HAL
open science

A Review of Milestones in the History of GUI Prototyping Tools

Thiago Rocha Silva, Jean-Luc Hak, Marco Winckler

To cite this version:

Thiago Rocha Silva, Jean-Luc Hak, Marco Winckler. A Review of Milestones in the History of GUI Prototyping Tools. 15th IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT 2015), Sep 2015, Bamberg, Germany. pp. 1-12. hal-01343040

HAL Id: hal-01343040

https://hal.science/hal-01343040v1

Submitted on 7 Jul 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Open Archive TOULOUSE Archive Ouverte (OATAO)
OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author-deposited version published in : http://oatao.univ-toulouse.fr/
Eprints ID : 15379

The contribution was presented at :
http://www.interactcongress.eu/events/447/Berlin-2015.html

To cite this version : Rocha Silva, Thiago and Hak, Jean-Luc and Winckler, Marco Antonio *A Review of Milestones in the History of GUI Prototyping Tools.* (2015) In: Workshop on User Experience and User-Centered Development Processes (IFIP WG 13.2 - 2015) in conjunction with the 15th IFIP TC13 INTERACT conference, 14 September 2015 (Bamberg, Germany).

Any correspondence concerning this service should be sent to the repository administrator: staff-oatao@listes-diff.inp-toulouse.fr

A Review of Milestones in the History of GUI Prototyping Tools

Thiago R. Silva, Jean-Luc Hak, Marco Winckler
ICS-IRIT, Université Paul Sabatier, Toulouse, France
{rocha,jean-luc.hak,winckler}@irit.fr

Abstract. Prototyping is one of the core activities of User-Centered Design (UCD) processes and an integral component of Human-Computer Interaction (HCI) research. Nonetheless, for many years, prototyping was synonym of paper-based mock-ups and only more recently we can say that dedicated tools for supporting prototyping activities really reach the market. In this paper we propose to analyze the evolution of prototyping tools proposed by the academic community to support research activities and prototyping tools that are aimed and improve the development process of commercial user interfaces. Thus, this paper presents a review of past and current graphical user interface prototyping tools, in order to set up the state of the art in this field, observing fundamental milestones of features over time. For that, we have screened publications presented since 1988 in some of the main HCI conferences and 113 commercial tools available on the web. The results enable a brief comparison of characteristics present in both academic and commercial tools, how they have evolved and what are the gaps that can provide insights for future research and development.

1. INTRODUCTION

Prototyping graphical user interfaces (GUIs) is considered one of the most important activities in a User-Centered Design process (UCD) as a mean to investigate alternative design solutions in the early and advanced phases of development. For this purpose, a lot of tools have been developed to support a set of activities, such as planning, sketching, designing and evaluating prototypes of user interfaces.

Prototypes, particularly in the early stage of designing GUIs, are an important way to communicate and discuss ideas and requirements [12]. Low-fidelity techniques are often used by designers to sketch and present new ideas and concepts about the interface that will be built. This kind of activity involves the users early in the design process, promoting an effective participatory design and improving the user experience. With the advance of software development lifecycle, medium and high-fidelity prototypes can be used to refine some features or provide more accurate information about interaction options.

In the academic context, since 1988, the most important conferences in the field of Human-Computer Interaction (HCI) have given space for tools developed in order to solve several scientific challenges related with this theme. However, dedicated tools for supporting prototyping activities only started to

have an impact in the market by 2003. Thus, we can observe a temporal gap between the research interest and market adoption.

The aim of this work is to investigate the state of the art in GUI prototyping tools, analyzing the main contributions in terms of new ideas and features, regarding the main milestones over time, as well as identifying new research gaps in this area. To accomplish that, we described a review of tools that have been presented to the academic community or provided in the commercial context. For academic tools, we systematically investigated the main conferences on HCI since 1982, with the first SIGCHI Conference. For commercial tools, we investigated the most common ones, constantly mentioned by designers and present in most professional activities related to prototyping. The next section of this paper presents the research protocol used to investigate tools in both academic and commercial context.

2. METHODOLOGICAL APPROACH

In this section we present the methods used for the analysis of the academic and commercial tools.

1.1. For Academic Tools

We sought top ranked HCI conferences and selected those that were sponsored or co-sponsored by ACM, IEEE and/or IFIP. Only proceedings published in English and available in digital form were considered. Domain-oriented conferences such as mobile, embedded, robot, pervasive and ubiquitous interfaces, were excluded from the analysis. We considered the following conferences and period of publication: ACM CHI (1982-2014), ACM UIST (1988-2014), ACM DIS (1995-2014), ACM EICS (2009-2014), IFIP INTERACT (1984-2013).

We screened papers that propose prototyping tools and/or describe the use of existing tools for supporting prototyping. We considered relevant those papers that included in the title and/or abstract any of the following keywords: prototype, prototyping tool, prototyping interface, wireframe, wire-framing, sketch, sketching, draws and drawing. With these keywords, 7.243 publications were selected. Subsequently, we excluded papers reporting tools developed for specific prototyping in specific environments (ex. sketches of buildings for architects, drawings for designers, circuits and physical devices for engineers and so on). We also did not consider papers that report model-based prototyping of multimodal user interfaces because our main interest is in tools that can support the concrete development of user interfaces, not only to model it. Finally, we mainly considered full papers. However, one tool that was not published as a full

paper, but describes important and concrete results for this field, was added to the final list of papers.

2.2. For Commercial Tools

There are a large number of commercial tools to prototype GUIs. 113 prototyping tools have been examined in two steps. The first step was to check the main features of each tool and roughly categorize them. Doing so, we checked the website of each tool, sorting them according to their similarities with other tools (e.g. the way they handle the creation of the user interface, the precision that can be achieved when describing the behavior of the prototype) and removing from the list each tool that does not match with the study. Then, we have defined some criteria that tools must comply with before doing further analysis. The criteria were: "Is the tool a standalone software or an extension/library/framework?", "Is prototyping generic interfaces possible?", "Is there a free trial of the tool?", "Is the tool still updated and documented?" and "Does the prototype produced with the tool support any interaction?" A few exceptions were made if the prototyping tool was featuring interesting functionalities, but eventually did not accomplish one or more criteria. In a second step we focused on the mechanisms used by prototyping tools for the creation of a prototype and especially the construction of the presentation and the dialogue.

With those criteria, we discarded 23 tools that did not draw attention during the first analysis. Some of these tools were not exactly a software tool, but just a library or theme to apply on existing software, so we focused on this existing software and its features instead. We also removed from the list tools that cannot produce a reusable file. The advantage in using software to produce a prototype is that they can easily be modified, reused and shared. Some tools also were not still updated and documented, so they were eliminated; as well as tools that were too specific and too restrictive, so prototyping generic interfaces was not possible.

3. SELECTED TOOLS

We have selected 17 papers really talking about academic prototyping tools which included the following tools: SILK [5], DENIM [6], DEMAIS [1] and CogTool [4] (from ACM CHI), Gambit [11] (at ACM DIS), GRIP-it (at ACM EICS), Mirage [7], Ensemble, Lapidary [8], Druid and Monet (at ACM UIST), SIRIUS, MoDE, SCENARIOO, Freeform and SketchiXML [2] (at IFIP INTERACT), and ActiveStory Enhanced [3] (from XP International

Conference). We have retained 90 commercial tools for detailed analysis. We have identified three categories depending on what can be prototyped with the tool: the behavior, the presentation (visual aspect) or both.

The first one gathers 9 tools that are more suited for representing the behavior of a prototype. In the second one, we have regrouped 8 drawing tools like Inkscape or Photoshop, where it is possible to create a visual prototype without caring about the behavior or the interactions possible. Finally, the last category corresponds to tools that can manage both graphical and behavior aspects and it features the remaining 73 tools. Mock-up tools and wireframe tools fall into this category. Therefore, we have decided to focus on this last category since they are mainly tools that are dedicated to the construction of fully functional prototypes.

4. SET OF MILESTONES OBSERVED

A temporal view of academic and commercial tools is presented in Figure 1. We can observe three main periods of interest in this kind of tool.

The first one, before 1995, coincides with the emergence of UIMS tools. The first tools mainly treated of high-fidelity prototypes, using mostly design elements from the final interface, and being strongly dependent on the hardware. The main advantage of the UIMS tools is that, after development and testing, the interface prototype can be directly attached to the application the prototype becomes the interface [7]. Since the emergence of UIMS tools, authors start to discuss the lifecycle of development processes using prototyping tools, redesign is discouraged because it takes time and is perceived as a “waste" of money, and it is suggested that evaluation should occur in early phases in the development process. Nonetheless, UIMS tools do not give the flexibility needed in the early stages of prototyping. Typically, in the brainstorm meetings, designers are more focused on describing important aspects of the problem to be solved, more related to requirements than to interface aspects. Even if one can rapidly build a prototype to demonstrate his/her ideas using this kind of tools; they won’t be able to deliver a solution that does not consider design aspects which are unnecessary for the early phases. Tools such as PowerPoint and Visio are often reported as a mean to support the edition of the presentation of user interfaces; even though these tools should be generic.

The second period (1995-2005) we can observe in Figure 1 is represented by tools in which one can really draw a functional prototype; some of them support sketching. SILK [5] and DENIM [6] are the first tools designed for hand sketched GUIs and are still the most representative tools that use sketch recognition. Landay et al. [5, 6] believe sketches are important for

communicating ideas with other team members and gaining valuable feedback early in the design process.

This period was followed by an increasing interest in other ways to prototype interfaces and the inclusion of behavior modeling which can be roughly dated after 2005, leading up to a third and last moment, with a substantial increase of commercial tools from 2007 to now.

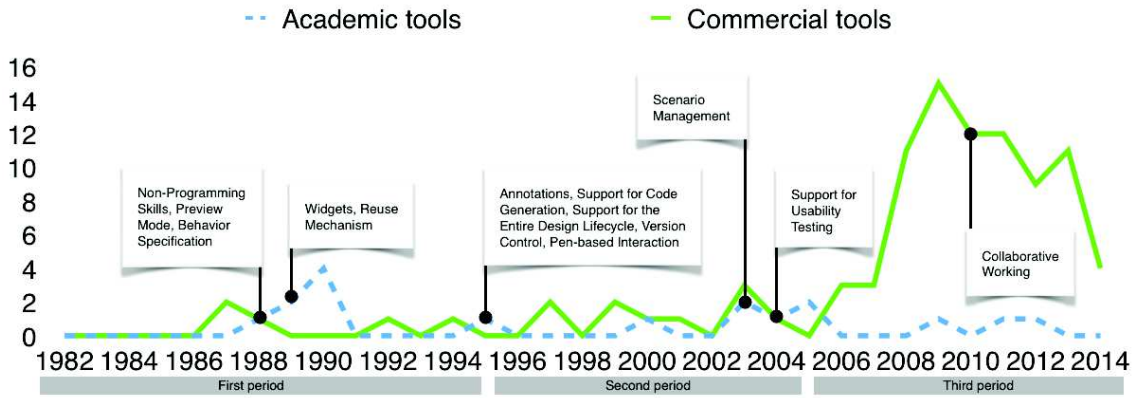

Figure 1. Number of both academic and commercial tools per year.

Hereafter, we describe some of the main features, set in the time by milestones, observed during the analysis of both academic and commercial tools.

4.1. Non-Programming Skills

UIMS tools started providing non-programming skills with resources that allow designers to build their interfaces without programming skills. The goal was to allow users to concentrate on what is to be done rather than how to do it [7]. One way to accomplish this objective is to give users the ability to directly manipulate the representations of concepts from the task domain (e.g., design objects). Furthermore, authors of UIMS tools believe that special purpose systems are more likely to provide the kind of semantic and articulatory directness necessary to allow the user to concentrate on the task rather than the tool. Examples of tools pursuing this goal include MIRAGE [7] using objects, and its successors Lapidary [8], Ensemble and Druid. Most of the tools that came after this milestone do not require programming abilities from users, even with Lapidary [8], for instance, demanding some Lisp programming ability to express more refined behavior. Nowadays, it is common sense between developers of GUI tools that they should simplify the activity of designers and interface engineers, and requiring some level of programming skills is a throwback. Because of that, all the tools analyzed work with abstract elements and behavior models as a way

to provide prototyping resources for the users, without requiring any kind of ability to program software. We believe that this is a well-established feature today.

4.2. Pen-Based Interaction

Pen-based interaction regards the possibility to interact with prototypes through a pencil and paper metaphor. This kind of interaction uses an electronic pen to really draw the interface in a touch device. Landay and Myers [5] believe that designers feels more comfortable using pencil and paper to prototype initial versions of their ideas, discuss and present them in brainstorm meetings.

The first time this feature appeared was in SILK [5], followed by DENIM [6], Freeform, SketchiXML [2], ActiveStory Enhanced [3] and Gambit [11]. Authors of these tools observed that all designers sketch with pen on paper as a regular part of their design process, even though eventually all of them end up using computerized tools. This activity allows designers to explore the space of possibilities more effectively through sketching than using computer-based tools, at least during the early parts of the process.

Given sketching characteristics, normally prototypes of this kind are throwaway. Additionally, features using sketch recognition are also provided, in a way that sketches made by hands can be recognized and interpreted by the tool. In this case, widgets (in a higher level of fidelity) are automatically generated to support further evolutions in the prototype that might not be throwaway. Such kind of technique is presented in all of the tools listed above.

However, even more than 20 years after SILK [5], there are no commercial tools that implement this kind of feature using sketch recognition. Tools like Blueprint, Cacoo, Mockup Plus, NinjaMock and Pidoco support both palette and sketching methods of interaction, but not sketch recognition. Even so, it is definitely an unpopular feature among commercial tools.

4.3. Widgets

Widgets have been used to build prototypes since Lapidary [8]. Their use guides the major part of tools that work with a palette as interaction technique nowadays. Widgets have the advantage to facilitate the process of element manipulation, offering a fast manner to set various components as menu bars, buttons, windows and form elements. Even tools that work with a sketching mechanism like SILK [5] and DENIM [6] set up a library with known elements (drawn before) and treat them as a widget for future uses. The restriction of a palette of widgets is that the prototype is limited to the components that are

available on the prototyping tool and their representation. For instance, Balsamiq provides only low-fidelity widgets.

4.4. Behavior Specification

Prototyping tools facilitate the representation or the implementation of interactions and actions. This behavior specification makes it possible to define higher-precision and evolutionary prototypes, though the tools are not equal in the possibilities they offer.

As we have only analyzed academic tools with clear purposes of prototyping, almost all of them provide some kind of behavior specification. Lapidary [8] was the first one we noticed. It provides interesting features and resources that lead to dialogue construction. The dialogue can ensure the task sequencing and so the different screens of the interface. Other behavior building mechanisms using event handling on widgets can set up more refined behavior, too. Animations are another way to define elementary behaviors through a predetermined scenario.

It is possible to distinguish tools by the way they represent the dialogue. Tools like Pidoco represent the sequencing of screens through a state machine or a diagram. Tools can use similar mechanisms to ActiveStory Enhanced [3] or Balsamiq which support only basic interactions with hyperlinks between screens. Tools like Axure or JustInMind can use conditions, variables to modify properties and states on top of the hyperlinks. Finally, some tools like Appery.io and ScreenArchitect support the use of programming code.

4.5. Collaborative Working

In our study, we have found 25 tools that feature functionalities allowing designers to collaborate on the same prototype. Appery.io and Hotgloo are among the first prototyping tools that support collaborative work. Collaboration is made possible in two different ways: in a synchronized environment and in an asynchronous environment. Tools that support synchronized modifications like HotGloo or InVision allow several users to work at the same time on the same prototype. Every modification is applied on each instance of the prototype. On top of that, functionalities that help users to know what collaborators are doing are often available like a tele-pointer or a chatroom. Tools that support asynchronous collaboration like Axure or GUI Design Studio are mainly based on revision control where each user has a local copy of the prototype and annotations.

As for academic tools, Gambit [11] is really dedicated to collaborative working and supports sketch production and visualization on different devices, session storage and retrieval, private and/or public production of sketches, over

providing a broad view of the drawings (like papers arranged on a wall) and a fine view of them.

4.6. Reuse Mechanism

Reuse is always important to reduce the workload of designers and users. In the prototyping domain, tools generally use widget libraries to provide reusability: they provide templates and pre-defined behaviors. Sketching tools like SILK [5] and SketchiXML [2], by the use of UI widget representations, support the reuse mechanism of interface elements which have already been taught to the system before. DENIM [6] and Gambit [11], in contrast, do not support reusable components.

Commercial tools like Appery.io, HotGloo, iRise, Protoshare and UXPin feature the usage of breakpoints and screen versions, thus promoting reuse of design for multiple devices. This method supports multiplatform design by resizing and repositioning elements that have already been created. It is also possible for some tools to apply a theme on the prototype like ForeUI or MockupScreens. Doing so will switch the appearance of the prototype without recreating it. Another way to reuse existing work is to import an image of a prototype designed using any drawing tool like a paper prototype. After that, the prototyping tool is used to add interactions for the imported prototype. Some commercial tools are based on this mechanism of import and interaction adding like MarvelApp, Flinto or Notism.

4.7. Scenario Management

Scenario-based design is a family of techniques in which the use of a future system is concretely described at an early point in the development process. Narrative descriptions of envisioned usage are then employed to guide the development of the system and the conduct of experiences [10].

Scenario management refers to the ability of the tools to work with different scenarios and manage them in an integrated way with prototypes and behavior descriptions. It is not an easy feature to implement because it is strongly dependent on the complete development processes and their models, so their implementation normally becomes too restrictive. Despite the fact that this feature has appeared first in Freeform, a restrictive tool working as a Visual Basic 6 plug-in, in 2003, there has not been much evolution since then. If scenarios are seen as simple annotations in the prototype, we can consider several tools providing some kind of feature to treat them, but if we see those scenarios like complete requirements specifications, which need to be managed and

controlled through the whole lifecycle, there are currently no tools able to solve this problem.

4.8. Preview Mode

Preview mode is an important feature to enable visualization of one executable version of the prototype. In this mode, we can execute and simulate all interactions specified during the construction of the prototype. Users can test the application as a rough final product. It is important, in this case, to visualize how the prototype will really appear in a real environment and to promote usability testing and collect adequate feedback from particular stakeholders. MIRAGE [7], Lapidary [8], SILK [5] and DENIM [6] feature this mode whereas a third application is needed for SketchiXML [2]. Most commercial tools include a preview mode.

4.9. Support for Usability Testing

Prototypes can be used to support usability tests by collecting data from users. Indeed, it is possible to store useful information that can be measured while the prototype is being used (time spent on each screen, the area clicked, etc.). Tools like CogTool [4], Solidify, PickFu, IntuitionHQ provide functionalities that help to create usability tests. For instance, it is possible to add instructions or questions to the test of the prototype and to create tasks that have to be accomplished. With ActiveStory Enhanced designers can export the prototype to a web-based Wizard-of-Oz testing tool, allowing test participants to remotely walk through a UI while recording metrics such as mouse movements and time spent on pages. After carrying out user tests, collected data are made available with an interface dedicated to the management of results.

4.10. Support for Code Generation

Another advantage of prototyping using a software is that it is possible to automatically generate code that can be used directly or refactored afterwards. SILK [5] generates code for an old OpenLook Operating System and Freeform for Visual Basic 6. Both of them are very restrictive in terms of environment. SketchiXML [2] and Gambit [11] produce interface specifications and generate code in UsiXML, an open-source format based on XML.

Among all commercial tools that have been studied, 21 of them can generate, for example, web pages based on the prototype created. Well-known tools like AppSketcher, Axure, ForeUI and JustInMind directly generate these pages, considering they use code for dialogue descriptions, in this way providing a

mechanism to support other phases of the design lifecycle and evolving the prototype to the final user interface.

4.11. Version Control

Version control is often related to the fact that any document or software that is created can be modified at several times during its lifecycle. It allows each user to check the current state of the document, different versions that exist and the reason of any modification. Version control is a mechanism that is interesting for prototyping since the prototype is constantly evolving due to feedback, needs and requirements that emerge throughout any project, and because a prototype can be declined in several versions depending on the options of design that are considered by the designers.

SILK [5] supports version control through design history, and started using this feature in 1995. DENIM [6], in contrast, does not support version control. Another interesting feature when designing several solutions and options of design is the ability to compare two versions. Commercial tools like Alouka, Codiqa, FluidUI, HotGloo and JustInMind support only version control. Concept.Ly is able to compare two different screens using a slider.

4.12. Annotations

The annotation system is an interesting feature since it may be a way to collect various feedback on problems that are identified by the annotators and to communicate with users [9].

Annotations are dated information and can be used to keep a trace of the evolution of a prototype since they might influence the development. That information can take several forms like inquiries, decisions, constraints and specifications, use cases of a software, problems encountered by users, advantages found in other software, test data or even ideas.

Several ways of annotating have been implemented in prototyping tools. For instance, SILK [5] and DEMAIS [1] supports textual annotations as an input design vocabulary. Alouka, Balsamiq, inPreso, Lumzy and WireFrame Sketcher support annotations through widgets, the simplest method. Axure, MockupScreens and JustInMind support this feature as a property of widgets. There are also tools that have a dedicated annotation mode like Concept.Ly, ForeUI and NinjaMock.

4.13. Support for the Entire Design Lifecycle

Support for the entire design lifecycle means that designers can work with the same prototype since the early stages of development, evolving it towards most refined levels and becoming the final user interface through an evolutionary and iterative development lifecycle. This characteristic is a frequent constraint to adopt current prototyping tools in several development processes. The time wasted building a throwaway low-fidelity prototype becomes an adverse argument to obtain stakeholders support.

SILK [5] supports the transformation process of the sketches to real widgets and graphical objects, but no more than that. SketchiXML [2] and Gambit [11], even being a sketching platform, need to be integrated with other UsiXML tools to support several levels of prototyping. Thus they need a third application to provide that. DENIM [6] and DEMAIS [1] do not support different refinement levels, so they do not cover the whole lifecycle (they do not generate finished HTML pages, for example). DENIM [6] just allows the navigation among different representations in a web-design prototype, such as site maps, storyboards and mock-ups.

Some tools like ScreenArchitect support model description by providing links between prototypes and models like state machines, leading then to a more integrated environment in the UCD development processes.

5. CONCLUSION

We can observe these milestones as a large spectrum of features being covered by prototyping tools over the time. As we have supposed, original and innovative features come from academic tools, generally providing solutions to problems that will be addressed by commercial tools some years after. We have also observed, in particular cases, that some features like pen-based interaction, presented by SILK [5] twenty years ago, seem to be not interesting to be adopted by tools used in the commercial context.

Another aspect we can highlight is the pool of commercial tools launched after 2008, when Balsamiq came up. These tools have incorporated the most aspects we report in this paper, providing, in different levels, implementations of these concepts, and many times, being strongly repetitive in their qualities. Nevertheless, it shows a continued interest from both the academic and industrial community in this theme, suggesting an open space of research in several points.

This is a research still under development and is part of an initial investigation about prototyping tools. This state of the art on existing prototyping tools will help us to have a better understanding of the remaining gaps of features that can

support the software development process through the whole lifecycle of prototyping.

Features and current directions point to an accurate analysis of the main gaps of features and open research problems, based on prototyping as support activity for the development lifecycle. Regarding these gaps, we have already identified little support of the tools for annotation activities in a requirements process. Tools that treat annotations as a property and not as a single remark put in the prototype better support the specification process of requirements. Even though, the way they capture the information coming from this annotations is not profitable to be used for supporting business rules, specification of needs or functional descriptions.

Another important gap already identified is related to integrated support for development models. Task and system models, when considered, are normally no integrated to the prototyping activity in most of the current tools. At this way, it is hard to work in an integrated environment where it is possible to build low-fidelity prototypes, evolve them to more refined ones and, from scenarios, requirements annotations and constraints, support the development of models and check the user interface according to a unified prototyping specification.

References

1. Bailey, B. P., & Konstan, J. A. (2003, April). Are informal tools better?: comparing DEMAIS, pencil and paper, and authorware for early multimedia design. In *Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 313-320). ACM.
2. Coyette, A., & Vanderdonckt, J. (2005). A sketching tool for designing anyuser, anyplatform, anywhere user interfaces. In Human-Computer Interaction-INTERACT 2005 (pp. 550-564). Springer Berlin Heidelberg.
3. Hosseini-Khayat, A., Ghanam, Y., Park, S., & Maurer, F. (2009, January). ActiveStory Enhanced: Low-Fidelity Prototyping and Wizard of Oz Usability Testing Tool. In *XP* (pp. 257-258).
4. John, B. E., Prevas, K., Salvucci, D. D., & Koedinger, K. (2004, April). Predictive human performance modeling made easy. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 455-462). ACM.
5. Landay, James A., and Brad A. Myers. "Interactive sketching for the early stages of user interface design." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press/Addison-Wesley Publishing Co., 1995.

6. Lin, J., Newman, M. W., Hong, J. I., & Landay, J. A. (2000, April). DENIM: finding a tighter fit between tools and practice for Web site design. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (pp. 510-517). ACM.
7. McDonald, James E., Paul DJ Vandenberg, and Melissa J. Smartt. "The mirage rapid interface prototyping system." Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software. ACM, 1988.
8. Myers, B. A., Zanden, B. V., & Dannenberg, R. B. (1989, November). Creating graphical interactive application objects by demonstration. In *Proceedings of the 2nd annual ACM SIGGRAPH symposium on User interface software and technology* (pp. 95-104). ACM.
9. Naghsh, A. M., Dearden, A., & Özcan, M. B. (2006). Investigating annotation in electronic paper-prototypes. In Interactive Systems. Design, Specification, and Verification (pp. 90-101). Springer Berlin Heidelberg.
10. Rosson, Mary Beth, and John Millar Carroll. "Usability engineering: scenario-based development of human-computer interaction". Morgan Kaufmann, 2002.
11. Sangiorgi, U. B., Beuvens, F., & Vanderdonckt, J. (2012, June). User interface design by collaborative sketching. In Proceedings of the Designing Interactive Systems Conference (pp. 378-387). ACM.
12. Schvaneveldt, R, W., et al. "Towards a Modular User Interface" (CRL Technical Report No. MCCS-85-10). Computing Research Laboratory, New Mexico State University, Las Cruces, New Mexico. 1985.