Jan 17th, 12:00 AM

A user involvement measurement technique for user involvement metrics

Parinitha Nagaraja
Siemens Corporation, United States of America, parinitha.nagaraja@siemens.com

Helmut Degen
Siemens Corporation, United States of America, helmut.degen@siemens.com

A user involvement measurement technique for user involvement metrics

Parinitha Nagaraja[1] and Helmut Degen[1]

Siemens Technology, Princeton, USA
{parinitha.nagaraja,helmut.degen}@siemens.com

Abstract. Efficiency of use has always been a topic of interest for human-computer interaction and in usability. One of the challenges is to be able to measure the efficiency or inefficiency thereof for different types of user involvement including user actions and system actions. The paper proposes a technique to measure user involvement times based on a user involvement taxonomy [1]. The technique uses tracking of mouse and keyboard events as well as the software application process start time. The measurement technique is applied to the Windows Calculator for a simple addition task. The measured times are reported. The elapsed user involvement time is 5.827 seconds. 5.216 seconds (89.5% of elapsed involvement time) is allocated to elapsed hands-on time and 0.611 seconds (10.5% of elapsed involvement time) for elapsed waiting time. The measured elapsed waiting time (0.611 seconds) and the average waiting time (0.122 seconds) are in line with Seow's benchmark boundaries [2]. The user involvement metrics provide context for the individual measured events and make them interpretable and usable.
Keywords: User involvement, efficiency, metrics, measurement technique, Windows application

1 Introduction

Efficiency of use has always been a topic of interest for human-computer interaction and in usability. There are still many problems unsolved, particularly for emerging user behaviors. Users are increasingly involved in semi-automated, parallel executed processes. Many users perceive current waiting times and system requests for user decisions as unnecessarily long (waiting times, inappropriate necessity of monitoring) or as unnecessary (irrelevant decisions). Design decisions and resulting system behavior can lead to long waiting times and unnecessary decision requests. Such an undesirable behavior impacts negatively the enterprises' need for high productivity and the user's need for efficient task completion. From that perspective, it is beneficial to minimize mandatory user involvement in the design process and, at the same time, to keep the user sufficiently in control of the process and its outcome. This should also consider the trend that users take part in several semi-automated processes or are required to use several systems in parallel.

To determine the current efficiency of user involvement, a measurement technique is needed. Inefficiencies can be caused by system actions, user actions, or both. Therefore, we need a technique that can measure both. Known techniques measure "time-on-tasks"

17th International Conference on Wirtschaftsinformatik
February 2022, Nürnberg, Germany

(summative usability tests) or response times (performance testing). Such tools and techniques do not cover mentioned enhanced user involvements types. There is a rich body of knowledge regarding response time. However, many of them are captured in lab situations. “They [response time estimates] should, indeed, be verified by extended systems studies-not in artificial laboratories using abstract tasks-but in carefully designed, real-life task environments and problems” [3]. The proposed measurement technique addresses the limitation, mentioned about 50 years ago.

This paper proposes a measurement technique and implementation to measure enhanced types of user involvement. The technique is based on user involvement metrics taxonomy [1] and applied to a simple example. The initial implementation as described in this paper focuses on user interaction metrics only.

In section 2, enhanced forms of user involvement are introduced. The metrics introduced in section 2 are used as evaluation criteria for related work that is discussed in section 3. Section 4 describes the measurement technique and experiment. Section 5 discusses the results and outlines future steps.

2 Intended Use

2.1 Evolved user involvement

The first personal computers were launched in the 1970s and 1980s. The introduction of a graphical user interface (GUI) for the mass market 1984 was a break-through to make computers usable for ordinary people, and not only computer scientists or computer enthusiast [4]. Till around the end of the 1990s, the use of PCs can be characterized by:

- Context of use: PCs were practically immobile and in one fixed location
- User - device mapping: In offices, there was a single PC for a single user. In private homes, one PC was often shared.
- Application: device mapping: Applications were used on one computer only.
- Application: feature mapping: Applications had many features
- Application to application integration: Applications where stand-alone and not connected with each other.
- Data storage location: Data were stored on the computer only, or on connected storage devices.
- User involvement type: Users interacted with applications. It means the use of an application is user controlled and user driven. It can be called “manual”.

We have seen several technology advances during the last 25 years. It includes the invention of the world wide web (1992) [5], the introduction of mass-market mobile devices and services (during the 1990s) [6, 7], broadband internet (ongoing) [8], and cloud services (around 2006) [9]. Such changes made it possible to allow users to evolve the way they use (mobile) computers and (mobile) applications:

- Context of use: Computers became mobile devices (incl. laptops) and are used everywhere.
- User – device mapping: One user uses several devices, partially at the same time.

– Application – device mapping: The same application runs on several devices, sharing the same data which are stored in the cloud.
– Application – feature mapping: Applications tend to be more specialized and have significantly fewer features (“mobile first” approach). It means we use more applications today.
– Application to application integration: Many applications are integrated with other applications and share data and status information.
– Data storage location: Data are often stored in the cloud, and some data are still stored locally.
– User involvement type: Users still interact with applications. However, many applications today run in the background and inform (interrupt) users about a status change or request inputs from users. This form of user involvement is called “intervention” and is useful for (semi-) automated processes [10].

2.2 User involvement metric taxonomy

As we can see, user involvement has evolved during the last 50 years. To be able to measure user involvement, the metrics need to adopt, too. That is the reason for a new user involvement metric taxonomy (see [1], including definitions).

The user involvement taxonomy as shown in Figure 1.

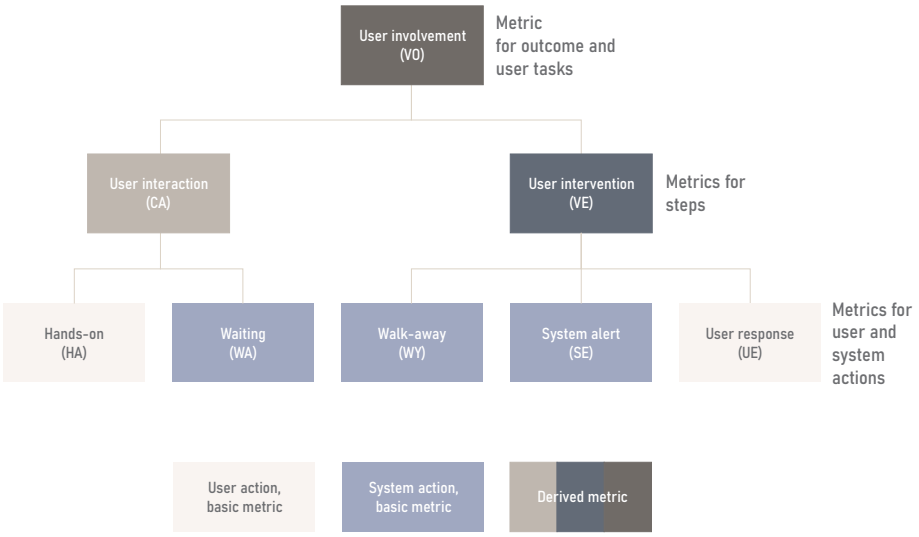

Figure 1. User involvement metric taxonomy [1]

Hands-on (HA) A user interacts with a system and drives the task completion forward. Hands-on can be measured as time and frequency.

Waiting (WA) A system processes the user’s input and updates the system output (which becomes input for the user). Waiting can be measured as time and frequency.

User Interaction (CA) User interaction is the sequence of hands-on and waiting. It can be measured as time and frequency.

Walk-away (WY) Walk-away expresses that a user can walk away from a system while the system performs a task without the need for a user to provide inputs. Walk-away is measured as time and frequency.

System alert (SE) System alert is the process of providing an alert to the user as a response to an incident. It is measured in time and frequency.

User response (UE) The user response is the process of a user to constructively respond to a system alert. The user response is measured as time and frequency.

User Intervention (VE) User intervention consists of walk-away, system alert, and user response. It can be measured in time and frequency.

User Involvement (VO) Consists of either interaction, intervention, or both. User involvement is a metric for user tasks and for the generation of an outcome of value, consisting of one or more user tasks. It is measured in time, frequency, and productivity.

3 Related Work

We looked at related work considering one of the following approaches: Summative Usability Test, Performance Testing, and Web Analytics. The introduced user involvement metrics are used as criteria to demonstrate that the proposed measurement technique is different and innovative, compared to the discussed related work (see Table 1).

Table 1. Criteria and Approaches

Approach / Criteria	CA	HA	WA	WY	SE	UE	VE
Summative Usability Test	Y	N	N	N	N	Y	N
Performance Testing	N	N	Y	Y	Y	N	N
Web Analytics	(Y)	(Y)	N	N	N	(Y)	(Y)

Summative usability test is a test for efficiency and measures time-on-task metric [1]. The time-on-task is the time taken by the user to complete a task. In other words, it can measure the user interaction time and user response time but not others.

Performance testing is the process of measuring a software system's performance metrics, e.g., response time, throughput, and resource utilization [11, 12]. The measurement for response time starts with an event triggering a system process and ends when a system function or system process is completed or at a certain state [11]. It does not consider user-initiated actions as a starting point that may or may not lead to an event triggering a system process [13]. Response time covers all system action metrics, including waiting time (WA), walk-away time (WY), and system alert time (SE).

Web analytics is the process of collecting and analyzing website data with the goal of optimizing website usage and experience [14, 15]. It is a continuous process of identifying the user behavior that is most profitable on the website [15]. User behavior and user-related data are tracked typically by enabling JavaScript on each webpage [15]. Web analytics does not track system initiated actions. Therefore, it cannot measure waiting (WA), walk-away (WY), and system alert (SE). Though web analytics can track user behavior within a website/application it cannot track the user behavior outside of the website since the technology relies on JavaScript page tagging [14]. For example, the opening or closing of a browser cannot be tracked by a website's JavaScript page tags. Hence, hands-on time (HA) and user response (UE) are partial yes (Y).

It is important to note that none of the approaches can be used to measure all the user involvement metrics mentioned in Figure 1. In a nutshell: None of them measures system actions and user actions.

4 Technique to measure user involvement

4.1 Research question [Parinitha]

The paper addresses the following research question:
RQ.1: How to measure selected user involvement metrics of a software application?

4.2 Measurement technique

Design details The measurement technique includes two main components: Event tracker and Log analyzer. The Event tracker monitors user and system actions for a given task and records the events to a log file. Log analyzer parses the event logs and computes the user involvement times for the task.

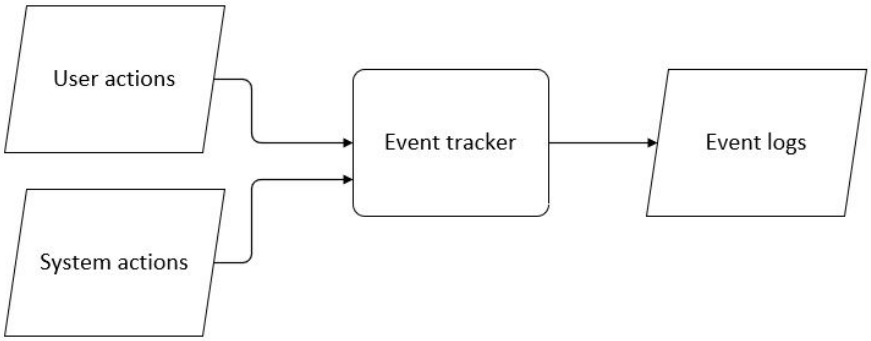

Figure 2. Event tracker

Event tracker User actions for a given task are performed using mouse and keyboard. Events generated by the mouse are "OnClick", "OnMove", etc. Events generated by

the keyboard are “OnPress”, “OnRelease”, etc. Event tracker listens to these events and records them in a event log file with timestamps.

System actions are responses of a system or an application to a user or a system action for a given task. Event tracker monitors the system actions by tracking the user interface changes in the application and by tracking the application process. These events are recorded by the event tracker in the same event log file with timestamps.

Log analyzer Log analyzer consumes event logs and event metrics mapping table as inputs to compute the user involvement measurements for a given task.

- User involvement as defined earlier consists of either interaction, intervention, or both. In this case, it is user interaction i.e. hands-on and waiting times. Both hands-on time and waiting time are intervals. They have a start time and an end time. These start and end times are event timestamps in the event logs. Hands-on time is the difference between the start time and end time of a user action. Waiting time is the difference between the start time and the end time of a system action.
- Event logs consist of all events (user and system) with timestamps recorded by the event tracker.
- Event metrics mapping table contains information about the start and end events for every user and system action (see Table 2).

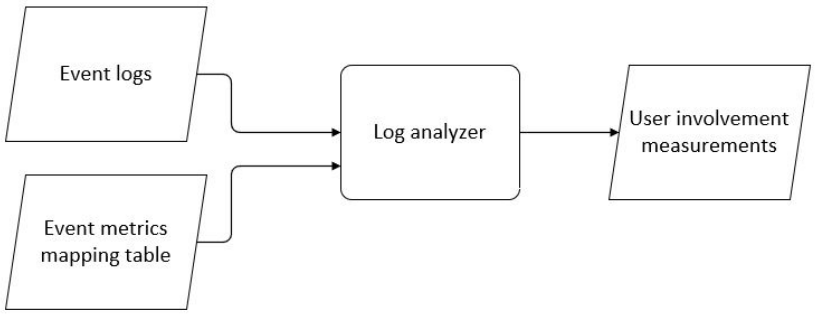

Figure 3. Log analyzer

Implementation details The measurement algorithm was implemented in Python. The steps in the algorithm for computing user involvement measurements (hands-on and waiting times) for a given task are as follows:

1. Create a log file to record user and system actions
2. Launch a thread to listen to mouse events (OnMove, OnClick)

3. Launch a thread to listen to keyboard events (OnPress, OnRelease)
4. Create a method to return the application process start time
5. Create an application driver to track the UI changes
6. All the threads continually write the user/system events with a timestamp to the event log file. The log analyzer takes event logs and event metric mappings as input to compute the hands-on time and waiting times (user involvement measurements).

Mouse and keyboard events Event tracker uses Pynput, a python library that provides methods to monitor input devices - mouse and keyboard [16].

Application process To retrieve the application process start time, psutil a python library is used [17].

User interface changes Appium together with WinAppDriver is used to monitor the UI of a Windows desktop application [18–20]. Appium is selenium based automation framework that interacts with the platform specific application drivers such as the WinAppDriver for Windows, XCUITest for iOS, etc. [21, 22].

Table 2. Event metrics mapping

Action	Start event	End event	Metric
User action	Mouse event "OnMove" or end event of a system action	Keyboard event "OnPress" or mouse event "OnClick"	Hands-on time
System action	End event of a user action	User interface display completion or application process start time	Waiting time

4.3 Experiment and results

The algorithm was applied to measure the user involvement for a Windows Calculator for a simple task of "addition". The process of the experiment is as shown in Figure 4.

Figure 4. Process of user involvement measurement for Calculator

Once the event tracker is started, the event tracker starts monitoring user and system actions. The mouse, keyboard, application process and changes in the calculator's user

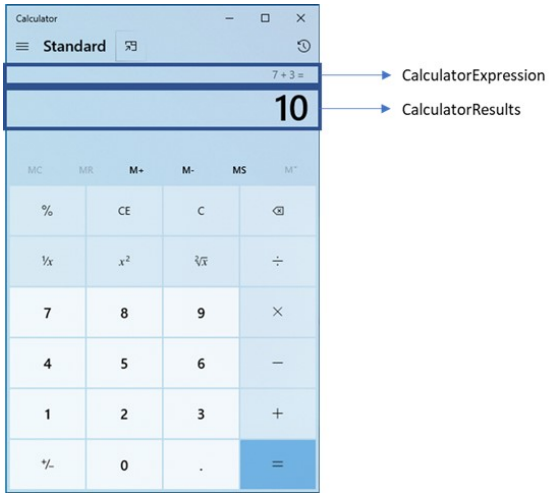

Figure 5. Calculator UI sections

interface are tracked. The two calculator UI sections that are monitored by the event tracker (CalculatorResults and CalculatorExpression) are as shown in Figure 5.

Figure 6 shows the start and end events for user and system actions (only the first 4 actions are shown). The event metric mapping table that is consumed as an input by the log analyzer to compute the user involvement measurements for the "addition" task is shown in Table 4 in appendix.

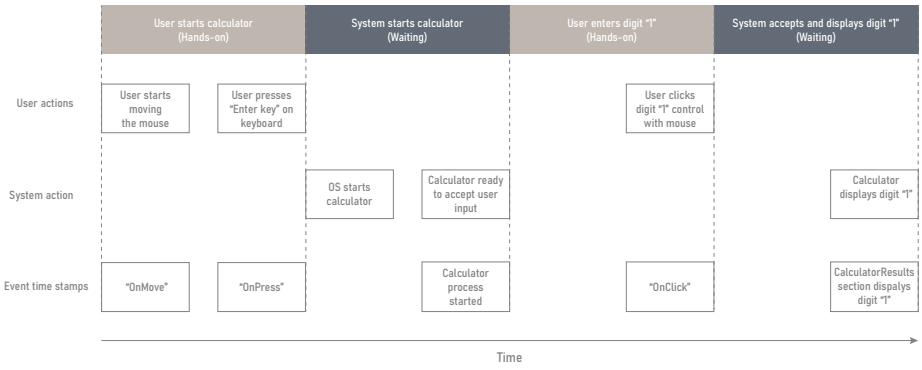

Figure 6. Start and end events for user and system actions

The user involvement measurement results for "addition" task are shown in Table 3.

The experiment was performed on a computer with the following specifications: HP laptop with an Intel(R) Core(TM) i7-10850H CPU @ 2.70GHz 2.71 GHz processor, 6 Cores(s), 12 logical processors, installed RAM of 32.0 GB (31.8 GB usable). The operating system was Windows 10 Enterprise, 64-bit operating system, x64-based processor, version 20H2, build 19042.1348 with the Windows feature experience pack 120.2212.3920.0. Windows Calculator version is 10.2103.8.0.

Table 3. Measurement of the user involvement time for the Windows Calculator

Action	Start time	End time	Measurement (mm:ss)	Metric
User starts calculator	23:38:43.534	23:38:45.827	00:02.293	Hands-on time (HAT)
System starts calculator	23:38:45.827	23:38:45.832	00:00.005	Waiting time (WAT)
User enters digit "1"	23:38:45.832	23:38:47.352	00:01.520	Hands-on time (HAT)
System accepts and displays digit "1"	23:38:47.352	23:38:47.470	00:00.118	Waiting time (WAT)
User enters operator "+"	23:38:47.470	23:38:48.055	00:00.585	Hands-on time (HAT)
System displays operator "+"	23:38:48.055	23:38:48.244	00:00.189	Waiting time (WAT)
User enters digit "2"	23:38:48.244	23:38:48.628	00:00.384	Hands-on time (HAT)
System accepts and displays digit "2"	23:38:48.628	23:38:48.844	00:00.216	Waiting time (WAT)
User enters operator "="	23:38:48.844	23:38:49.278	00:00.434	Hands-on time (HAT)
System displays operation result	23:38:49.278	23:38:49.361	00:00.083	Waiting time (WAT)
			00:05.216	Elapsed hands-on time (EHAT)
			00:00.611	Elapsed waiting time (EWAT)
Elapsed/Relative times			00:05.827	Elapsed interaction time (ECAT)
			89.5%	Relative hands-on time (REHAT)
			10.5%	Relative waiting time (REWAT)

5 Discussion and Future Work

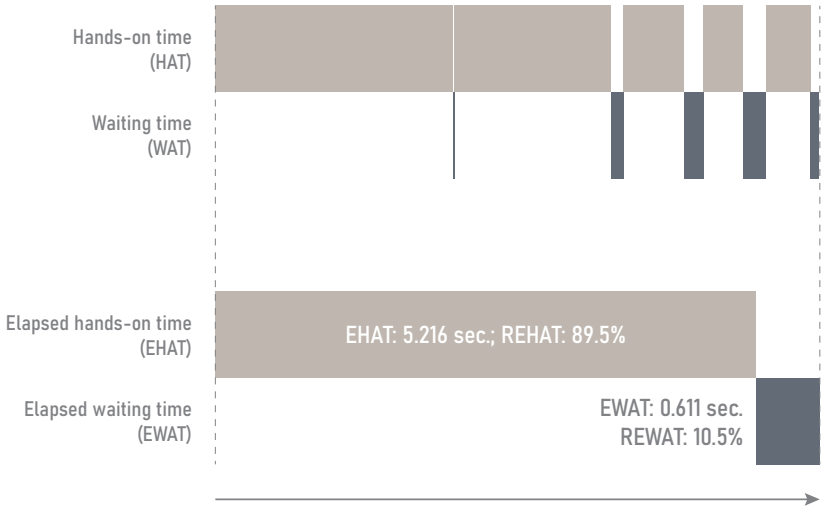

Figure 7. Calculator user involvement measurements

This paper proposes a technique to measure the efficiency of user involvement based on the user involvement metric taxonomy [1] (see Figure 1). Existing techniques such as summative usability tests, performance testing, and web analytics cannot measure elapsed user involvement including system actions and user actions. Measuring user and system actions allows to identify the root causes of inefficiencies that can be caused by system performance (measured with system actions) or by usability issues (measured with user actions).

The paper describes a measurement technique involving two main components: Event tracker - to track user and system actions, and Log analyzer - to compute the user involvement measurements from the captured logs. The Event tracker used to track user and system action uses libraries and tools such as Pynput, Psuitl, and Appium. Appium which is mostly used for functional testing is leveraged to track user interface changes in the current implementation [13]. It also helps to extend the implementation to applications on other platforms such as iOS and Android just by using platform specific drivers [21]. The technique proposed can be extended to webapps by using Selenium webdriver [23].

The measurement technique is applied to the Windows Calculator for a simple addition task and the results are as shown in Table 3. The results show that 5.216 seconds (89.5% of elapsed user involvement time) is allocated to elapsed hands-on time and 0.611 seconds (10.5% of elapsed user involvement time) for elapsed waiting time. The elapsed user involvement time is 5.827 seconds. These measurements show that the elapsed waiting time is significantly lower than the elapsed hands-on time (see Figure

7). The measured elapsed waiting time (0.611 seconds) is in line with the response time benchmark boundary of 0.5-1 seconds [2]. The average waiting time (0.122 seconds) is in line with the continuous benchmark boundary of 2-5 seconds which means that the user stays in the flow [2]. In general, the user involvement metrics provide context for the individual measured events and make them interpretable and usable.

The measurement technique has several limitations. (L1) It does not recognize user's actions if they are not executed with interaction devices, e.g., mouse movement, keystrokes, etc. (L2) The start of a user action is difficult to identify because it either starts with a cognitive process or with use of an interaction device which may or may not contribute to the task at hand. (L3) We could not find a way so far to determine automatically when a Windows application is ready for user involvement. Initially, the process start time was considered as an indication that the calculator is ready for use by the user. After some experiments with other native Windows applications, it was found that the application process appears in the task manager before the application is ready for user involvement or user inputs. (L4) Another limitation is the calculator example itself, which is small and high performant.

In order to measure user involvement times, user intervention measurements (both time and frequency) are also needed. A technique to measure user intervention times is a fruitful topic for future research. Another possible extension is the automatic detection of deviations from user involvement benchmarks.

References

1. Degen, H.: Efficiency of user involvement: an underrated opportunity for value creation. In: Proceedings of 17th International Conference on Wirtschaftsinformatik February 2022, Nürnberg, Germany (2022)
2. Seow, S.C.: Designing and Engineering Time: The Psychology of Time Perception in Software. Addison-Wesley Professional, 1 edn. (2008), https://www.pearson.com/us/higher-education/program/Seow-Designing-and-Engineering-Time-The-Psychology-of-Time-Perception-in-Software/PGM166241.html
3. Miller, R.B.: Response Time in Man-Computer Conversational Transactions. In: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I. pp. 267—277. AFIPS '68 (Fall, part I), Association for Computing Machinery, New York, NY, USA (1968), https://doi.org/10.1145/1476589.1476628
4. Apple: Apple Introduces Macintosh Advanced Personal Computer. https://web.stanford.edu/dept/SUL/sites/mac/primary/docs/pr1.html (Jan 1984), accessed 18. Dec. 2021
5. Berners-Lee, T.: World Wide Web. http://info.cern.ch/hypertext/WWW/TheProject.html (Dec 1990), accessed 18. Dec. 2021
6. Farley, T.: Mobile telephone history. Telektronikk 101(3/4), 22–34 (Feb 2005), https://www.telenor.com/wp-content/uploads/2012/05/T05_3-4.pdf
7. ETSI: 5G. Standard, European Telecommunications Standards Institute, Sophia-Antipolis, France, https://www.etsi.org/technologies/5G
8. A History of Speed as the World Wide Web Turns 25. https://www.ncta.com/whats-new/a-history-of-speed-as-the-internet-turns-25 (Aug 2016), accessed 16. Jan. 2022
9. Arutyunov, V.V.: Cloud computing: Its history of development, modern state, and future considerations. Scientific and Technical Information Processing 39, 173–178 (Jul 2012), https://doi.org/10.3103/S0147688212030082

10. Schmidt, A., Herrmann, T.: Intervention user interfaces: A new interaction paradigm for automated systems. Interactions 24(5), 40–45 (aug 2017), https://doi.org/10.1145/3121357
11. Gorton, I.: Essential Software Architecture (2. ed.). (01 2011)
12. Jiang, Z.M., Hassan, A.E.: A survey on load testing of large-scale software systems. IEEE Transactions on Software Engineering 41(11), 1091–1118 (2015)
13. Quental, N.C., de Albuquerque Siebra, C., Quintino, J.P., Florentin, F., da Silva, F.Q.B., de Medeiros Santos, A.L.: Automating gui response time measurements in mobile and web applications. In: Proceedings of the 14th International Workshop on Automation of Software Test. p. 35–41. AST '19, IEEE Press (2019), https://doi.org/10.1109/AST.2019.00011
14. Bekavac, I., Garbin Praničević, D.: Web analytics tools and web metrics tools: An overview and comparative analysis. Croatian Operational Research Review 6, 373–386 (10 2015)
15. Kumar, V., Ogunmola, G.: Web analytics for knowledge creation: A systematic review of tools, techniques, and practices. International Journal of Cyber Behavior, Psychology and Learning 10, 1–14 (12 2019)
16. pynput. https://pypi.org/project/pynput/1.7.5/ (Nov.), accessed 15. Dec. 2021
17. psutil documentation. https://psutil.readthedocs.io/en/latest/, accessed 10. Dec. 2021
18. Appium. http://appium.io/, accessed 10. Dec. 2021
19. Windows Application Driver. https://github.com/Microsoft/WinAppDriver (Jan 2020), accessed 15. Jan. 2021
20. You can use WinAppDriver by itself or with Appium. https://github.com/microsoft/WinAppDriver/blob/master/Docs/UsingAppium.md, accessed 15. Dec. 2021
21. Introduction to Appium. https://appium.io/docs/en/about-appium/intro/, accessed 10. Dec. 2021
22. Microsoft, T.C.A.: WinAppDriver and Desktop UI Test Automation. https://techcommunity.microsoft.com/t5/testingspot-blog/winappdriver-and-desktop-ui-test-automation/ba-p/1124543 (Jan 2020), accessed 15. Jan. 2022
23. Selenium. https://www.selenium.dev/, accessed 15. Dec. 2021

6 Appendix

Table 4. Calculator event metrics mapping table

Action	Start event	End event	Metric
User starts calculator	User starts moving the mouse Event: OnMove	User presses "Enter key" on keyboard Event: OnPress of key ‘enter’	Hands-on time
System starts calculator	See previous end event of user action	Calculator ready to accept user’s input Event: Calculator process started	Waiting time
User enters digit "1"	See previous end event of system action	User clicks digit "1" control with mouse Event: OnClick of digit "1"	Hands-on time
System displays digit "1"	See previous end event of user action	Calculator displays digit "1" Event: CalculatorResults section displays digit "1"	Waiting time
User enters operator "+"	See previous end event of system action	User clicks operator "+" with mouse Event: OnClick of operator "+"	Hands-on time
System displays operator "+"	See previous end event of user action	Calculator displays operator "+" Event: CalculatorExpression section displays expression	Waiting time
User enters digit "2"	See previous end event of system action	User clicks digit "2" control with mouse Event: Onclick of digit "2"	Hands-on time
System displays digit "2"	See previous end event of user action	Calculator displays digit "2" Event: CalculatorResults section displays digit "2"	Waiting time
User enters operator "="	See previous end event of system action	User clicks "=" operator with mouse Event: OnClick of operator "="	Hands-on time
System displays operation result	See previous end event of user action	Calculator displays operation result Event: CalculatorResults section displays result	Waiting time