Intelligent Interaction Techniques (IIxT) – Proposal

Brad A. Myers
Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
bam@cs.cmu.edu

Abstract

Interaction techniques (IxTs) are the low-level, reusable components out of which user interfaces are designed, including menus, scroll bars, text input fields, and also copy-paste, text-entry, and selecting objects. The IxTs for graphical user interfaces (GUIs) were well established in the 1980s, with relatively minor additions and tweaks for smartphones in the 2000s. Most of today's AI user interfaces involve a chat window, which is an excellent interaction for some tasks, but is generally considered separate from the GUI IxTs. I argue for making the IxTs *themselves* more intelligent, so users can freely mix modalities, even within the same interaction. This will require research into new IxTs, and also into the infrastructure that will enable these intelligent IxTs (IIxTs) to be built. There are also significant security, privacy and economic implications to this vision.

Author Keywords

Interaction Techniques (IxT); Intelligent Interaction Technique (IIxT), Widgets; Intelligent User Interfaces; Smart widgets.

1 Introduction

An interaction technique (IxT) is formally defined as starting when the user performs an action that causes an product to respond, and includes the direct feedback from the product to the user [20, section 1.3.8]. Interaction techniques are generally reusable across various applications. Examples of interaction techniques include on-screen widgets like menus, scrollbars and buttons, but also the low-level ways of interacting with on-screen objects, such as swiping gestures to scroll, copy-and-paste, drag-and-drop, text entry and editing, graphical

selection and editing, etc. An important distinction is that *applications*, like Microsoft Word, GMail, Facebook, or Yelp, are *composed* out of (many) interaction techniques, and are not themselves considered an IxT. Most of the interaction techniques used in regular user interfaces, often called graphical user interfaces (GUIs), were developed in the 1970s and 1980s, and were updated slightly with smartphones in the 2000s. My book [20] has a comprehensive discussion of this topic.

Interaction techniques are often collected in a library to make them easier for others to reuse. For example, low-level widgets are often provided by a *toolkit* code library, and support for other IxTs like copy-paste and undo are provided by *frameworks* [17].

What counts as making an IxT *intelligent*? This is defined (somewhat vaguely) as an interaction technique that uses Artificial Intelligence (AI) to make it work [20, section 18.2]. One defining feature of all intelligent user interfaces is that previously deterministic IxTs will become probabilistic and therefore may be incorrect.

Intelligent user interface (IUI) or AI interaction research is often concerned with the *whole* user interface and how it behaves, usually composed of conventional IxTs. For example, today's successful AI applications, such as Claude, Firefly in Adobe products, CoPilot in Microsoft products, Figma Design Agent & Figma Make, etc., are mostly chat interfaces (which use a conventional text entry IxT) *along side* conventional GUIs, rather than having the AI *integrated* into the interaction techniques themselves.

There has always been at least some research on intelligent interaction techniques (IIxTs), often published at UIST and CHI, which has been inspirational (e.g., [5, 25, 28, 29, 31]). However, commercial IxTs have used older AI techniques which were not particularly successful, such as the Microsoft Office XP "personalized menus" from 2001 which tried to be "smart" about which items to display.

I argue that with today's much better AI, including Large Language Models (LLMs), Multimodal LLMs (MLLMs), Generative AI (GenAI), etc., it is time to revisit how to create reusable, composable low-level interaction techniques that are intelligent and useful, especially for graphical user interfaces.

2 Why GUIs are Still Relevant

Some have argued that we will not use conventional graphical user interfaces, desktops or smartphone screens or their interaction techniques once natural language and speech interfaces get better. I disagree. There are definitely some tasks that are easier when performed using natural language, but there are others than are not. Many other writers have also made this argument (e.g., [11, 24]), so I will just list a few points here. In later sections, I discuss how to augment IxTs to preserve their advantages while making them more intelligent.

Ben Shneiderman's original list of properties elucidates situations when *direct manipulation* interfaces are important and useful: when it is feasible to have "**continuous representation of the object of interest**" with "**rapid, incremental, reversible operations whose**

impact on the object of interest is immediately visible" [23]. When GUIs were being invented in the 1970s and 1980s, and similarly at the beginning of the smartphone era in the 2000s, users had a small number of files and applications and their icons, so arranging or listing them on the screen was feasible. Today, even when we have hundreds to millions of items on our devices, there are times when it is still faster to list and tap on items, and other times when an intelligent search is preferred or necessary. For example, I have about 30 apps on the front screens of my smartphone that I regularly want to quickly tap on, but for most of the others, I switch modalities and type the name to search. I strongly prefer to find files by location in the folder tree, whereas I almost only find email messages by using textual search. And when we find the objects of interest, we usually want to see them in a list, so we can perform operations easily and quickly – I want to hit the "Delete" keyboard key or the trashcan icon for most of my emails.

Another advantage of GUIs is that sometimes users do not know what can be done or do not know what things are called, so expecting them to express it, even in natural language, is problematic, compared to being able to browse a menu of options. The menu also helps teach users the correct vocabulary for the current task.

For understanding data, the visualization field [8] has shown that often understanding only comes from providing a visual representation that the user can view and interact with, rather than trying to describe the data in words.

Similarly, users may prefer to visually browse to find the items of interest if the distinguishing features are visual, rather than having to describe them using natural language. Producing (speaking or typing) and consuming (hearing or reading) natural language is still a linear, sequential process, whereas visual search and understanding is parallel, and can sometimes be performed in the background when the user's cognition is focused on a foreground task.

3 Historical Examples of Intelligent IxTs

As mentioned in the introduction, there is too much research on intelligent user interfaces in general to summarize here (see the IUI conference series, CHI, UIST and a few summaries [20, chapter 18] [11, 27]). There are even too many intelligent IxTs to summarize here, but I present a few examples to help clarify what I mean by reusable interaction techniques which are intelligent.

The Put-That-There system [6] from the 1970s demonstrated an important early example of how speech recognition could enhance pointing, including when there were *multiple* references in the same utterance (hence, "put *that there*." Some IUI systems today allow the AI to access the selected object(s), but I don't know of any that support *temporal* information in the selections and speech recognition to be able to support knowing which object goes with which part of the utterance.

The use of AI in text editing techniques dates back to at least 1966 with Warren Teitelman's *Do What I Mean (DWIM)* spelling-correction system [26]. The auto-correct, auto-complete and auto-fill capabilities of text editing have continually improved over time [20, section 8.4], and now LLMs enable much longer completions (up to the whole document or code!), and

more sophisticated editing [31], and are available in many places where users need to enter text.

A different kind of IIxT is highlighted by Adobe Photoshop and related products, where selection and operations on parts of pictures are augmented by image recognition. For example, one can select and extract just a face or a person from a photograph, intelligently differentiating their hair and body from the background. I classify this as an IxT since it can be available across any application that requires image selection or editing.

Finally, the areas of virtual reality (VR), augmented reality (AR), and mixed reality (MR), collectively called extended reality (XR), have long needed new interaction techniques. The user still needs to identify and operate on the objects in the 3D scenes, but conventional 2D GUI widgets do not work well. AI in the form of image, gesture and speech recognition, is often used in these environments to improve the accuracy and success of the UIs.

4 Proposed Examples of IIxTs

Virtually all interaction techniques [20] could potentially be improved by being more intelligent. While some areas (like text entry) have received a lot of attention, others have not yet been investigated. Rather than trying to be comprehensive, this section presents some examples.

4.1 Intelligent Referencing

Indicating the object of interest, also called **pointing** [20, chapter 4] or **selecting** [20, sections 8.6.3 and 10.5], has always been fundamental to user interfaces for all technology. Early research showed that the "mouse" pointing device [10] was faster and more accurate than alternatives [7], but current systems use less accurate pointing devices such as finger touches. Furthermore, users are now tasked with selecting among significantly more objects (e.g., among tens of thousands of email messages or photographs) and of more complex objects (to identify which part of the 3D avatar should be selected). There are many ways that this could be more intelligent.

Imagine if the user could simultaneously *describe* the desired selection while pointing. For example, performing a low-resolution indication of an area, and expressing the details in natural language ("at the left of that character", "right-most point of that curve"). Besides just natural language clarifying the graphical properties, with today's AI, we can provide real "semantic snapping" [12] that takes into account the *meaning* of the desired object ("the iris of the left eye", "just the name of the restaurant", "the red cars"). There could be easier ways to have *multiple* selections ("add all of these", "select all the phone numbers in this paragraph"). These examples also suggest how users might *fix* the selection set through a combination of direct manipulation (shift-touch to toggle items) and natural language ("undo the last one"). This might involve more intelligent checking for outliers to help users avoid selecting undesired objects [16]. The selections could be saved, to help users reestablish their

context ("take me to where I left off yesterday")[1].

The pointing infrastructure and the natural language both need to do a better job of recording the temporal aspects, so users can refer to multiple items in a single utterance ("put that there"), but also so they can refer to past actions ("deselect the first one").

Finally, we need better ways to *refer* to items without necessarily operating on them, so users can tell the AI *about* some items. Sometimes, systems must have different modes, like the Run versus Test modes in Interactive Builder programs [17] like Microsoft Visual Basic or programming-by-example systems [21]. We developed a "wiggling" interaction technique as one way to reference objects at a location without triggering the hovering or clicking actions on them [15].

Searching using text has always been an important way for users to find the items of interest. This has become the predominant method for many people to find files, email messages, and apps on smartphones, as the number of items has gotten too large to memorize or search visually. Intelligent searching using text is already becoming available, but this can become even more intelligent by enabling searching with other properties beyond the text of objects, such as the semantic properties mentioned above. Graphical search was researched by a few systems [14, 19], but was too hard to provide, which AI may make easier. Similarly, *associative search* [9], where users can find items using meta properties like their creation time, location or other items they are related to, can also be provided.

4.2 Other Intelligent Widgets

Conventional widgets could also be made smarter. Smart **menus** were unsuccessful in Windows XP in 2001, in part because they did a bad job of predicting what the user might want, whereas today, such approaches can be much more successful. More interesting is integrating menus with semantic search, so users can express what they want, and the system would show users the menu options, keyboard shortcuts and icons for that, so users learn the faster methods to perform that action in the future. Intelligent **customization** of the menus and other commands could make it much easier for users to personalize their UIs. For example, previous versions of Microsoft Toolbars provided extensive customization, but this was removed with the introduction of the ribbon in Office 2007 [20, section 9.5], because users messed it up too easily — an intelligent mechanism might improve both the personalization and usability, so users could issue commands more efficiently.

Research has shown that for certain tasks, users waste a lot of time **scrolling** to find what they want to look at, and especially scrolling *back* to where they were [13]. An intelligent scrolling mechanism would combine scrolling manually with jumping to specific places in the document (like based on a table of contents), setting bookmarks in the documents, and with a mechanism for returning (undoing the scroll) [1, 19]. It could also help coordinate scrolling of *multiple* documents, for example, for comparing the related content across web pages or documents. Combining intelligent search and scrolling mechanisms might result in a generalized **intelligent navigation** mechanism.

[1]Thanks to Haiyi Zhu for contributing this idea.

Providing **undo** of interaction techniques could be generalized beyond scrolling – I often want to undo moving or closing a window, undo a change to the selection, or of other actions which are currently not undoable. We have also lost the ability to undo many operations in web and mobile applications that we used to be able to undo on desktops, which should be restored intelligently. Techniques for **selective undo** have been demonstrated by research and a (very) few commercial systems, where the user can specify to undo only *some* of the previous actions (e.g., "Undo just the formatting changes, but keep the text edits", "Undo just the changes to this paragraph, but keep the changes elsewhere that I made afterwards") [3, 19, 22, 30], but they have always been hampered by clunky UIs. An intelligent mechanism would allow users to simply express which operations to keep and undo.

Having an intelligent agent always available (like Apple Siri, Google Ask Gemini, or Microsoft Copilot), but which was aware of the interaction techniques and operations, could provide ubiquitous undo and **help** everywhere, and would provide a locus where there might be an intelligent **slider** for controlling how much autonomy is desired.[2]

4.3 Intelligent Operations Across Applications

Today's cross-application interaction techniques are limited to copy-and-paste and drag-and-drop, which could be made more intelligent, and others could be added.

Intelligent selection could be the first step of intelligent **copy-and-paste**. In 2004, we investigated having the paste operation take into account the structure of the data on the clipboard, and allow it to be pasted into multiple places, like putting the parts of an address into different web form fields [25], but with today's AI, many transformations of the data could be supported, including allowing users to express what is desired. For example, "paste as bullet points", "paste as bibtex citations", or "paste translated into English"[3]: These could also be extended to support **drag-and-drop** as well.

Other operations **across applications** should be supported. Agents today have some limited abilities to remote control and collect information from some applications on the web, and to a lesser extent on the desktop and smartphones, but this should be expanded. One can easily say in natural language "Group my flight, hotel and restaurant reservations together," but this is still difficult to perform if they are from independent applications.

4.4 Intelligent Accessibility

The hooks that would allow intelligent interaction techniques would be a boon for accessibility. Most low-level IxTs on major platforms today have lots of accessibility hooks and settings, but not typically IxTs for the web (like menus on web pages). Providing a natural-language interface that is available at a low-level everywhere would also be a tremendous advantage for vision-impaired and physically disabled individuals who can type or provide speech input.

[2]Thanks to Dan Saffer for contributing this idea.

[3]Thanks to Motahhare Eslami for some of these examples.

5 Architectural considerations for IIxTs

Most interaction techniques of the types discussed here are provided in libraries and frameworks (with the exception of many kinds of menus in web and smartphone applications). Therefore, some of the proposed ideas could be realized entirely within those layers. However, most of the interesting forms of intelligence discussed above require knowledge about the applications themselves, and of the user. This imposes significant barriers from **architectural, security and privacy** perspectives. For example, I have been told that the reason that the text entry keyboard on Apple smartphones does not provide better proposed completions is that it is *not* allowed to know much about the application for privacy reasons.

The IIxTs described here would likely be developed by the platform vendors who now provide the toolkits (Microsoft, Apple, Google). This means that they will be able to devote significant resources to their development and testing. However, IIxTs have even higher requirements for security and privacy than intelligent apps, since users will typically not be able to opt out of using them. Further, this might mean even more **consolidation and power** for the platform vendors.

To provide the proposed cross-application intelligent interactions discussed above, the agent will need access to information from those applications, which today is not available. APIs or screen scraping can provide to Siri or Claude some limited information and control of the application, but full access would require a new software architecture for applications besides the siloed apps of today. Researchers have called for this for many years (see my "Open Data Model" from 1998 [18], the "Semantic Web" from 2001 [4], and "A World Without Apps" from 2019 [2]), and it is needed for many forms of end-user programming (EUP) and programming by example (PBE) [21]. However, it has always proven difficult to provide due to business reasons (companies want to keep their own data proprietary) and architectural challenges (it is very difficult to provide sufficient access without being overwhelming).

However, if an intelligent agent is going to be built at all, it will need significant access to this kind of data, so the privacy, security and architectural challenges will need to be addressed anyway.

6 Conclusions

Providing interactions in low-level toolkits and frameworks has always been the best way to make it easy (and even possible) for developers to include them in their applications. This will also be true for intelligent interaction techniques. Users will then have access to much better usability, effectiveness and accessibility within and across their applications. Another advantage of providing intelligence at the interaction technique level is that it will then behave *consistently* across applications and be ubiquitously available by default. However, significant challenges remain, in all of the design, architecture, security, privacy, economic and business aspects.

Acknowledgements

Some of the ideas in this paper were contributed by Motahhare Eslami, Dan Saffer, and Haiyi Zhu. Thanks also for contributions to this paper from Scott Hudson, Niki Kittur, Nik Martelaro, Bernita Myers, and John Zimmerman.

References

[1] Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. Dwell-and-spring: Undo for direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, pages 1957–1966, Austin, TX, 2012. doi: 10.1145/2207676.2208339. URL https://dl.acm.org/doi/10.1145/3706598.3713554.

[2] Michel Beaudouin-Lafon. Uist 2019 visions – a world without apps. Talk at UIST'2019: 32nd Annual ACM Symposium on User Interface Software and Technology, Oct. 20–23, 2019, New Orleans, LA, 2019. URL https://www.youtube.com/watch?v=ntaudUum06E.

[3] Thomas Berlage. A selective undo mechanism for graphical user interfaces based on command objects. *ACM Transactions on Computer-Human Interaction*, 1(3):269–294, 1994. doi: 10.1145/196699.196721. URL https://dl.acm.org/doi/10.1145/3706598.3713554.

[4] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001. doi: 10.1038/scientificamerican0501-34. URL https://doi.org/10.1038/scientificamerican0501-34.

[5] Eric Allan Bier and Maureen C. Stone. Snap-dragging. In *Computer Graphics (Proceedings of SIGGRAPH '86)*, volume 20, pages 233–240, Dallas, TX, 1986. URL https://dl.acm.org/doi/10.1145/15922.15912.

[6] Richard A. Bolt. "Put-That-There": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80)*, pages 262–270, July 1980. doi: 10.1145/800250.807503. URL https://dl.acm.org/doi/10.1145/965105.807503.

[7] Stuart K. Card, William K. English, and Betty J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics*, 21(8):601–613, 1978. doi: 10.1080/00140137808931762. URL https://www.tandfonline.com/doi/abs/10.1080/00140137808931762.

[8] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, 1999. URL https://shop.elsevier.com/books/readings-in-information-visualization/card/978-0-08-051573-1.

[9] Duen Horng Chau, Brad Myers, and Andrew Faulring. What to do when search fails: Finding information by association. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*, pages 999–1008, Florence, Italy, 2008. doi: 10.1145/1357054.1357208. URL https://dl.acm.org/doi/10.1145/1357054.1357208.

[10] W. K. English, D. C. Engelbart, and M. L. Berman. Display selection techniques for text manipulation. *IEEE Transactions on Human Factors in Electronics*, HFE-8(1), 1967. doi: 10.1109/THFE.1967.232994. URL https://ieeexplore.ieee.org/document/1698228.

[11] Jie Gao, Simret Araya Gebreegziabher, Kenny Tsu Wei Choo, Toby Jia-Jun Li, Simon Tangi Perrault, and Thomas W. Malone. A taxonomy for human-LLM interaction modes: An initial exploration. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems (CHI EA '24)*, Honolulu, HI, USA, 2024. doi: 10.1145/3613905.3650786. URL https://dl.acm.org/doi/10.1145/3613905.3650786.

[12] Scott E. Hudson. Adaptive semantic snapping—a technique for semantic feedback at the lexical level. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 65–70, Seattle, WA, USA, 1990. doi: 10.1145/97243.97253. URL https://dl.acm.org/doi/10.1145/97243.97253.

[13] Andrew J. Ko, Brad A. Myers, Michael Coblenz, and Htet Htet Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering*, 33(12):971–987, 2006. URL https://ieeexplore.ieee.org/document/4016573.

[14] David Kurlander and Eric A. Bier. Graphical search and replace. In *Computer Graphics (Proceedings of SIGGRAPH '88)*, volume 22, pages 113–120, Atlanta, GA, 1988. doi: 10.1145/378456.378495. URL https://dl.acm.org/doi/10.1145/378456.378495.

[15] Michael Xieyang Liu, Andrew Kuznetsov, Yongsung Kim, Joseph Chee Chang, Aniket Kittur, and Brad A. Myers. Wigglite: Low-cost information collection and triage. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22)*, Bend, OR, USA, 2022. doi: 10.1145/3526113.3545661. URL https://dl.acm.org/doi/10.1145/3526113.3545661.

[16] Robert C. Miller and Brad A. Myers. Multiple selections in a smart text editor. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces (IUI '02)*, pages 103–110, San Francisco, CA, 2002. URL https://dl.acm.org/doi/10.1145/502716.502734.

[17] Brad A. Myers. User interface software tools. *ACM Transactions on Computer-Human Interaction*, 2(1):64–103, 1995. doi: 10.1145/200968.200971. URL https://dl.acm.org/doi/10.1145/200968.200971.

[18] Brad A. Myers. The case for an open data model. Technical Report CMU-CS-98-153, Carnegie Mellon University, School of Computer Science, August 1998. URL http://reports-archive.adm.cs.cmu.edu/anon/1998/CMU-CS-98-153.pdf.

[19] Brad A. Myers. Scripting graphical applications by demonstration. In *Proceedings of SIGCHI'98: Human Factors in Computing Systems*, pages 534–541, Los Angeles, CA, 1998. doi: 10.1145/274644.274716. URL https://dl.acm.org/doi/10.1145/274644.274716.

[20] Brad A. Myers. *Pick, Click, Flick! The Story of Interaction Techniques.* ACM Books,

New York, NY, 2024. doi: 10.1145/3617448. URL https://www.cs.cmu.edu/~bam/ixtbook/index.html.

[21] Brad A. Myers, Richard G. McDaniel, and David S. Kosbie. Marquise: Creating complete user interfaces by demonstration. In *Proceedings of the INTERCHI '93 Conference on Human Factors in Computing Systems*, pages 293–300, Amsterdam, The Netherlands, 1993. doi: 10.5555/164632.164928. URL https://dl.acm.org/doi/10.1145/169059.169225.

[22] Brad A. Myers, Ashley Lai, Tam Minh Le, YoungSeok Yoon, Andrew Faulring, and Joel Brandt. Selective undo support for painting applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*, pages 4227–4236, Seoul, Korea, April 2015. doi: 10.1145/2702123.2702543. URL https://dl.acm.org/doi/10.1145/2702123.2702543.

[23] Ben Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, 16(8):57–69, 1983. doi: 10.1109/MC.1983.1654471. URL https://ieeexplore.ieee.org/document/1654471.

[24] Ben Shneiderman and Pattie Maes. Direct manipulation vs. interface agents. *interactions*, 4(6):42–61, 1997. doi: 10.1145/267505.267514. URL https://dl.acm.org/doi/10.1145/267505.267514.

[25] Jeffrey Stylos, Brad A. Myers, and Andrew Faulring. Citrine: Providing intelligent copy-and-paste. pages 185–188, Santa Fe, NM, 2004. doi: 10.1145/1029632.1029665. URL https://dl.acm.org/doi/10.1145/1029632.1029665.

[26] Warren Teitelman. *Pilot: A Step Toward Man-Computer Symbiosis*. PhD thesis, Massachusetts Institute of Technology, 1966. URL https://en.wikipedia.org/wiki/DWIM.

[27] Sarah Theres Völkel, Christina Schneegass, Malin Eiband, and Daniel Buschek. What is "Intelligent" in intelligent user interfaces? A meta-analysis of 25 years of IUI. In *Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI '20)*, pages 477–487, Cagliari, Italy, 2020. doi: 10.1145/3377325.3377500. URL https://dl.acm.org/doi/10.1145/3377325.3377500.

[28] Jacob O. Wobbrock, Brad A. Myers, and John Kembel. Edgewrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST '03)*, pages 61–70, Vancouver, British Columbia, Canada, 2003. doi: 10.1145/964696.964703. URL https://dl.acm.org/doi/10.1145/964696.964703.

[29] Jackie Junrui Yang, Monica S. Lam, and James A. Landay. Dothishere: Multimodal interaction to improve cross-application tasks on mobile devices. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST '20)*, pages 35–44, October 2020. doi: 10.1145/3379337.3415841. URL https://dl.acm.org/doi/10.1145/3379337.3415841.

[30] YoungSeok Yoon and Brad A. Myers. Supporting selective undo in a code editor. In *Proceedings of the 37th International Conference on Software Engineering (ICSE 2015)*,

pages 223–233, Florence, Italy, May 2015. doi: 10.5555/2818754.2818784. URL https://ieeexplore.ieee.org/document/7194576.

[31] Tim Zindulka, Jannek Maximilian Sekowski, Florian Lehmann, and Daniel Buschek. Exploring mobile touch interaction with large language models. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025. doi: 10.1145/3706598.3713554. URL https://dl.acm.org/doi/10.1145/3706598.3713554.