Software Environment for Research on Evolving User Interface Designs

Juan C. Quiroz, Anil Shankar, Sergiu M. Dascalu, Sushil J. Louis

University of Nevada, Reno, USA
{quiroz, anilk, dascalus, sushil}@cse.unr.edu

***Abstract*—We investigate the trade off between investing effort in improving the features of a research environment that increases productivity and investing such effort in actually conducting the research experiments using a less elaborated, albeit sufficiently operational environment. The study case presented is an interactive genetic algorithm environment we created to evolve user interfaces designs. We present three productivity improvements integrated in our environment and examine whether on the long run the research productivity can be in fact increased by spending development time on enhancing the research tools rather than on performing the research itself. The three improvements are the integration of the entire system interface into a main wxPython window, the addition of a runs manager for setting up multiple experiments, and the creation of a data manager for effective exploration and visualization of data produced in the experiment runs. We also discuss several guidelines for transitioning a research environment such as ours from a researcher's tool to an end-user's tool.**

***Index Terms*—Research software, research productivity, interface design, evolution.**

I. Introduction

USER interface (UI) design is a complex, time consuming, and expensive process because of its explorative and iterative nature. "Design is rarely a straightforward process and typically involves much iteration and exploration of both requirements and design solutions" [13]. Guidelines of style help user interface designers to evaluate UI designs since guidelines provide principles for the use of color, the use of font, the use of margin and spacing, and the layout of widgets [1], [2], [4], [8]. We have used an interactive genetic algorithm (IGA) to allow user interface designers to explore the space of UI designs through evolutionary techniques [11], [12]. Through evolution, the UI designer is able to rapidly explore creativity and gain insight into various designs. Our approach allows the user to incorporate both expert knowledge in the form of objective design metrics (or guidelines) and subjective human preferences into the UI design process through an interactive genetic algorithm (IGA).

Our research software environment provides a front-end to the interactive genetic algorithm. We allow the user to configure the IGA behavior through the GUI. Our current environment provides limited functionality and does not address several usability and efficiency issues. For example, we use XUL, a markup language for UIs, as the target language for our UIs because of its flexibility and ease with which widgets can be manipulated [9]. Due to the limited support of XUL rendering with wxPython, our environment implementation language, we dump the IGA output to a file every generation to be viewed by the user through a system capable of rendering XUL. To visualize these file we then use the Mozilla web browser.

In this paper, we present three modifications to the existing environment aimed at improving research productivity: (1) integration of XUL output into the main wxPython window; (2) a manager for specifying experiment runs; and (3) a manager for the analysis and visualization of data produced from the many experiment runs. We discuss how such improvements to the environment empower the user and increase research productivity by providing the user with an intuitive tool that reduces tedious tasks. We also look into the effort needed to improve the research environment and assess its worthiness versus alternatively spending this effort on actually conducting research experiments using the existing (less developed) environment. In other words, we discuss two different approaches: the first is to invest some time and resources to better prepare the research tools (and then conduct the experiments), the second is to focus immediately on conducting the experiments and advancing research (by using less elaborated, albeit operational research tools).

In addition, the long-term goal is to deploy our environment to user interface designers. However, the environment needs to be prepared for the context on which it will be used for our intended audience. We foresee two audiences, researchers and end-users. The current environment is tailored towards researchers. We discuss how we will go about moving our environment from a "researcher's tool" to an "end user's tool". Our discussion focuses on how we will conduct this transition.

We hope that the discussion presented in the paper will help other researchers customize their experiment environments and tools developed for the end-users and thus strengthen the work of both the research and end-user communities.

The paper is structured as follows. Section II presents an

The authors are with the Evolutionary Computing Systems Lab (ECSL), in the Computer Science and Engineering Department at the University of Nevada, Reno, NV 89557 USA.

overview of our environment for UI evolution and the current status of the project. Section III shows the improvements made to the existing environment to increase research productivity. In section IV we discuss the process by which we plan to transition our tool from a researcher's tool to an end-user's tool. Section V presents a discussion of related work. Finally, in Section VI we present our concluding remarks and outline directions of future work.

II. Overview of Our Research Environment

Our research software environment is a front-end to an interactive genetic algorithm (IGA).

A. Interactive Genetic Algorithms

A genetic algorithm (GA) is a search technique based on the principles of natural selection and survival of the fittest [3]. It consists of a population, where individuals are potential solutions to the problems to be solved. Solutions are probabilistically recombined and mutated, favoring the reproduction of high fitness individuals. The process continues for several generations. We can usually determine the fitness of individuals algorithmically. However, there are times when determining the fitness of individuals is difficult if not impossible. Interactive genetic algorithms (IGAs) replace the objective fitness evaluation with human subjective evaluation. By incorporating human subjective input, we can instill human expertise, emotion, and intuition into the evolutionary process [7]. UI evolution, because it is both guided by expert knowledge taken from guidelines of style and a human sense of aesthetics, is a suitable problem for the IGA domain. For a survey of IGAs the reader is invited to consult [7].

B. User interface Design Evolution

We use an IGA to evolve user interface designs [11], [12]. UI design is a complex and time consuming process. The design process is driven both by guidelines of style and human expertise. The problem with guidelines of style is that applying them beyond specific cases and interpreting the guidelines is itself a major problem [13]. The IGA empowers UI designers to explore the space of UI designs, and by doing so to instill creativity and inspiration into various possible designs.

We encode UI layouts as individuals in the IGA population. The user is presented a subset of the population, consisting of the best UI designs in the population, and then selects the UI design he/she likes the best and the UI design he/she likes the least. This user feedback guides the evolution of the UI designs from generation to generation. Furthermore, we incorporate computable objective metrics taken from guidelines of style, to guide the evolution along with the user input. The objective and subjective heuristics are combined in a linear weighted sum. The combination of objective heuristics allows the UI designer to evolve UI designs which both reflect user preferences and which adhere to coded guidelines of style.

In our previous work we found the user able to effectively bias the evolution of UI designs [11], [12]. The current coded guidelines are: (1) a high contrast between the panel background color and the widgets' color, and (2) a low contrast between widget colors. The first guideline enforces legibility by having different foreground and background colors. The second guideline enforces widgets to have a similar shade of color, instead of having each widget with an independent color. Lastly, we layout the widgets in a grid construct. The grid construct aligns widgets in rows and columns, implicitly enforcing another guideline of style.

C. The Environment

The environment allows the user to configure the IGA parameters. As shown in Figure 1, settings that can be modified include the crossover rate, mutation rate, population size, number of individuals to display, the objective and subjective weights of the fitness linear sum, and the frequency of user input. The user can further customize advanced options such as using the roulette wheel or the tournament selection for the IGA, and if tournament selection is chosen, the ability to specify the tournament size and the probability of choosing the winner of the tournament.

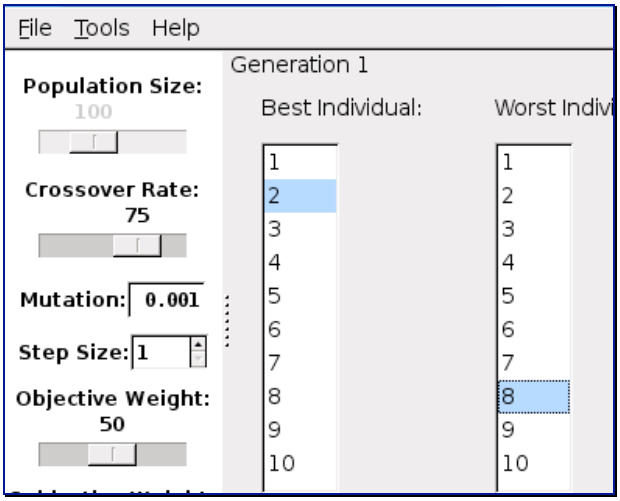

Fig. 1 User interface evolution environment.

The IGA was implemented with Python and the GUI with wxPython. Python was chosen because it enabled us to do agile programming through fast prototyping, continuous refactoring, and iterative redesigns.

XUL, the XML User Interface Language, is a cross platform markup language for user interfaces [9]. XUL supports a large variety of widgets, ranging from simple buttons, textboxes, and radio buttons, to tree controls, popup menus, and menubars. We used XUL as the target language because of its flexibility and ease with which widgets can be manipulated and styled through CSS stylesheets. Through XUL we can also treat and manipulate UI designs as tree structures.

D. User Interface Specification

A user defines a UI to be evolved by writing the list of widgets to be evolved in XUL format. XUL, as a subset of XML, is intuitive and straightforward. A button in XUL is

defined by "<button label='I am the label for this button!'/>". The XUL list is loaded into the environment through a file dialog, and evolution can begin. Currently Python has limited support for XUL renderers. Consequently, we write the evolution output to a XUL file, and view the file with the Mozilla web browser, which uses the Gecko rendering engine to render XUL through the browser [9]. We made the web browser refresh every few seconds, to keep the user from having to refresh the web browser window every generation. An example of a subset of user interface "individuals" at generation 0 is shown in Figure 2.

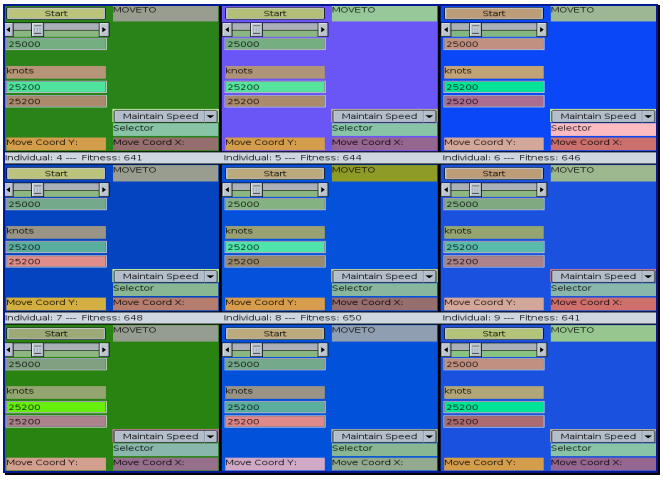

Fig. 2 User interface individuals at generation 0. These individuals were written in XUL and rendered with the Mozilla web browser.

III. Investing Time on Development Versus on Research Experimentation

We have worked on improving the effectiveness of our research environment by implementing three new features: (1) integrating the UI output into wxPython instead of writing it to a XUL file; (2) adding an experiment manager to handle numerous experiment runs and their organization; and (3) adding a data manager to navigate and visualize the large amounts of data generated by running many experiments.

A. wxPython Integration

As implemented (in a simpler way) in the previous version of our research environment, the dumping of the visualization of individuals to a XUL file presents efficiency and usability issues. First of all, both the Mozilla web browser and the Sysetm E environment must be started every time in order to be able to view the population status and to provide relevant feedback to the IGA. Once Mozilla has been started, the user must then open the XUL file to which the output was sent. Lastly, the user has to constantly switch back and forth between the UI evolution environment itself (to enter the user input of the best and worst UI displayed) and the Mozilla browser (to see what the UIs at the current generation look like).

We propose a design solution to the aforementioned problem: to integrate in the main wxPython window the rendering of individuals being evolved instead of writing it to a XUL file. Alternatively, we could implement the entire GUI with XUL. The advantage of the latter approach is the ability to make the system available online and thus have users evolve GUIs through the web. However, the challenge in this is the communication overhead between the XUL widgets and the python IGA backend, which would require extensive processing. Therefore, we have opted for the former solution.

The integration into wxPython, illustrated in Figure 3, has several advantages. First, the user does not have to keep switching back and forth between Mozilla and the main environment interface. Second, the user selection becomes intuitive and less error prone. A left double click on an individual selects it as the best UI design, and a right double click on an individual selects it as the worst UI design. Another advantage is that productivity improves through having the session runs faster. With the XUL output viewed on Mozilla, the user had to switch windows and refresh the browser to see the latest subset for user evaluation. Instead, with wxPython integration the update is almost instant, speeding up the IGA session. Overall, the evolution of individuals and their visualization becomes straightforward.

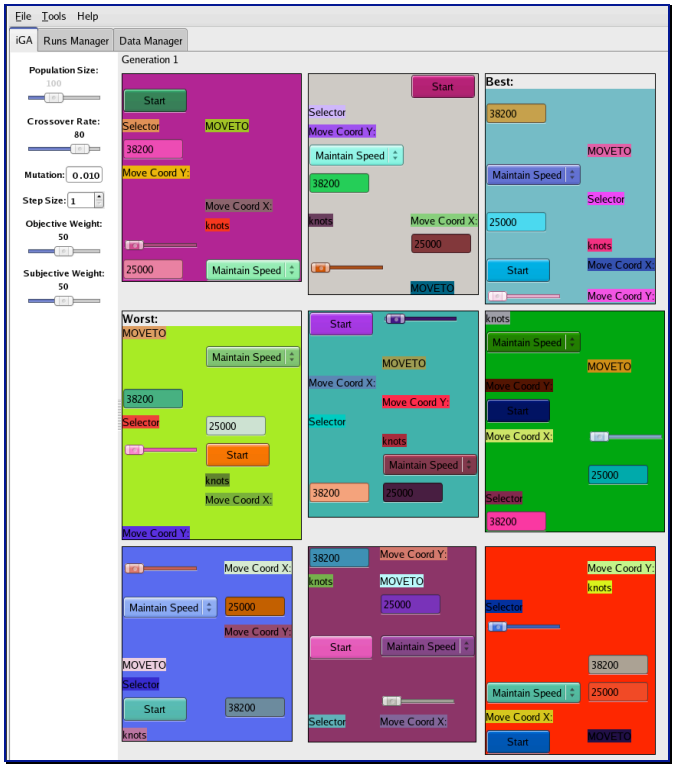

Fig. 3 Integration of displaying XUL individuals into the main wxPython window.

B. Experiment Runs Manager

The second improvement was to add a manager of experiment runs. The interface of this experiment runs manager, shown in Figure 4, allows the user to specify as many experiments as are desired and their configurations. The processing is parallelized, so that the user does not have to manually run the code in multiple machines. Configurations for an experiment include all settings that would usually be set

through the main interface, as described previously in sub-section II.C. An experiment is added by clicking on the "Add" button. Clicking the "Start" button begins running the experiments, with a progress bar showing the status of each experiment as time goes by.

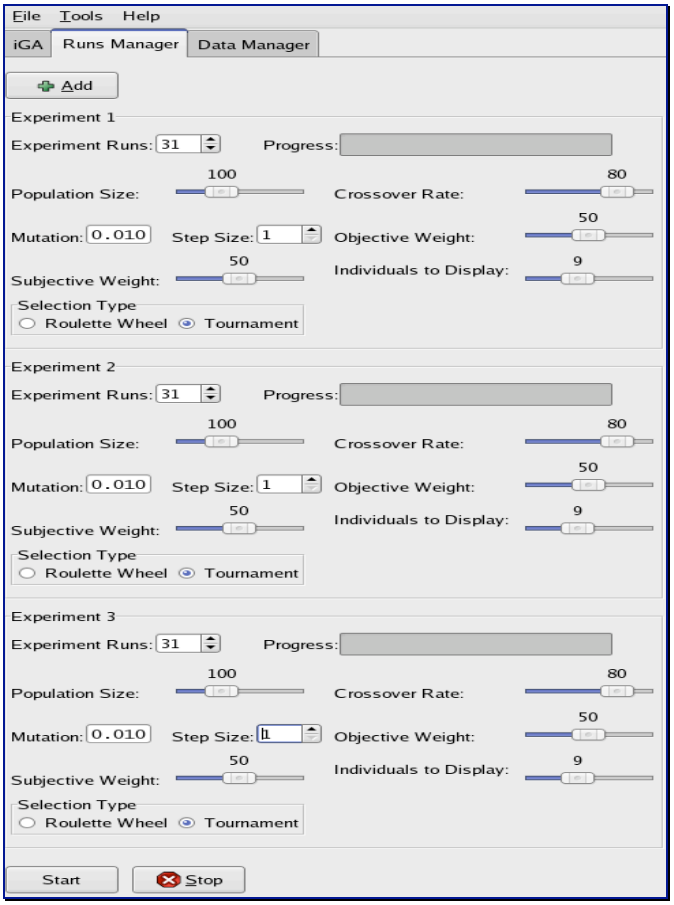

Fig. 4 The interface of the experiment runs manager.

In our previous work, an experiment consisting of 31 runs was done for each of the main results presented, which amounted to a lot of time setting up experiments in separate nodes in a cluster and tedious data management. The experiment runs manager developed to increase research productivity abstracts all that away, allowing the user to run as many experiments as necessary, with the ability to customize almost every aspect of the IGA for each experiment, parallelize the processing to have the experiments run as fast as possible, and automate the organization of the vast amounts of data resulting for each experiment.

C. *Data Manager*

The third improvement we have integrated in our environment is closely related to the experiment runs manager. The data manager allows the user to browse and explore the data produced from the many runs of each experiment intuitively. Each experiment and its corresponding runs are organized in a tree construct, with two visualization modes.

First, clicking on an experiment expands its children (the experiment runs) and displays an average plot of the results from the experiment. An example is shown in Figure 5. Second, clicking on one of the runs from an experiment displays the data in a spreadsheet, as shown in Figure 6.

Usually, a script is written to parse the data produced by the many IGA experiments. The data then needs to be fed into a plotting program, such as xgraph or gnuplot to view the results. The data manager takes care of retrieving and organizing the data from each experiment, and allows the user to rapidly make sense of the vast amounts of data through the plots. This third environment enhancement also saves a significant amount of time and makes easier the work of the researcher.

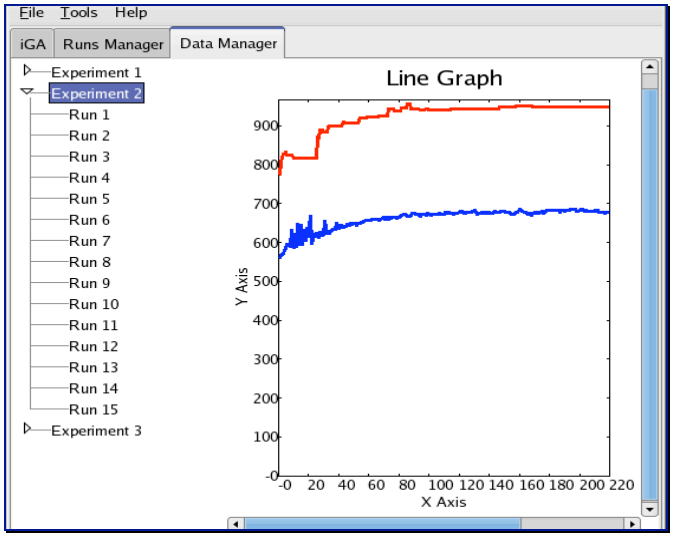

Fig. 5 The data manager organizes data resulting from experiments into a tree structure. Clicking on an experiment shows a plot of the experiment results.

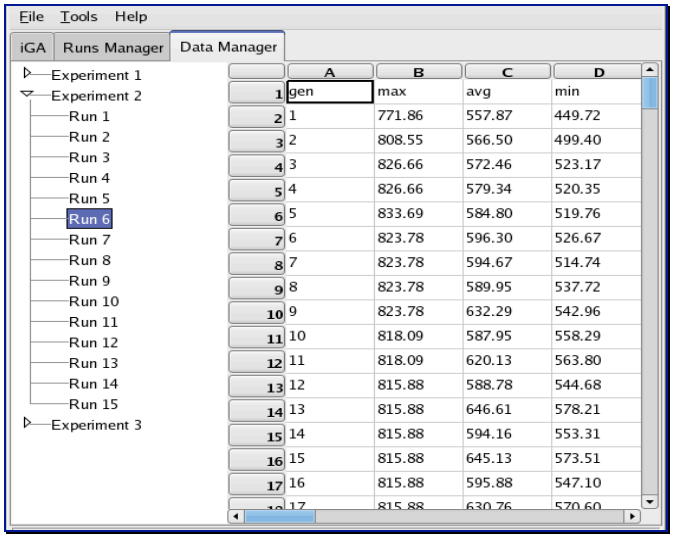

Fig. 6 The run view of the data manager.

D. *Future Productivity Improvements*

To further increase research productivity, we would like to incorporate in our environment a more intuitive way to define a UI to be evolved. An option would be to have the user define

a GUI in a development environment such as wxGlade or NetBeans, and have the representation of it loaded into our environment to be evolved. Another alternative is to have the user define the widgets to be evolved inside the environment, by presenting the user with a list of basic widgets and have the user simply drag and drop widgets into an empty panel. Once the panel was filled with the user desired widgets, then it can be evolved.

Also, we would like to further abstract the UI specification by allowing the user to specify the type of data that needs to be represented by the UI, and then have our tool evolve both the type of widget used to represent the data and the organization of the widgets.

Code generation of an interface design in the population is also necessary. When a user is satisfied with a design, it would be desirable to generate the code for the selected UI. The user also needs the ability to edit a UI design that is "good enough" to the user's preferences, such as moving widgets around, or picking a similar shade of color to the existing color.

IV. Transitioning from a Research Tool to an End User Tool

We would like not only to further improve the productivity of the environment but also to make it available to regular users (non-researchers). In order to conduct user studies and to deploy the system for widespread use, we need to address some usability issues. We foresee having two modes for the environment, an *end-user mode* and *a researcher mode*. There are advanced features which a researcher could use, such as the parameters and configuration of the genetic algorithm. However, the end-user (which, in our tool's case is a user interface designer) may not care or understand about such configurations, hence they need to be abstracted.

The researcher mode would allow for configuration of both high level and low level details, giving the researcher the complete control over how the IGA should behave. On the other hand, it does not make sense to present the end-user, a user interface designer, with a cluttered interface and configuration options that are bound to confuse and affect the systems' usability and engagement.

The UI designer should be presented with a minimalist interface, with an organization and representation that would be useful for users not familiar with IGAs. This can be accomplished by reducing technical jargon and presenting the user with leverage tools to achieve the desired goals, in this case the exploration of user interface designs. For example, on the context of UI design, it does not make sense to present the user with a slider for "crossover" and "mutation," since it does not correlate to the task at hand. A better approach would be to present the user with sliders for "variety," "creativity," or the degree to which the system should "stick to my choices!"

When designing a tool for researchers we need not shy away from presenting a plethora of configurations. We also allow the configuration of the UI through XML files. Hence, the advanced user (the researchers) need not search through the code to change the UI specification used as defaults.

The end-user tool would basically contain a subset of the functionality presented in the research tool. For example, allowing a user to change the degree of variety in the UIs presented to the user can be done in the background through higher crossover rates and an aggressive selection algorithm. For the advanced user, who wishes to explore how the degree of variety affects the population dynamics, he or she can switch to the advanced mode, and configure low and high level details of the IGA. The sets of features available in researcher mode and, respectively, end-user mode are summarized using use cases in Table I. It can be seen that in our tool's case, with the exception of the "Extended help" feature the end-user mode functionality is a subset of the researcher mode functionality.

The AI research community has developed several GA implementations, yet there is no encompassing framework that pieces them together. Furthermore, for users interested in the use of evolutionary techniques and with a weak programming background, it can be intimidating diving through hundreds of lines of code and customizing a GA to the problem at hand. Through our environment we hope to provide an efficient and usable front-end to both an GAs and IGAs, for the benefit of both the research and end-user communities.

TABLE I
USE CASES IN END-USER AND RESEARCHER MODES

	Use Case	End-User Mode	Researcher Mode
1	Define user interface	√	√
2	Load user interface definition		√
3	Customize high level IGA details	√	√
4	Customize low level IGA details		√
5	Start IGA	√	√
6	Stop IGA	√	√
7	Open IGA state	√	√
8	Save IGA state	√	√
9	Select best and worst UI	√	√
10	Undo evolution step	√	√
11	Redo evolution step	√	√
12	Edit evolved UI	√	√
13	Run batch mode		√
14	Extended help	√	

V. Related Work

In terms of environments to support GA-based evolution of user interface designs very little information is available in the literature. In fact, the only other reported work on evolving user interfaces is by Oliver, Monmarché, and Venturini [6] and by Monmarché et al [15]. However, this work explores

the evolution of website styles, while our work is focused on the evolution of layout and style of GUI widgets. Thus, our research is rather unique and, we believe, also very promising in terms of increasing UI design productivity via an IGA-driven evolution while incorporating both established guidelines of style and individual designer preferences [11], [12].

While unique in terms of specific research supported, our software environment can however be considered illustrative for two significant challenges faced by scientific researchers: first, how to balance the need for fast research results with the need for better research tools that could improve research productivity in the long run and, second, how to prepare the transition of tool used for research to a tool accessible by a general category of end users. Nevertheless, because we have started the exploration of literature for reports on the above two topics and found only rather few such reports so far [16], [17] [18], it seems these challenges are yet to be addressed thoroughly.

VI. Conclusions and Future Work

We have presented a software environment for research on evolving user interface designs and described three improvements to the environment aimed at increasing research productivity by automating tedious tasks that had to be conducted previously by the user. Because our research productivity has been significantly increased, we believe that in our environment's case investing effort in developing new features of the research software is beneficial in the long run.

In addition, we have presented a discussion on transitioning the research environment from a researcher's tool to an end-user's tool, and looked into how changes to the current environment could bridge the gap between these two types of tool.

We believe that the two challenges addressed in this paper, increasing research efficiency via additional tool development and preparing a research tool for transition to an end-user tool, are highly important to researchers and deserve thorough investigation. We believe there is a huge potential for numerous beneficial research and development projects in tackling these challenges.

In addition to planned improvements to our research tool described in subsection III.D, a direction of future work we intend to initiate soon is to quantitatively assess the increase in productivity brought by the improvements integrated in our environment (we will collect long term data for this purpose).

Also, we intend to develop an application programming interface (API) and generalize our IGA tool such that it could be used by the AI research community as a front-end to genetic algorithms. Lastly, we would like to elaborate a set of guidelines for researchers to be followed when preparing and distributing tools that can be used by the end-user community.

Acknowledgments

This material is based upon work supported by the Office of Naval Research under contract number N00014-03-1-0104 and upon work supported by the National Science Foundation under Grant No. 0447416.

References

[1] Apple Human Interface Design Guidelines. http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/

[2] GNOME Human Interface Guidelines 2.0. http://developer.gnome.org/projects/gup/hig/

[3] Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[4] Java Look and Feel Design Guidelines. http://java.sun.com/products/jlf/ed2/book/

[5] Lagoon – Research and Education Platform in AI, Evolutionary Computing Systems Lab (ECSL), University of Nevada, Reno, USA. http://lagoon.cse.unr.edu

[6] A. Oliver, N. Monmarché, and G. Venturini. Interactive design of web sites with a genetic algorithm. In *Proceedings of the IADIS International Conference WWW/Internet*, pp. 355–362, Lisbon, Portugal, 2002.

[7] H. Takagi. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proc. of IEEE*, **89** (9): 1275-1296, 2001.

[8] Windows XP – Guidelines for Applications. http://www.microsoft.com/whdc/Resources/windowsxp

[9] XULPlanet.com. http://xulplanet.com.

[10] H. Thimbleby. User interface design with matrix algebra. *ACM Trans. on Computer-Human Interaction*, **11**(2): 181–236, 2004.

[11] J. C. Quiroz, S. M. Dascalu, and S. J. Louis. Human guided evolution of XUL user interfaces. In *Proceedings of ACM CHI '07 Human Factors in Computing Systems*, San Jose, CA, USA, May 2007 (to appear).

[12] J. C. Quiroz, S. J. Louis, and S. M. Dascalu. Interactive evolution of XUL user interfaces. In *Proceedings of the 2007 Conference on Genetic and Evolutionary Computation (GECCO-2007)*, London, England, July 2007 (to appear).

[13] Benyon, D., Turner, P., and Turner, S. *Designing Interactive Systems: People Activities, Contexts, Technologies*. Addison-Wesley, 2005.

[14] W.C. Kim and J.D. Foley. Providing high-level control and expert assistance in the user interface presentation design. *In Proceedings of CHI '93, the ACM SIGCHI Conference on Human Factors in Computing Systems,* Amsterdam, The Netherlands, 1993, pp. 430-437.

[15] N. Monmarché, G. Nocent, M. Slimane, G. Venturini, and P. Santini. Imagine: a tool for generating HTML style sheets with an interactive genetic algorithm based on genes frequencies. In *Proc. of IEEE Intl. Conference on Systems, Man and Cybernetics*, 1999, pp. 640-645.

[16] L.A. Mallak, Using information technology to leverage research productivity. *In Proceedings of the 1996 International Conference on Engineering and Technology Management*, pp. 351-356.

[17] C. Ryan, B. Tewey, S. Newman, T. Turner, and R.J. Jaeger. Estimating research productivity in assistive technology: a bibliometrics analysis spanning four decades. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, **12**(4): 422-429, 2004.

[18] J.T. Yao and Y.Y. Yao. Web-based information retrieval support systems: building research tools for scientists in the new information age. In *Proceedings of the 2003 IEEE International Conference on Web Intelligence*, pp. 570-573.