. http://dl.acm.org/citation.cfm?id=2449411

19. Ghita Jalal, Nolwenn Maudet, and Wendy E. Mackay. 2015. Color Portraits: From Color Picking to Interacting with Color. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 4207–4216. DOI: http://dx.doi.org/10.1145/2702123.2702173

20. Mikko Kuhna, Ida-Maria Kivelä, and Pirkko Oittinen. 2012. Semi-automated magazine layout using content-based image features. In *Proceedings of the 20th ACM International Conference on Multimedia (MM '12)*. ACM Press, 379. DOI: http://dx.doi.org/10.1145/2393347.2393403

21. Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2257–2266. http://dl.acm.org/citation.cfm?id=1753667

22. Maryam M. Maleki, Robert F. Woodbury, and Carman Neustaedter. 2014. Liveness, Localization and Lookahead: Interaction elements for parametric design. In *Proceedings of the 2014 conference on Designing interactive systems*. ACM, 805–814. http://dl.acm.org/citation.cfm?id=2598554

23. Peter Moulder and Kim Marriott. 2012. Learning how to trade off aesthetic criteria in layout. In *Proceedings of the 2012 ACM Symposium on Document Engineering (DocEng '12)*. ACM Press, 33. DOI: http://dx.doi.org/10.1145/2361354.2361361

24. Dianne Murray. 1993. An ethnographic study of graphic designers. In *Proceedings of the Third European Conference on Computer-Supported Cooperative Work 13–17 September 1993, Milan, Italy ECSCW'93*. Springer, Kluwer Academic Publishers, 295–309. http://dl.acm.org/citation.cfm?id=1241934.1241954

25. Mark W. Newman and James A. Landay. 2000. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In *Proceedings of the 3rd Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques (DIS '00)*. ACM, New York, NY, USA, 263–274. DOI: http://dx.doi.org/10.1145/347642.347758

26. Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM Press, 1221–1224. DOI: http://dx.doi.org/10.1145/2702123.2702149

27. Ricardo Farias Bidart Piccoli, Rodrigo Chamun, Nicole Carrion Cogo, João Batista Souza de Oliveira, and Isabel Harb Manssour. 2011. A novel physics-based interaction model for free document layout. In *Proceedings of the 11th ACM symposium on Document engineering*. ACM, 153–162. http://dl.acm.org/citation.cfm?id=2034725

28. Donald A Schön. 1983. *The reflective practitioner: how professionals think in action*. Basic Books, [New York].

29. Evan Schrier, Mira Dontcheva, Charles Jacobs, Geraldine Wade, and David Salesin. 2008. Adaptive layout for dynamically aggregated documents. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI '08)*. ACM Press, 99. DOI: http://dx.doi.org/10.1145/1378773.1378787

30. Nishant Sinha and Rezwana Karim. 2015. Responsive Designs in a Snap. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015)*. ACM, New York, NY, USA, 544–554. DOI:http://dx.doi.org/10.1145/2786805.2786808

31. Anselm Strauss and Juliet M. Corbin. 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications. 312 pages. https://books.google.fr/books?id=wTwYUnHYsmMC

32. Ryan Sukale, Olesia Koval, and Stephen Voida. 2014. The Proxemic Web: Designing for Proxemic Interactions with Responsive Web Design. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp '14 Adjunct)*. ACM, New York, NY, USA, 171–174. DOI: http://dx.doi.org/10.1145/2638728.2638768

33. Michael Terry, Elizabeth D. Mynatt, Kumiyo Nakakoji, and Yasuhiro Yamamoto. 2004. Variation in element and action: supporting simultaneous development of alternative solutions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 711–718. http://dl.acm.org/citation.cfm?id=985782

34. Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and Explore with a Layout Optimiser. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. ACM, New York, NY, USA, 543–555. DOI: http://dx.doi.org/10.1145/2901790.2901817

35. Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-Oriented Drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4610–4621. DOI: http://dx.doi.org/10.1145/2858036.2858075

36. Pengfei Xu, Hongbo Fu, Chiew-Lan Tai, and Takeo Igarashi. 2015. GACA: Group-Aware Command-based Arrangement of Graphic Elements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM Press, 2787–2795. DOI:http://dx.doi.org/10.1145/2702123.2702198

37. Loutfouz Zaman, Wolfgang Stuerzlinger, Christian Neugebauer, Rob Woodbury, Maher Elkhaldi, Naghmi Shireen, and Michael Terry. 2015. *GEM-NI*: A System for Creating and Managing Alternatives In Generative Design. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM Press, 1201–1210. DOI: http://dx.doi.org/10.1145/2702123.2702398