Application of usability testing to GUIs in the electronic design automation industry

Kirolos George
Marwa Adel
Reem ElAdawi
Siemens EDA

Email: kirolos_george@siemens.com
marwa_adel@siemens.com
reem_eladawi@siemens.com

Abstract

A graphical user interface (GUI) enables software users to setup, initiate, and control software execution using visual elements, standardized task menus, and simple data entry to replace tedious and error-prone command line data entry. A GUI should enable users at all experience levels to select run conditions and run the software quickly and accurately. The usability testing goal is to determine how well a GUI satisfies three parameters: effectiveness, efficiency, and satisfaction. Usability evaluation helps product engineers, developers, and quality control engineers learn how all users use the GUI, determine user experience satisfaction, and identify opportunities for improvement. We focus on GUIs used with electronic design automation (EDA) tools (software programs) that help design companies and manufacturing foundries design, simulate, and verify both the physical construction and performance of integrated circuit (IC) chip designs, and predict the outcome of IC fabrication processes. We discuss the work in progress on EDA GUIs, and how usability testing techniques can be applied and automated to improve software quality and user experience (UX) to augment manual usability testing. We evaluate a GUI used to invoke an EDA tool, gather feedback, evaluate results, and prepare recommendations to enhance the UX. We employ usability test conventions and formal usability testing by monitoring 34 participants while they use the GUI. These activities result in the identification of 56 enhancement opportunities which, when implemented, improve both the perceived UX and the usability of the GUI, satisfying the three usability parameters of effectiveness, efficiency, and satisfaction.

Biography

Kirolos George: Senior software quality assurance/software testing engineer for Siemens Digital Industries Software, Calibre Pattern Matching team. Received his B.Sc. in Electrical Engineering from Communications and Electronics department Ain-Shams University, and MBA from ESLSCA Business School.

Marwa Adel: Senior quality assurance engineer for multiple products for Siemens Digital Industries Software. Received her BSc. from the Computers and Systems Engineering department of Ain Shams University.

Reem ElAdawi: Test engineer director at Siemens Digital Industries Software. Received her B.SC., M.Sc., and Ph.D. from the Electronics and Communication department of Ain Shams University.

1 Introduction

1.1 GUI usability

A graphical user interface (GUI) enables software users to setup, initiate, and control software execution using visual elements such as buttons and icons, standardized task menus and options, and simple data entry, replacing tedious and error-prone command line data entry. To reduce the risk of setup errors and simplify tool invocation, a GUI should enable users at all experience levels to select the appropriate run conditions more quickly and run the software accurately. While functional testing ensures that a GUI for an application under test performs exactly as expected using its designed functionality, usability testing determines how efficient, usable, intuitive, and learnable the GUI is in real-world conditions. The usability testing goal is to determine how well a GUI satisfies the three usability parameters: effectiveness, efficiency, and satisfaction.

Usability testing and evaluation are important tasks in the GUI design process and are considered among the more important aspects of software testing in this era of focus on the user experience (UX). Usability evaluation helps product engineers, developers, and quality control engineers learn how both expert and novice users use the GUI, determine how satisfied users are with their product experience, and identify and prioritize opportunities for improvement.

2 GUI usability in EDA

2.1 EDA industry

As digital transformation becomes the main theme and driver of many industries, software and hardware in the form of integrated circuits (ICs) now sit at the heart of product functionality. Advancements in the semiconductor industry allow IC designers to add more electrical circuits that support the new and expanded functionalities that are the main drivers of this digital transformation. The design of analog and digital electronic circuits, prototyping, testing, and production are the foundation of electronics engineering (Alajbeg and Sokele 2019).

In 1960, researchers for leading industrial and academic labs developed the first computer-aided design (CAD) tools to help and support engineers creating the layouts of electrical circuits (Brayton, et al. 2015). Since then, electronic design automation (EDA) tools continue to support and enhance IC engineer productivity through the dramatic reductions in IC size coupled with an equally dramatic growth in complexity and functionality. Many EDA tools exist to help electronic design houses and manufacturing foundries design, simulate, and verify both the physical construction and the designed performance of IC chip designs, and predict the outcome of IC fabrication processes, all before delivering a design for fabrication. These verification and simulation processes ensure that designed chips are constructed in compliance with foundry requirements, will perform according to design specifications, and will meet desired production yield targets. A billion-transistor chip can now be designed, debugged, and tested far more easily and quickly using sophisticated EDA tools that reduce time to market while ensuring product quality. Without the EDA industry, it is unlikely we would have the electronic devices and systems that are now a part of our daily lives (Lavagno, Martin and Scheffer 2006).

2.2 EDA GUI usability

Historically, EDA software engineers have been more concerned with developing the functional software that helps simulate highly sophisticated IC designs using complicated and difficult algorithms than focusing on the UX. This bias toward functional accuracy typically meant engineers were more likely to use technology-oriented methods rather than user-oriented methods (Nayebi, Desharnais and Abran 2012), which often resulted in GUIs that were not intentionally designed or optimized for usability.

Because usability testing in the EDA industry is still not commonplace, our focus is on graphical user interfaces (GUIs) used with EDA tools. We discuss and illustrate the work in progress on EDA GUIs, and how usability testing techniques can be applied and automated to improve the software quality and UX, and augment manual usability testing. We apply formal usability test conventions to evaluate the usability of the GUI in the Calibre Pattern Matching tool (Calibre Pattern Matching n.d.) from Siemens Digital Industries Software. Our research uses qualitative methods such as the 10 Nielsen usability heuristics evaluation and walkthrough. We monitor 34 participants while they use the Calibre Pattern Matching GUI, gather feedback, evaluate results, and prepare recommendations to enhance the UX with the current GUI. These activities result in the identification of 56 enhancement opportunities which, when implemented, improve both the perception of the UX and the usability of the Calibre Pattern Matching GUI, satisfying the three usability parameters of effectiveness, efficiency, and satisfaction.

3 Usability definitions and techniques

Usability as defined by ISO 9241-210 is the extent to which a software product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use (ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems 2019). Usability testing is concerned with examining and evaluating the user interface (UI). A UI contains all the components of the software product that provide information and controls for the user to accomplish certain tasks through the system. The objectives of usability testing are to:

1. Evaluate whether the usability requirements are satisfied or not
2. Discover any usability problem that requires correction
3. Measure the usability of a software product.

To deliver a high-quality user-friendly software product, a set of usability evaluations are performed iteratively:

a. **Formative** or **exploratory** evaluation is conducted early in the development cycle (typically during the design and wireframing or prototyping phases) to get ideas and guide the design
b. **Summative** evaluation is done just before or after implementation to measure the usability of the software product. It is a quantitative method, and it focuses on gathering measurements about the effectiveness, efficiency, or satisfaction of a software product.

All these evaluations are done to discover any usability problems, defined as defects that may result in difficulty when performing tasks through the UI. These defects impact users' ability to achieve their goals effectively, efficiently, and/or with satisfaction. Usability problems can lead to time waste, user dissatisfaction, confusion, delay, or failure to complete some tasks.

3.1 Usability components

Usability is counted as a quality attribute that evaluates how effective, easy, and user-friendly a GUI is. For our project, usability is defined by five quality components (Nielsen, Usability 101: Introduction to Usability 2012):

1. **Learnability**: Is the software easy to learn the first time users encounter it?
2. **Efficiency**: Once users have learned the software, how quickly can they achieve their goals in such software?
3. **Memorability**: When users return to the software after a period of not using it, can they easily remember how to use such software?
4. **Errors**: How many errors do users make, and how does the software help users recover from errors?
5. **Satisfaction**: How pleasant is it to use such software, and how likely are users to recommend such software to others?

There are many other quality attributes (Seffah, et al. 2006) (Umara and Ghazalib 2014), but one key additional attribute we include is (Nielsen, Usability 101: Introduction to Usability 2012):

6. **Effectiveness**: Can users achieve their desired goals using such software? Does the software do what the users need?

Our usability evaluation activities include usability reviews, usability testing and user surveys (Certified Tester Usability Testing (CT-UT) 2018), as shown in Fig. 1.

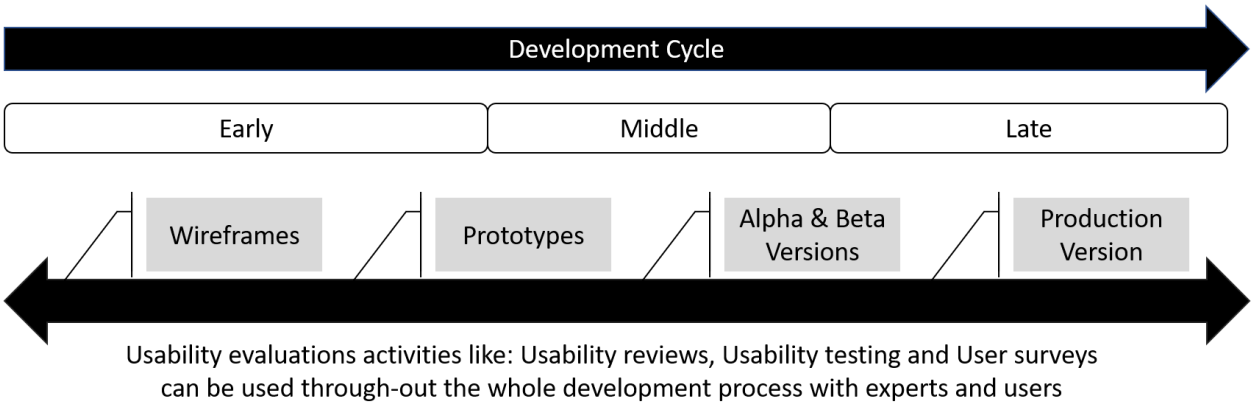

Fig. 1. Usability evaluation activities that can be used throughout the whole GUI development cycle to assess GUI usability.

3.2 Usability reviews

Usability reviews consist of three types:

a. **Informal** usability reviews, which can be conducted by anyone. An informal usability review is a usability review based on the judgment of one or more reviewers who examine or use a software product to identify potential usability problems. Informal usability reviews are often based on opinion, personal experience, and common sense.
b. **Expert** usability reviews, which are conducted by usability experts or subject matter experts (UX Expert Reviews 2018).
c. **Heuristic evaluation**, which is a usability review in which a small group of evaluators (preferably usability experts) inspect a software product and judge its alignment with well-recognized usability principles (heuristics), identifying where the product does not follow those heuristics. A usability heuristic is a generally recognized rule of thumb that provides reliable and useful guidance to a reviewer during the usability evaluation of a software product. Jakob Nielsen created a widely recognized list of 10 heuristics to ensure ease of use and maintenance. Complying with these heuristics ensures the software product meets the three main specified goals of usability testing which are: effectiveness, efficiency, and satisfaction (Nielsen, 10 Usability Heuristics for User Interface Design 1994). Jakob Nielsen recommends using three to five evaluators to lead to the maximum benefit, as one evaluator typically discovers only 35% of usability problems (Barnum 2020). Heuristic evaluations are considered cheap, quick, and easy to do, and can usually identify approximately 50% of real usability problems (Au, et al. 2008).

It is common to combine informal and expert usability reviews with heuristic evaluation when evaluating usability.

3.3 Usability testing

Usability testing is likely to identify most major usability problems, as it invites real users to execute real tasks under observation, where usability issues can be seen and recorded. Users are encouraged to

verbalize their actions, which aids discovery of even more usability problems. Studies show that six to eight test users are sufficient to uncover real usability problems (Au, et al. 2008). Usability testing watches a single person using the software product or a prototype to detect what and where pain points occur that confuse and frustrate the user. Results are then used to propose solutions for such usability problems (Krug 2014).

Usability testing has four steps (Barnum 2020):

1. Preparing usability test
2. Conducting usability testing sessions
3. Analyzing the findings
4. Communicating the findings with stakeholders.

Usability testing consists of a series of usability test sessions. In each session, a usability test participant performs representative tasks on the software product, or a wireframe or prototype of the software product. A test session is conducted by a moderator (a neutral person) and observed by observers. Usability testing should be done under conditions as close as possible to those under which the software product will be used, such as a mock office, a lab, or a home office environment. When possible, observation of usability test sessions should be done from a neighboring room so that observers and/or stakeholders can observe the effect of the actual software product on real people without influencing the test. Preparations for a usability testing begin with writing a usability test plan, which is a short description of the key tasks to be tested. A usability test script specifies a sequence of actions for the execution of a usability test. It is also a checklist used by the moderator of a usability test, containing the following information:

1. Activities for preparing the usability test session before the test participant arrives
2. Briefing instructions
3. Pre-session interview questions
4. Usability test tasks
5. Post-session interview questions

During the usability test session, the note-taker records usability observations, usually by writing them down. Usability observations reflect both events that cause problems or that have a positive effect on effectiveness, efficiency, and satisfaction. After all usability test sessions have been completed, the moderator and the note-taker each separately extract the usability findings from their observations. These findings reflect the observations that they consider most important. The moderator and the note-taker then meet to discuss their findings and create a usability test report consisting of a merged list of the usability problems and findings. These findings are shared and discussed with stakeholders (product development and product owners) The moderator logs the agreed-upon findings in the defect tracking tool, tracks the problems to resolution, and reviews the implemented solution. If the implemented solution represents a risk, it should be subject to another usability test.

3.4 User surveys

A user survey is a usability evaluation activity whereby a representative sample of users are asked to report subjective evaluation into a questionnaire based on their experience in using the software. User surveys can be used to evaluate the levels of user satisfaction with a software product. Surveys are useful as they allow users to participate anonymously, and they are generally a cheap and easy tool. Surveys can include various demographics of people to compare their answers (Marsh 2016).

4 Usability testing challenges in the EDA industry

Usability testing is a very challenging activity in the EDA industry for numerous reasons:

1. Customers use the EDA tools when simulating their designs, which are confidential and cannot be exposed outside their companies or disclosed or shared with anyone

a. Cannot easily access real users in their normal environment (Bezerra, et al. 2014)
b. Sometimes we can't know the exact way users use our tools

2. Some users are used to traditional methods of using EDA tools and are reluctant to adopt new practices
3. Product development team members usually focus on functionality and performance rather than usability (Nayebi, Desharnais and Abran 2012)
4. Can be difficult to justify investments in usability and UX efforts to management (Sauro 2016)
5. Cannot build code within the software to track most-used features
6. Pressures of time and budget considerations (Bergstrom, et al. 2011)
7. Difficult to recruit and bring in real users to participate in usability testing
8. Multiple personas (e.g., foundry engineers vs. design engineers) may use the same EDA tool in different environments and conditions

5 Introduction of new UX process

5.1 Adopting usability testing

A GUI consists of components (widgets) that provide information and controls for the user to accomplish specific tasks with the software tool. We have been performing GUI functional testing and running automated functionality tests for years. We have functional specifications for all tools, so we begin by writing a functional test plan, then perform functional testing. This kind of testing determines if the GUI functionally behaves as it is designed to do.

Testing how easily or intuitively a GUI performs for users is usability testing. A software product can work exactly as intended in the specifications and still have serious usability problems, such as:

- Complicated/unclear error messages
- Unclear use model/steps of operations
- No enabled/disabled buttons to help guide the user
- Bad/poor look and feel
- No undo/redo to correct user mistakes
- Extra/unnecessary steps that confuse the user

While most product design teams address usability at the end of the product development life cycle, we wanted to address usability throughout the product development life cycle, as shown in Fig. 1. We started by adding usability testing to our test process. We used the three usability evaluations activities—usability reviews, usability testing, and user surveys—to gather feedback from both internal team members and real users to evaluate the usability of the GUI used in the Calibre Pattern Matching tool (Fig. 2).

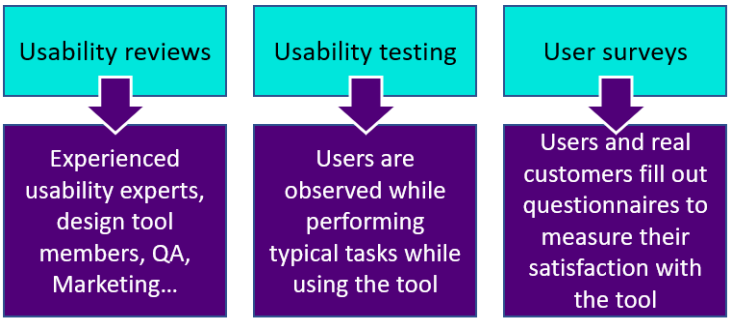

Fig. 2. A variety of processes were used to collect feedback on GUI usability.

The usability test is a black box technique for measuring the overall UX when using the tool. The outcome of a usability test is a report or list of findings that includes:

- Measurements of how easy/difficult, intuitive, and learnable the tool is (based on the usability components discussed in 3.1)
- Action items and bug/enhancement reports to define changes to the tool based on the feedback received

Our GUI usability testing was based on:

1. UI design patterns/usability standards (for example, input feedback, Undo/Redo actions, and usage of module tabs)
2. Usability requirement checklist

The usability requirement checklist we adopted (Table 1) contained 10 heuristics based on the Nielsen heuristics discussed earlier, in addition to a check to ensure the development team provides the whole team with wireframes or low-fidelity prototypes to assess.

Table 1. Usability heuristics

Usability heuristic	**Desirable characteristics**
Visibility of system status	Visible, relevant, timely feedback to keep users informed of system operations and status.
Match between system and real world	System messages use words, phrases, and concepts familiar to users, and display in a natural and logical order.
User control and freedom	System provides clear and simple undo and redo functions to enable users to correct inadvertent usage mistakes without complicated or lengthy steps.
Consistency and standards	System employs consistent usage across all operations that complies with platform standards.
Error prevention	When irreversible or potentially problematic events cannot be eliminated from GUI by design, system provides review/confirmation option before users commit to the action.
Recognition rather than recall	System makes objects, actions, and options visible across all operations to simplify user recognition. Instructions/help for system use are visible or easily retrievable when needed.
Flexibility and efficiency of use	System provides “accelerators” that can be accessed by expert users to speed up interactions and tailor frequent actions, while still supporting novice users with full guidance.
Aesthetic and minimalist design	Dialogues contain only the minimum of language needed to achieve the task.
Help users recognize, diagnose, and recover from errors	Error messages are expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.
Help and documentation	Support/help information is easy to search, is focused on the user’s current task, lists specific steps to be executed, and contains complete but minimal language.

We adopted an informal UX testing approach in our overall testing techniques, using the following activities:

1. Include UX testing in functional test plans.
2. Validate the usability of the GUI under test against UX components and standards.
3. Add automated test scripts to simulate user actions in the GUI tool and output the results data as part of the test validation.
4. Regularly run automated usability testcases against the GUI to guarantee consistency across versions and updates.
5. Review wireframes and low fidelity prototypes provided by the development team for evaluation and assessment.

5.2 Usability testcases automation

Evaluating the usability of new features can be done using multiple processes. We can manually walk through the different user scenarios of the feature, use batch application programming interface (API) functions to access its different functionalities, or add automated testcases that simulate the required tests.

Testing the usability of an application under test (AUT) is now performed as part of the regular GUI functional testcases automation. To support our GUI usability evaluation process, we added usability checks to our automated testcases that address the functional testing for our products for the flows we want to check for usability. These usability checks simulate user actions in the GUI, such as clicking on different widgets to perform steps that must be executed to reach or perform a specific functionality, then dump their content and state (enabled/disabled) to evaluate the results.

The main goal of adding usability checks to these automated testcases is to ensure we consistently test the GUI tool against the UX conventions and the 10 Nielsen heuristics to confirm its usability remains intact. In the event of future code changes, the previous tool behavior acts as a baseline to enable us to determine if the code changes result in a “failure” of usability compliance that requires remediation.

We also added a dedicated usability testing section inside our test plans for tool features to address the level of testing that needs to be done to ensure tool use is effective and intuitive.

For example, testing a new dialog includes the following steps:

1. Check that the new dialog invocation is accurately placed
2. Add automated testcases to dump the default values of the dialog
3. Dump the tool tips of all widgets to check text consistency from run to run or identify text changes
4. Dump and check the state of the widgets, (selected/unselected) and/or (enabled/disabled)
5. Check the number of steps required to perform or reach target functionality to look for unnecessary steps
6. Dump all warnings and error messages to check that text is comprehensive, clear, intuitive, and correct
7. Check and verify that generated messages include updated data for any values that are changed
8. Dump the font type and font colors of the dialog widgets and check against usability standards
9. Check the background color for the drawing area to ensure the text is easily readable
10. Take screenshots of the drawing area and save them for image comparisons
11. Use the undo/redo functions of the tool to ensure dialogs retrieve previous input(s) correctly
12. Ensure keyboard actions result (like pressing the tab buttons) result in the desired function (including forward and backward, where applicable), and that keyboard shortcuts work properly

We run these automated testcases as usual on different platforms/environments, as part of the complete functional test suite regression. They can fail if any of the usability conditions are not met, or if unexpected code changes were applied. These tests can be updated as needed with GUI enhancements.

Clicking the widgets and dumping their content is performed using scripts (typically Python). During the testcase execution, all Python commands are performed in the same order, and dumps are compared after the testcase is fully run. The testcase will fail in any of these conditions:

- The script failed to access any widget because it’s invisible, disabled, or moved to a different frame/dialog
- The content of any dump is different from its saved (baseline) version
- Any messages or output of the test are changed
- Any GUI behavior has changed

5.2.1 Mapping UX heuristics with automated test scenarios

We try to measure our usability heuristics against all new features, using automated usability tests to avoid any unneeded or random outcome of the GUI. With reference to the heuristics checklist in Table 1, we continuously add automated scripts to check and verify usability.

Table 2. Automated usability verification

Usability heuristic	**Automation**
Visibility of system status	Dump AUT messages and warnings. Dump AUT status bar.
Match between system and real world	Verify all labels, error and warning messages, tool tips use simple, real-world language. Verify all error and warning messages provide clear guidance for accurately performing task. Dump label names. Dump error and warning messages.
User control and freedom	Verify tool dialogs include specified or required user controls (ex: Cancel, Apply, Undo, Redo buttons). Dump user control state status (e.g., enabled/disabled). Verify user controls are enabled and clickable when needed.
Consistency and standards	Verify labels, tooltips, messages, and warnings, generated by the GUI are comprehensive, correct, and compliant with Calibre product standards. Verify similar functionalities in various Calibre products use the same shortcuts and are placed in similar drop-down menus.
Error prevention	Verify unneeded widgets are disabled. Verify tool prevents bad data (e.g., special characters in line edit fields). Verify tool provides clear warning/opt-out message to users before performing any deletion action to prevent data loss. Verify dialog widgets include only specified/standard actions (e.g., Apply, Cancel, Close buttons). Verify provided error messages provide corrective guidance and prevent actions likely to fail. Verify warning and error messages ask for confirmation of risky/irreversible task before it is performed. Dump all widgets' status and verify they are only enabled in correct order of testcase execution.
Recognition rather than recall	Verify user choices are recognized and saved for future checks. Dump tooltips and verify they provide accurate, clear guidance to users for performing specified scenarios. Dump tool transcripts and verify user choices are printed correctly to minimize the need for user memory recall. Verify indicators provide users with flag if anything is edited/changed (Fig. 3). Dump dialog fields before and after user actions and verify AUT remembers user's choices. Dump selected menus and verify the last-edited items appear first in the list.
Flexibility and efficiency of use	Dump selected menus and verify the last-edited items appear first in the list. Use AUT shortcuts and verify they function correctly and invoke the correct dialogs.

	Test the search and filter features to verify user can easily and quickly reach intended data. Dump dialog fields and verify default values are set correctly to speed up the process and enable user to perform intended goal efficiently.
Aesthetic and minimalist design	Dump widget frames and windows sizes to verify GUI provides the same view on different environment (OS, machine resolutions, platforms). Dump font styles and file window types (native or modified) to verify GUI provides the same view on different environment (OS, machine resolutions, platforms). Verify users can use menu button(s) to select which dialogs to view, keeping unselected dialogs hidden (Fig. 4).
Help users recognize, diagnose, and recover from errors	Dump messages and warning messages and verify they provide accurate error correction guidance. Verify selected actions can be performed using multiple alternative methods in the GUI.
Help and documentation	Dump GUI help messages and options and verify they are comprehensive, accurate, and easy to read. Dump tool tips and help buttons and verify the information accurately guides users to correct tool use.

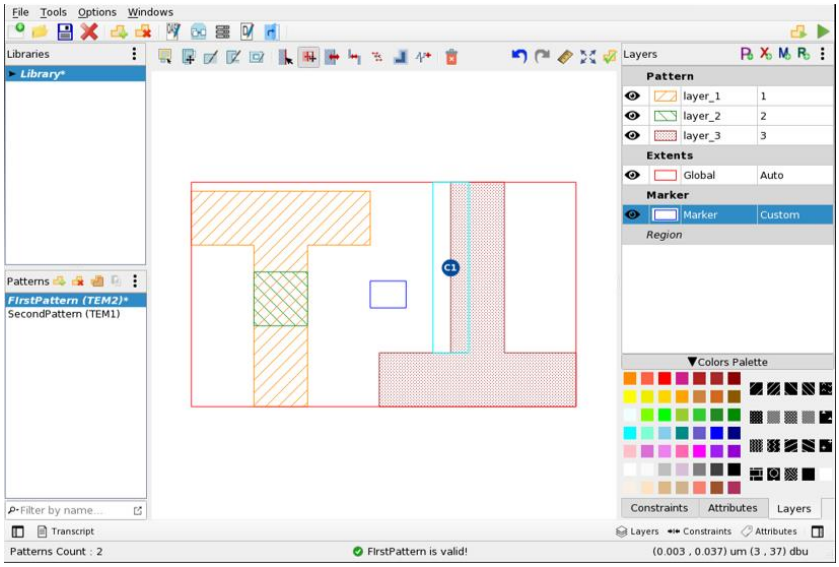

Fig 3. Verify flags are used to indicate item is edited but not yet saved. Under Libraries, “Library” is displayed in bold and italic format and has an asterisk. In Patterns, “First Pattern (TEM2)” is displayed in bold and italic format and has an asterisk. These flags indicate those items have changed, but have not been saved.

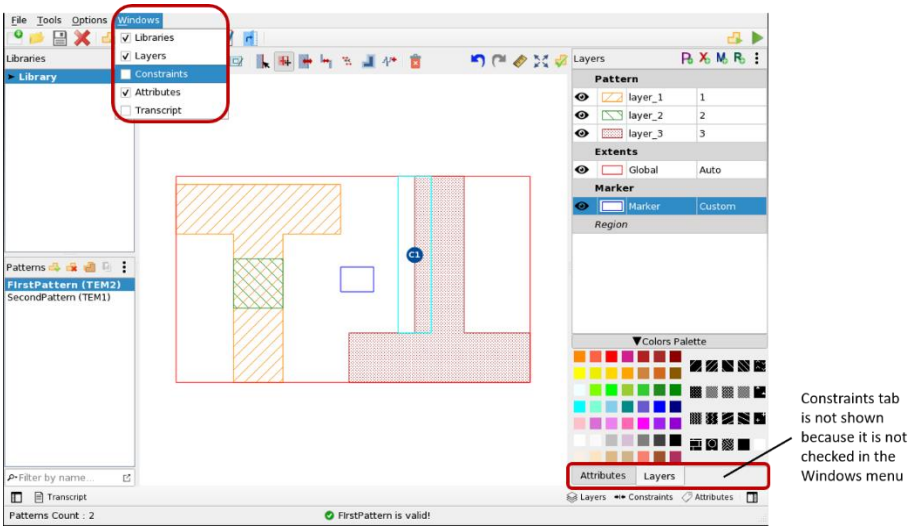

Fig. 4. Verify user can choose which widgets to view. The constraints tab is not displayed because it was not checked in the menu options.

6 U2U success story

6.1 U2U usability testing

Siemens product usability labs are part of a formal usability testing event held during the annual Siemens User-to-User (U2U) conference (User2User n.d.). During the U2U usability labs, product teams can directly engage customers to collect feedback on new product enhancements or usage flows.

6.1.1 Usability test lab

Setting up and running a usability test lab involves multiple steps:

1. The product team (including quality assurance/software tester(s), developer(s), and product owner) identifies a product/features to be tested for usability.
2. The product team prepares exact usage scenarios for the product/features.
3. The product team prepares a set of questions about the feature(s) for which they want to gather customer feedback to measure user satisfaction with the GUI. These questions are all related to the product UX while performing the scenario steps.
4. Computers are set up with the tools, executables, test data, and usage scenarios.
5. Two staff members are assigned to each station. One serves as a moderator, and the other is available to answer in-depth technical questions and troubleshoot if issues arise with the product itself.
6. Users from different expertise levels are invited. Some are customers actively using the product, while others are potential customers, or individuals with no previous experience with the tool.
7. Users are instructed to perform the specific tasks in the usage scenario:
    a. Using a specified order of steps
    b. Using specific test data
8. Users are asked to provide their feedback to session moderator using the think aloud technique.
9. Users can ask the technical person questions about the product or any issues they encounter.
10. Users agree to answer specific questions related to the exercise/task they performed.
11. Both staff members observe users and take notes as the users go through the lab and ask questions

6.1.2 Data collection and analysis

After the U2U usability lab event:

1. Test feedback is collected, and data is extracted and analyzed.
2. A final report is prepared containing the findings of the usability test, including difficulties encountered by users, recorded observations, questions asked by users during testing, and user feedback.
3. The report is presented to the product team.
4. Product team uses report data to identify and implement GUI changes to improve the UX.
5. All the usability test labs are added into the GUI test suite as automated testcases to automate and perform the exact steps users were asked to perform in the U2U lab event.

6.2 Usability testing results

Three usability evaluations activities—usability reviews, usability testing, and user surveys—were used to gather feedback from internal team members and real-world users to evaluate the Calibre Pattern Matching GUI. Mixing quantitative and qualitative methods helped provide an understanding of how the UX affects brand attitudes, letting us measure what people think against what they do. Using the information gleaned from usability evaluations contributed to identifying several usability gaps and opportunities in the Calibre Pattern Matching GUI, resulting in 56 distinct enhancement tasks. While implementing the enhancements, we were able to watch participants interact with the wireframes (a basic blueprint for the software product) (Marsh 2016) and prototypes, as well as the final product, to determine what worked and what did not (Bergstrom, et al. 2011) (Norman 2013). Implementing these changes contributed to a 28% increase in UX satisfaction, as calculated by dividing the number of the filed usability items by the number of filed functionality items within a certain time span.

6.2.1 Findings caught by usability testing and the ten Nielsen heuristics and informal usability review

A comparison of the previous pattern capture tool, in which users had to select which capture mode to apply, to the new pattern capture tool after deploying the usability evaluations enhancement is shown in Fig. 5. The new view satisfies Fitts' law (Interaction Design Foundation n.d.), which states that the amount of time required for a person to move a mouse to a target area is a function of the distance to the target divided by the size of the target. Therefore, the longer the distance and the smaller the target's size, the longer it takes. The pattern capture tool was redesigned to add a visual display of capture mode options inside a much larger button, making it easier, more intuitive, and faster for users to review options and select the desired capture method. In addition, we reduced the potential number of clicks by setting a default value (the current directory) for the output file path. If the default value is acceptable, the user can proceed with the minimum number of clicks; otherwise, the user has the flexibility to either select a certain area or enter the desired coordinates and the required cell name.

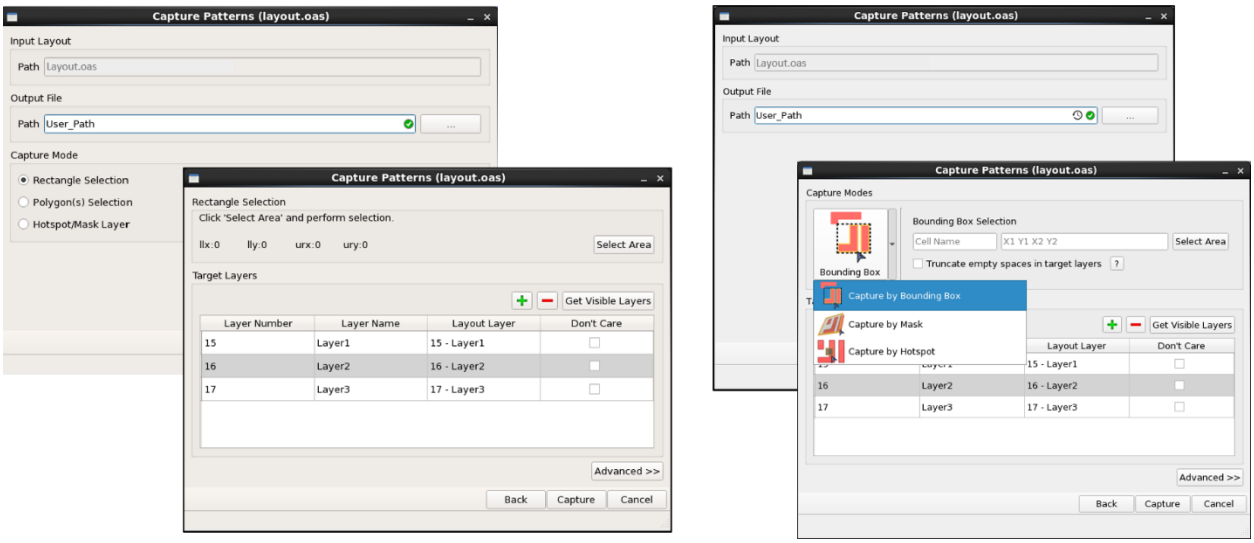

Fig. 5. Comparison of the old pattern capture GUI (left) to redesigned GUI (right). Visual selection options replace text-based options for faster, easier selection.

6.2.2 Findings caught by the Nielsen heuristics

Using the Nielsen heuristics to analyze usability revealed that the previous Calibre Pattern Matching main content viewing window required users to click either a polygons tab or constraints tab to choose among set of tools to use. By enhancing this content window to minimize the number of clicks and provide users with more control and freedom, the enhanced GUI provides users with a visual and intuitive set of frequently used tools, eliminating the secondary need to choose between polygon and constraints tools as shown in Fig. 6. This change better satisfied two of the Nielsen heuristics:

- Recognition rather than recall
- Aesthetic and minimalist design

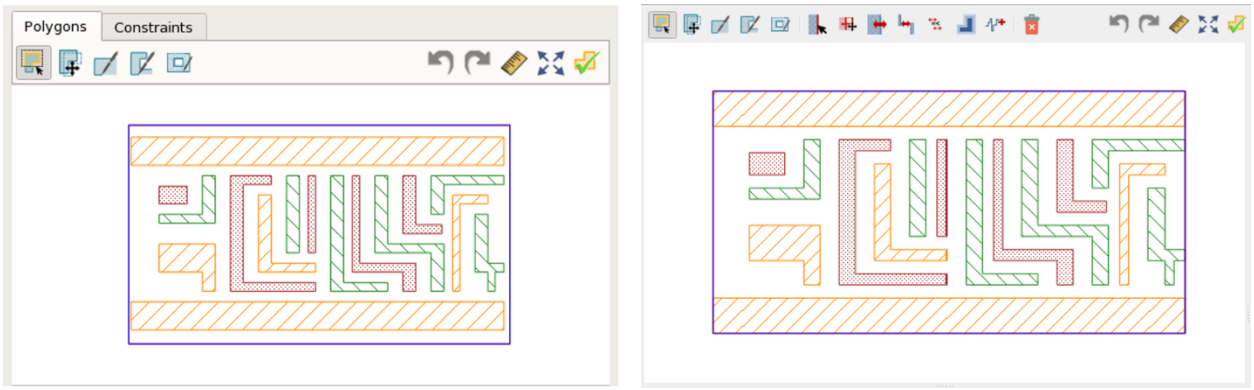

Fig. 6. Compared to the previous Calibre Pattern Matching main content viewing window (left) that required multiple clicks to select tools, the enhanced version (right) lets users select any tool quickly and easily using visual selection options.

6.2.3 Findings caught by usability testing

In the previous Calibre Pattern Matching Compile Library viewing window, users had to remember not to close the results window, so they could access the results multiple times without having to re-run to reach the results again. This usability bug was caught through the User2User Usability labs while the QA, development and product engineering team members were monitoring real users while they used the GUI

as shown in Fig. 7. After fixing this usability bug, a “Results” tab was added as shown, allowing users to choose whether to edit the run setting from the setup tab or to access the results from the Results tab.

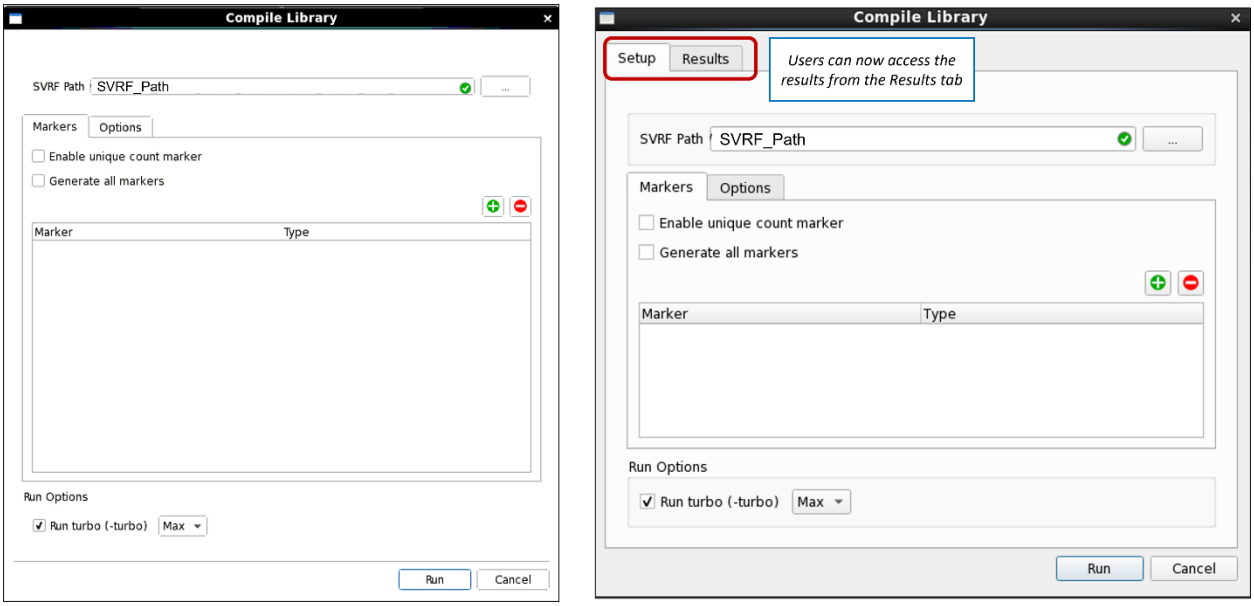

Fig 7. The Calibre Pattern Matching Compile Library window was enhanced to simplify results access based on findings obtained from usability testing.

6.2.4 Contribution of each usability evaluation method

The contribution of each usability evaluation method in identifying new enhancements for the Calibre Pattern Matching tool is shown in Fig. 8. Our usability testing activities contributed to the delivery of a premium quality GUI that eases the workload of users by helping them achieve their goals with a more intuitive UX (Walia and Casey 2021).

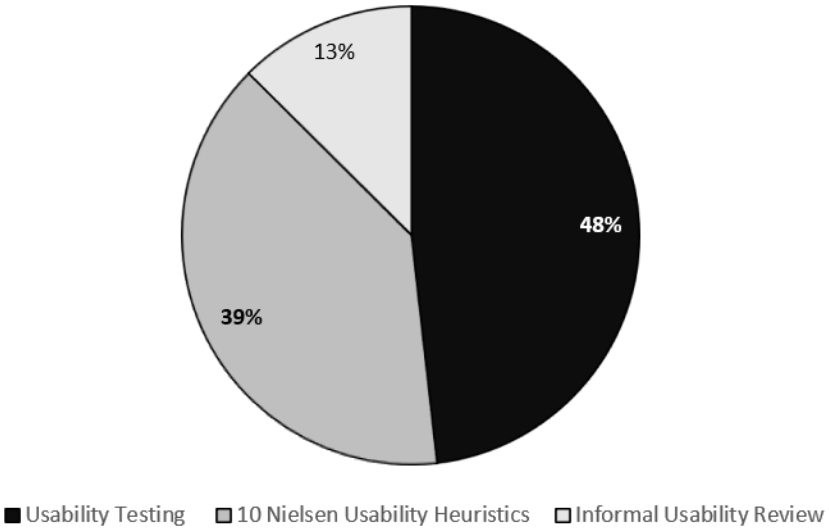

Fig 8. Percentage contributions of each usability evaluation method to improvements in UX.

7 Conclusion

Usability evaluations are an important component of today's software testing process, especially in the EDA industry. EDA tools contain sophisticated functionality, and their complexity increases with each new semiconductor technology manufacturing node. Poor usability not only affects brand image, but also lowers the productivity of users and blocks them from achieving their desired goals in an effective and efficient manner. Automating usability tests helps organizations deliver a high level of ease of use by providing user-friendly software that contributes to a positive holistic experience of users during and after their use of the tool. It also improves testing efficiency and makes it easy to integrate usability within the regression suite and the development process.

Enhancing the usability and the UX of software is not only the responsibility of the software testing/quality control/quality assurance teams, but also the development, software testing, and product owner team members. It is a common vision and effort that contribute to enhancing usability, starting from questioning any and all aspects of the tool design, to evaluating the software product with real users and gathering their feedback using multiple usability evaluation methods, preparing enhanced wireframes and prototypes and testing these in accordance with the usability heuristics, performing usability testing activities and analyzing the output, until launching the production version of the enhanced software. Usability analysis requires the use of suitable methods to understand user behavior and perception and to identify user needs and challenges to deliver a useful, effective, and efficient software product. Implementing and deploying a software process that includes usability analysis within the software product life cycle helps identify and eliminate potential usability problems early in the software development cycle.

We used multiple usability analysis methods to evaluate the usability of the Siemens Calibre Pattern Matching GUI, resulting in 56 distinct enhancements that resulted in improved performance, ease of use, and product perception. By developing and employing an automated usability testing framework within our complete functional test suite regression, we can quickly update our usability requirements and checks as needed in the future to ensure all our products satisfy the three usability parameters of effectiveness, efficiency, and satisfaction.

References

Alajbeg, Trpimir , and Mladen Sokele. 2019. "Implementation of Electronic Design Automation software tool in the learning process." International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). Opatija, Croatia: IEEE. pp. 532-536. doi:10.23919/MIPRO.2019.8757096.

Au, Fiora T. W., Simon Baker, Ian Warren, and Gillian Dobbie. 2008. "Automated usability testing framework." *The Ninth Conference on Australasian User Interface.*

Barnum, Carol M. 2020. *Usability Testing Essentials: Ready, Set...Test!* San Francisco, United States: Elsevier Science & Technology .

Bergstrom, Jennifer C. Romano, Erica L. Olmsted-Hawala, Jennifer M. Chen, and Elizabeth D. Murphy. 2011. "Conducting Iterative Usability Testing on a Web Site: Challenges and Benefits." *Journal of Usability Studies.*

Bezerra, Carla, Rossana M. C. Andrade, Rainara Maia Santos, Mourad Abed, Káthia Marçal de Oliveira, José Maria Monteiro, Ismayle Santos, and Houcine Ezzedine. 2014. "Challenges for usability testing in ubiquitous systems." *IHM '14: Proceedings of the 26th Conference on l'Interaction Homme-Machine.* 183–188.

Brayton, Robert, Luca P. Carloni, Alberto L. Sangiovanni-Vincentelli, and Tiziano Villa. 2015. "Design Automation of Electronic Systems: Past Accomplishments and Challenges Ahead [Scanning the Issue],." IEEE. pp. 1952-1957. doi:10.1109/JPROC.2015.2487798.

n.d. *Calibre Pattern Matching.* Accessed July 7, 2022. https://eda.sw.siemens.com/en-US/ic/calibre-design/physical-verification/pattern-matching/.

2018. "Certified Tester Usability Testing (CT-UT)." Accessed July 17, 2022. https://www.istqb.org/certifications/usability-tester.

Interaction Design Foundation. n.d. *Fitts' Law .* Accessed August 12, 2022. https://www.interaction-design.org/literature/topics/fitts-law.

2019. *ISO 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems.* July. Accessed July 7, 2022. https://www.iso.org/standard/77520.html.

Krug, Steve . 2014. *Don't Make Me Think, Revisited A Common Sense Approach to Web Usability.* New Riders.

Lavagno, Luciano , Grant Martin, and Louis Scheffer. 2006. *Electronic Design Automation for Integrated Circuits Handbook.* CRC Press.

Marsh, Joel . 2016. *UX for Beginners.* Canada.: O'Reilly Media.

Nayebi, Fatih, Jean-Marc Desharnais, and Alain Abran. 2012. "The State of the Art of Mobile Application Usability Evaluation." *Canadian Conference on Electrical and Computer Engineering.* IEEE. doi:10.1109/CCECE.2012.6334930.

Nielsen, Jakob. 1994. *10 Usability Heuristics for User Interface Design.* April 24. Accessed July 17, 2022. https://www.nngroup.com/articles/ten-usability-heuristics/.

—. 2012. *Usability 101: Introduction to Usability.* January 3. Accessed July 7, 2022. https://www.nngroup.com/articles/usability-101-introduction-to-usability/.

Norman, Don. 2013. *The design of everyday things.* Basic Books.

Sauro, Jeff . 2016. "The Challenges and Opportunities of Measuring the User Experience." *Journal of Usability Studies.*

Seffah, Ahmed, Mohammad Donyaee, Rex B. Kline, and Harkirat K. Padda. 2006. "Usability measurement and metrics: A consolidated model." *Software Quality Journal.* doi:10.1007/s11219-006-7600-8.

Umara, Muhammad Aminu , and Masitah Ghazalib. 2014. "Investigation into Usability Attributes for Embedded Systems Testing." *International Journal of Engineering and Technology.*

n.d. *User2User.* Accessed August 1, 2022. https://events.sw.siemens.com/en-US/u2uconference/.

2018. *UX Expert Reviews.* February 25. Accessed July 17, 2022. https://www.nngroup.com/articles/ux-expert-reviews/.

Walia, Ritu , and John Casey. 2021. "QA Best Practices: GUI Test Automation for EDA Software." *PNSQC .* PNSQC .