The Medley Interlisp Project: Reviving a Historical Software System

Eleanor Young*, Larry Masinter†, Herb Jellinek†, Eric Kaltman*, Abhik Hasnain*
*Media and Technology Studies
University of Alberta, Edmonton, Alberta
†Interlisp.org
InterlispOrg Inc., Los Altos, California, United States
esyoung1@ualberta.ca, lmm@interlisp.org, jellinek@interlisp.org, kaltman@ualberta.ca, abhik1@ualberta.ca

***Abstract*—The Medley Interlisp project is an endeavor in software recovery, preservation, and development. It aims to render Medley Interlisp, the final release of Interlisp, usable on modern operating systems and hardware environments, and to selectively add modern capabilities to Medley while conserving its value as a historically groundbreaking system. This paper describes the Medley Interlisp project's past and ongoing operations, decision-making, and the unique issues encountered in the process of restoring a historic software system to modern computing environments. The Medley Interlisp project's five years of hardships, successes, and discoveries regarding the unique issues of software recovery will help to create a blueprint from which similar long-term historical software recovery projects may benefit.**

***Index Terms*—software recovery, Interlisp, early programming, programming languages and software, non-profit organization.**

I. Introduction

The Medley Interlisp project was founded in 2020 by several original developers of Interlisp and is currently registered under the name InterlispOrg Inc. as a US 501(c)(3) non-profit organization. The project aims to revive Interlisp, a software development environment that was developed primarily at the Xerox Palo Alto Research Center (PARC) throughout the 1970s and 1980s. Interlisp is an advanced implementation of LISP, an early programming language that stood alongside Fortran, COBOL, and ALGOL in the 1950s and 60s.

To the authors' knowledge, there is no prior work outlining a concerted software recovery effort aimed at a specific historical software system. This paper aims to address that gap by describing the Medley Interlisp project's five years of hardships, successes, and discoveries along its unique journey of software recovery in the hopes that it will create a blueprint from which similar long-term historical software recovery projects may benefit.

The following main goals have guided the project's efforts:

1. Make Medley Interlisp easy to run on all common modern operating systems,
2. Expand Medley Interlisp's compatibility with modern computer peripherals such as modern computer mice, keyboards, and displays,
3. Implement structural changes that ease interoperability between Medley Interlisp and modern operating systems (e.g., allow Medley to use or at least translate to Unicode encoding standards instead of using Xerox Character Code Standard (XCCS), access to the system clipboard for easy copy-pasting, etc.),
4. Continue development of Medley in the form of fixing bugs, updating Common Lisp compatibility to the ANSI Common Lisp standard, and completing half-finished features that remained in the codebase when development was originally halted,
5. Compile a comprehensive bibliography of documents pertaining to Interlisp and render it useful to researchers,
6. And, most importantly, conserve Medley Interlisp's value as a groundbreaking historical system for historical and computational research.

All of these goals are subordinate to the final one, as without fulfilling that goal, the entire project's purpose falls flat. The Medley Interlisp project does not aim to simply build an artifact, but to record and preserve the history and context that surrounds that artifact by restoring it to a state of modern accessibility.

II. Medley Interlisp — History and Overview

As mentioned above, Interlisp was developed at the Xerox Palo Alto Research Center (PARC) throughout the 1970s and 1980s. Throughout its lifespan, Interlisp supported software research in AI, computational linguistics, graphical user interfaces, hypertext, and other early computing fields, emphasizing capabilities that facilitated a research-focused development cycle rather than programming to specifications. Its affinity for rapid prototyping is something that other software development environments lack, even today.

To explain where this unique affinity comes from, we must dive deeper into Interlisp's history[1] with a brief overview beginning in 1958. Implementation of the programming language LISP had just begun, led by John McCarthy. From the start, McCarthy's goals were to create an algebraic list-processing language for artificial intelligence research, and throughout its development created a way to express LISP programs as data types within LISP, meaning that programs could be

[1]Much of the history as told in this paper, especially those parts without other sources indicated, is sourced from conversations with Larry Masinter, Herb Jellinek, Ronald Kaplan, and other members of the Medley Interlisp project team who were Interlisp developers at Xerox PARC throughout the 70s and 80s.

written that process and write other programs [1]. Later, LISP 1.5 was implemented on the SDS 940 at Bolt, Beranek, and Newman, Inc (BBN) and was called BBN-Lisp. In 1970, programmers Warren Teitelman, Daniel Bobrow, Alice Hartley, Peter Deutch, and others contributed key features to BBN-Lisp that would later become hallmarks of its next iteration, Interlisp. Some of these features are the Do-What-I-Mean system, History package, and, within the History package, the first UNDO function [2].

The Do-What-I-Mean (DWIM) system, conceived of and designed by Warren Teitelman, is a semi-intelligent autocorrect for simple programming errors. When enabled, DWIM causes Interlisp to call the FAULTEVAL (or similar) function whenever an error is encountered instead of throwing an error as normal. This function then automatically corrects simple mistakes in the code and continues running the program, only noting that a change was made — doing what the programmer meant, not what they actually typed. If an error is determined to be potentially correctable, but the measured 'distance' between a correction and the error is past a certain threshold, the program stops and requests user confirmation for a change; if yes, the change is made and the program continues as normal. DWIM is a realization of Teitelman's idea that human time is more important than computer time [2].

The History package was the first of its kind. Instead of tossing away the information contained in operations performed by the user, the History package tracks the history of operations and allows their examination and resubmission, often with substitutions. This feature is now a staple of modern terminal windows. The UNDO function, then, can use the recorded information about each operation to reverse operations entirely, including any changes they made when first performed. The UNDO function also allows the user to undo operations out-of-order, e.g., the user can make six changes via six different operations, UNDO only the first operation, and the information changed by the first operation would revert without affecting the intervening operations. An entire edit session can also be undone, if the user wishes [2].

All three of these key features were introduced in 1970, when Interlisp was still BBN-Lisp. In 1972, Daniel Bobrow and Warren Teitelman left BBN and joined the newly formed Xerox Palo Alto Research Center. BBN and PARC continued supporting BBN-Lisp, with BBN handling low-level development and PARC taking over the LISP-based software package and utility development. To acknowledge that BBN-Lisp's development was no longer solely at BBN, its name changed to Interlisp after the move. Years of development at PARC resulted in further extensions to Interlisp (such as an advanced graphical window system in 1976), and in 1982, Interlisp-D was developed for Xerox PARC's computer workstations, fittingly called D-Machines [2].

D-Machines were high-performance personal workstations that ran pure Interlisp, all the way down. The operating system, development environment, file manager, debugger, display settings, window manager, and everything else were all written in Interlisp-D, which was a further extended version of Interlisp that included, among other features, the ability to operate within a graphical user interface.

Since every part of the system is written in the language of development, every aspect of the system is entirely modifiable without ever leaving Interlisp; programs themselves are stored as lists and can be manipulated like any other list can, allowing programs in Interlisp, like LISP, to write other programs. Over the next twelve years, Interlisp-D underwent various releases and, in 1985, began incorporating compatibility with parts of Common Lisp, another popular LISP dialect. 1988 saw the Medley 1.0 release, which ran on all Sun Microsystems workstations with SunOS, and in other Unix-based systems such as IRIX, AIX, HP-UX, and Ultrix.

In 1989, Xerox stopped supporting Interlisp development, instead passing the reins to Venue, a corporation headed by ex-PARC Interlisp developer John Sybalsky, with the agreement that Venue would maintain and develop Interlisp in exchange for the right to make derivative works. In 1992, Venue released Medley 2.0 for the MS-DOS 4.0 and the Xerox Daybreak, the last of the D-Machines. Interlisp won the ACM Software System Award in that same year.

After this point, Medley's history becomes much fuzzier. It is unclear exactly when development stopped, but scattered documentation shows that in 1999 Venue was distributing Medley 3.5, and somewhere between then and 2009, the year Sybalsky passed away, Medley's development ceased.[2] The version with which the Medley Interlisp project resumed development in 2020 was a fractured and in-development version of Medley 3.5, gathered from the remnants of the Venue corporation and an employee of Venue.[3]

This is all to say: Medley is the final version of Interlisp, which is a dialect of LISP, and runs in a self-contained environment that is entirely written in Interlisp; the operating system, the debugger, the development environment, the application being developed, and everything else are all written with the same toolkit. As a residential programming environment [3], mastery of Medley allows for rapid prototyping and a level of interactivity and customizability of the programming environment not found in modern systems. The effort to preserve and modernize Medley serves to both continue development of a system unique to those available today and to conserve and perhaps even bring attention to ideas of exploratory research-focused programming and residential programming environments embodied in the system.

III. The Road to Revival — Barriers to Progress

The project has encountered many challenges in its efforts to recover Medley, some more unique to the project's situation than others.

Perhaps most obviously, Medley Interlisp is old. The original developers on the project returned to Medley's develop-

[2]R. Kaplan, Co-Founder, InterlispOrg Inc., personal communication.

[3]These 'dark ages' of Medley could potentially be illuminated via thorough investigation of email records obtained from Venue, but the project team has not prioritized this so far.

ment having spent 30 years away from Interlisp. The process of relearning Medley has been a gradual and constant one and, as one might expect, has slowed progress at times. Members who did not previously know Medley have had to devote time to a system whose learning curve has been described as almost vertical [4]. This situation is exacerbated by the scattered state of Medley's documentation. Further, some of the project's members who have been uninvolved in software development for several decades have had to devote time to learning more modern software development tools, such as Docker and Git, which did not exist when they were last in the field.

Medley is also far from bug-free [5]. As a natural consequence of bringing a program from before the turn of the millennia into the 21st century, some Y2K bugs in Medley have been found (and fixed) and there are likely more still undiscovered. Important aspects of system architecture have also changed, such as how Medley was originally built for big-endian systems, but virtually all modern personal computers are now little-endian.

Further, Interlisp's implementation was based around a virtual machine with an instruction set specifically designed to support LISP operations [6]. On the D-Machines, the LISP virtual machine was implemented in microcode (small, extremely low-level instructions for the computer). However, when attempting to port Interlisp-D to more modern non-D-Machine hardware (initially Sun Microsystems's SPARCstations, circa 1989), which could not be microcoded, the virtual machine implementation had to be rewritten in C so that it could compile down to the host operating system's native instruction set. This virtual machine implementation in C, developed with collaboration between Xerox PARC and Fuji Xerox, is called Maiko and is still in place with Medley today. As part of the project's work, Maiko had to be updated and now must be maintained to allow Medley to continue to 'play nice' with modern operating systems while remaining accurate to its behavior on D-Machines.[4]

Furthermore, as mentioned in the previous section, Medley was not finished; the project resumed development from several large fragments of Medley from at least three different sources which were all slightly different versions.[5] The project's initial work went almost exclusively towards investigating these fragmented pieces of Medley, discovering where they fit together, then actually fitting those pieces together to end up, after a whole year, with a build of Medley that *actually worked*. This is all in addition to the codebase containing numerous incomplete implementations of features which also had to be identified, deciphered, and implemented after a 30-year break from so much as thinking about the system itself.

In the process of relearning Medley, the project also encountered issues with Interlisp's documentation. There are multiple large reference documents for Interlisp, all of which are partially out of date for the version of Medley the project is working with. The most recent documentation (which is still not accurate to Medley's current implementation) is the *Medley Language Reference* published in 1993 under Venue, which encompasses a hefty 787 pages. There exist other publications, but in addition to being out-of-date, much of their contents were never validated prior to publishing. Further, the actual tools of documentation, all of which are written in Interlisp, still require substantial recovery work before full functionality is restored.

The work done in the Medley Interlisp project is volunteer-driven. If the project wishes to spend funds on anything, it must either come from a grant, or from the private funds of someone generous enough to donate to its efforts. This also, of course, means that the project is dependent on the willingness of its members to volunteer their time and effort toward Medley's recovery.

Lastly, a specific goal of the project is to update Medley Interlisp's Common Lisp implementation to the 1994 ANSI Common Lisp standard. The project possesses a chunk of code that would bring Medley significantly closer to the ANSI standard; however, it consists of approximately 90,000 lines of code in total, and it would need to be implemented in the 'guts' of Medley itself. Additionally, since Medley development has certainly diverged from the implementation this code was originally designed for, incorporating it now would be a major effort, requiring extensive time and expertise on the part of experienced developers familiar with Medley.

IV. A Survey of Current Progress

This section covers what the Medley Interlisp project has been working on and what it has accomplished so far, along with some notes on what remains to be done.

A. Building Membership

The Medley Interlisp project attracted its current membership by proactive outreach and communication regarding the project's needs at various events [7]. The project had specific initiatives for which its members opened the door for focused assistance, such as: bibliographical maintenance, LispUsers (user contributed packages) documentation, and outreach in the form of video demonstrations and tutorials of various key Medley features. The project team made calls for contributions towards these initiatives and more at related symposiums or conferences they attended. The project remains open-source on GitHub today and still relies primarily on volunteer efforts of interested individuals to perform its work.

Members contribute to the project for a variety of reasons. Some members of the project team have specific goals related to Interlisp in mind, ranging from linguistic theory research, running applications that were built in Interlisp, or reclaiming the development cycle that Medley Interlisp allowed, but others contribute for the sake of curiosity, learning more about Medley, and the excitement of bringing a historical system like no other to the present. The eventual goal of the project's recruitment efforts is to achieve 'escape velocity', as put by

[4]N. Briggs, Member, InterlispOrg Inc., personal communication.

[5]L. Masinter, Co-Founder, InterlispOrg Inc., personal communication.

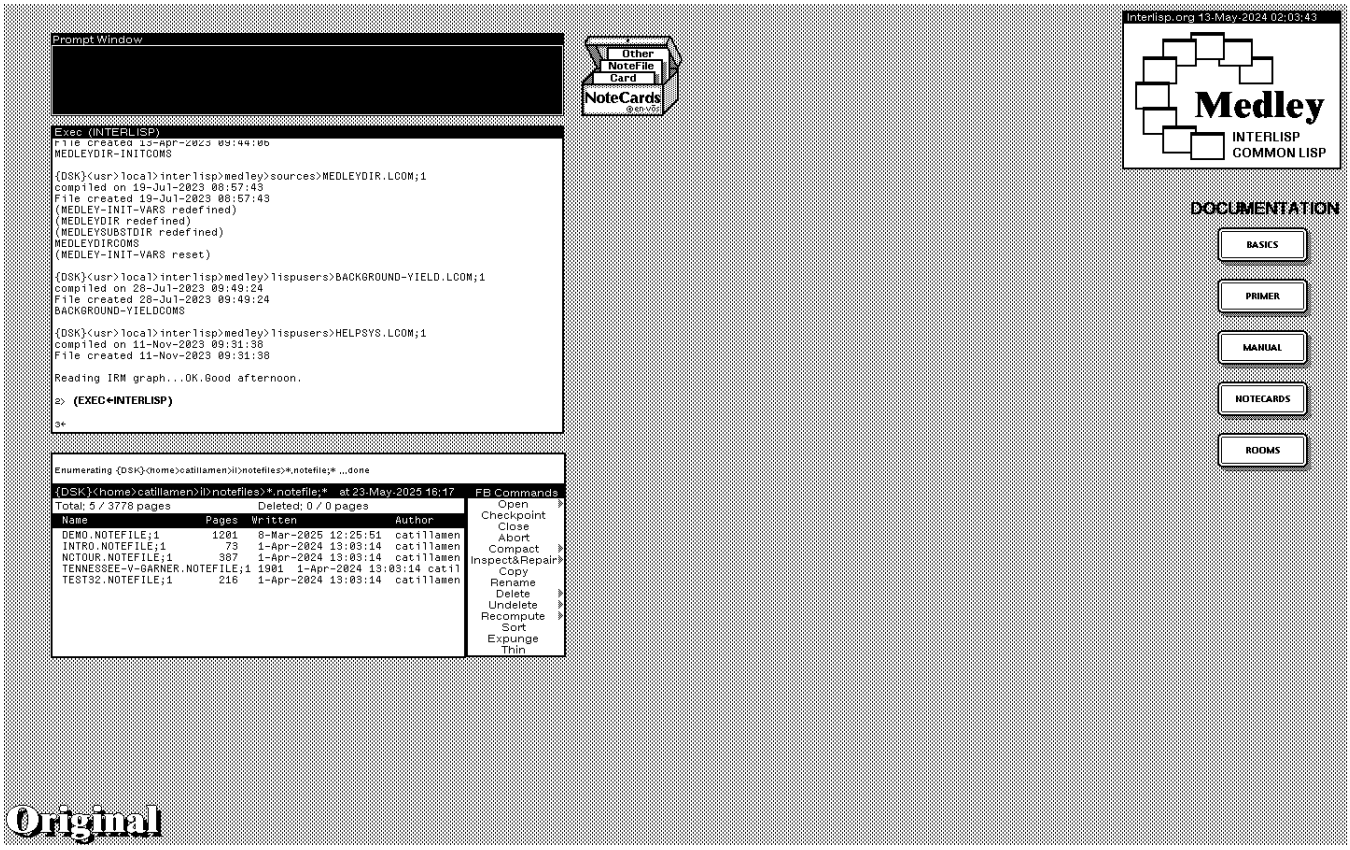

Fig. 1. Medley running on a Windows 10 system [8].

Larry Masinter.[6] Currently, the project's founders constitute a substantial portion of its active membership. In investing effort into outreach and recruiting new members whenever possible, the project hopes to increase the chance that it will continue to live even should its founders and current core membership no longer be able to contribute.

B. Modernizing Medley

The main body of work has gone into tinkering with the software of Medley itself. Medley has been updated to run on modern operating systems such as Linux, macOS, Windows 10 and 11, and more (see Fig. 1) [8]. A system called "Medley Online" has also been developed, which runs Medley in a Docker container that users access through a virtual network computing (VNC) connection [9]. Any files created are stored entirely within the Docker container, and users can save files after making an account. Without an account, users can still access the unrestricted system. This enables users to experience Medley without installing a single piece of software on their own machines so long as they have an internet connection, drastically raising accessibility; Medley could even be accessed in a place such as a public library with virtually no additional hassle. There is also a version of Medley Online that uses WebAssembly to run in a user's browser itself; it is currently in an experimental state and still under development, but accessible [10].

A software package called GITFNS (FNS being shorthand for "functions" — a common naming scheme in Medley) has also been developed for Medley [11]. This software package allows Medley to interact directly with the Git version control system in a variety of ways. Integrating with and adapting to modern software development is especially important due to Medley's own affinity for easing development cycles; with access to easy communication between Medley and Git, Medley is able to integrate with Git-based development rather than acting merely as an interesting alternative or even a stick in the proverbial bike wheel. With GITFNS, Medley can more easily coexist in the modern software development landscape.

Though Medley's implementation of Common Lisp is still out of date from the 1994 ANSI Common Lisp standard, the project has implemented LOOP, a major iteration construct present in the ANSI standard [12]. While there are still many discrepancies between Medley Common Lisp and the ANSI standard, LOOP's implementation is a large step that had significant impact without requiring prohibitive amounts of time or expertise.

As for updates that allow Medley to more easily interact with modern systems, a clipboard function has been implemented in Medley. The software package enables copying, cutting, and pasting into and out of Medley using the system

[6]L. Masinter, Co-Founder, InterlispOrg Inc., personal communication.

TABLE I
OUTDATED ENTRIES IN THE GLOSSARY OF *Medley for the Novice*.

Glossary Entry	Definition
button	(1)(n.) A key on a mouse. (2)(v.t.) To press one of the mouse keys when making a selection.
menu	A way of graphically presenting you with a set of options. There are two kinds of menus: pop-up menus are created when needed and disappear after an item has been selected; permanent menus remain on the screen after use until deliberately closed.
mouse	The mouse is the box attached to your keyboard. It controls the movement of the cursor on your screen. As you become familiar with the mouse, you will find it much quicker to use the mouse than the keyboard.
Mouse Cursor	The small arrow on the screen that points to the northwest.
window	A rectangular area of the screen that acts as the main display area for some Lisp process.

clipboard. Medley was originally built using Xerox Character Code Standard (XCCS) and uses it by default when reading and writing files. However, Medley also now has built-in input-output facilities for converting between XCCS and UTF-8 Unicode, meaning that Unicode poses no barrier for reading files into or printing files out of Medley. In the case of the clipboard functionality, pasting text outside of Medley converts the text from XCCS to the operating system's encoding as determined by the SYSTEM-EXTERNALFORMAT variable in Medley — which will, in nearly every case, be UTF-8 Unicode.

C. Medley Primer

In working to update and ease Medley's documentation and learning curve respectively, the Medley Primer, also called *Medley for the Novice*, has been undergoing extensive review and overhaul, and is currently in the middle of a rewrite. The overall goal is to review and polish the document's content while improving its pedagogical value for beginners to Medley Interlisp. In the process of reviewing, restructuring, and rewriting such an old document, the project has encountered several interesting issues.

The Primer has several instances of accurate-yet-outdated information. Its glossary defines a variety of terms still useful today, but also some that, in the modern day, become somewhat redundant. Some examples, taken verbatim (peculiarities of capitalization included) from the Primer's glossary, are shown in Table I.

For most users, this information is entirely superfluous. Even describing a mouse as "the box attached to your keyboard" is often untrue, since most modern computer setups have their mice plugged into a USB port on the computer itself rather than the keyboard, or connect wirelessly; and yet, these entries have a certain charm about them. They offer a view of the time that Medley was created in, which can be valuable historical information in its own right (or, at least, worth a few chuckles as someone scrolls through). For those unfamiliar with early computing, whether due to not being born yet or simply not being involved at the time, these entries can also serve as contextualizing information, showing that these matter-of-fact features of computing were at one time new and warranted defining.

Considering the historical value of these almost-parachronistic definitions, we have chosen to let them remain in the Primer. Though we have yet to decide its exact form, we plan to flag these entries in some way as historical — perhaps with a particular color, or even an antiquated clock emoji. Whatever the eventual specifics, the aim is to highlight the historical value their presence provides to the Primer while ensuring the Primer presents itself as a current, updated beginner's introduction to Medley. We anticipate other historical software recovery projects will have similar encounters when sifting through old documentation and we encourage their members to also consider the historical value of these parachronisms.

As we reconsider the Primer's previous approach to pedagogical design, we have returned to asking foundational questions about the Medley system's purpose and benefits for newcomers. How did Interlisp and Medley ask users to think differently in response to the standard mindset around programming of that time? What new culture did this system promote and encourage? What framework and lens should readers adopt as they dive into the Primer? What does this Primer aim to achieve, three decades of advancements later, situated in the current socio-technological climate? And more broadly, what is the relationship between the reader and the text? Staying true to technical writing conventions of that time, the Primer often took liberties in establishing a formal, distanced relationship with readers. The resulting tone and narrative uses passively voiced sentence structures delivered prescriptively, simple but independent examples to demonstrate various features of Medley and Interlisp, and encourages readers to learn intertextually. In contrast, breathing new life into this legacy Primer thirty years later requires a significant reframing of this document's self-presentation (and self-preservation) tactics. A necessary shift, from a Primer for a once-novel software environment for exploratory programming, to a Primer for a digital playground for exploration of historical digital artifacts, processes, systems, and conventions.

Our intended readers range from students in software archaeology courses, to one-stop hobbyists, to seasoned programmers, to researchers of computer history. The modernized Primer addresses two realities of these audiences: (a) readers with fleeting interests who are not quite ready to commit to a lengthy intertextual learning process of a complex system and a programming language new to them, and (b) readers who are well-versed in computer systems, comfortable with programming and are looking for adventures beyond the fundamentals. In acknowledgment of both use cases, our ongoing rewrite combines a few solutions: (1) an introduction to the Primer that contextualizes Interlisp and Medley in history and establishes why they were important to programming, what

readers can gain from this text, and how they should think about engaging with this system; (2) an authorship style that is personable, direct, and inclusive; (3) projects to complete, encompassing each section/chapter, which act as hooks that ground the learning process in solving interesting problems rather than unguided exploration; (4) broader projects and additional resources for further research and creation.

D. Interlisp Bibliography

The Medley Interlisp project continues to build and maintain an Interlisp bibliography, hosted in Zotero. The bibliography is intended to be an exhaustive enumerative bibliography, aiming to collect any and all works related to Interlisp and its various versions. This ranges from academic articles, reference manuals, software package documentation, advertisements in magazines, video lectures, and more. It also includes files and associated metadata for anything that discusses or pertains to Interlisp, Interlisp's development history (such as some LISP, BBN-Lisp, and Common Lisp items), or applications built in Interlisp. Essentially, the bibliography's scope encompasses anything and everything directly related to Interlisp as a topic.

Items were added to the bibliography from personal collections and light searching based on project members' own knowledge of notable works, such as Interlisp Reference Manuals, textbooks, and other notable publications.

Until recently, the bibliography was highly disorganized. Recent efforts have been made to tag items within the bibliography, clean up metadata, date and acquire digital files for as many entries as possible, and reorganize the bibliography's hierarchy. Some time has also been devoted to removing irrelevant items, or at least setting them aside. Initially, items were added to the bibliography in large groups, and this resulted in some entries that did not fit the above qualifications of relevance to Interlisp. We would have benefitted from clearer bibliographic standards from the start, or at least a more thoroughly defined organization schema, so that documents could have been tagged as they entered the bibliography.

The bibliography can be useful to a range of people, from computer historians to software developers working in Interlisp, or even those more casually interested in early programming (or specifically Interlisp). Of the documents referenced in the bibliography, the project hosts as many as possible in Zotero or on the project's website, [13], linking to external repositories only when necessary; the exceptions to this are stable external repositories, such as the Internet Archive.

V. External Collaboration — Freelancers and Universities

The Medley Interlisp project has collaborated on several occasions with external groups, some instances more successful than others. This section will briefly describe each instance.

A. Independent Freelancer Collaboration

In 2021, the project's members were seeking bibliographic expertise to assist with tidying up many unsorted, untagged, and generally disorganized items in the Interlisp bibliography shortly after its inception. To this end, the project briefly hired, through the freelancer marketplace Fiverr, a small number of freelancers that had proficiency in Zotero. While contracted, these freelancers went through many items in the bibliography and added tags and other metadata.

This process highlighted some difficulties in working with external parties, however. Due to their tenuous involvement with the project and lack of experience with Interlisp and adjacent topics, the freelancers' tagging of items in the bibliography was often surface-level, or even inconsistent and inaccurate. They were also unable to meaningfully assist with the broader organization of the bibliography. This was largely due to their lack of familiarity with the Medley Interlisp project's niche situation; gaining that familiarity requires time spent embedded within the project, its history, and its goals.

B. University Collaboration

The project found more fruitful collaboration with the Software History Futures and Technologies (SHFT) group, a post-secondary research group led and organized by Eric Kaltman. Through SHFT, the Medley Interlisp project initially hired several California State University Channel Islands undergraduate students to assist with the project's work in 2023. These students assisted in ways such as tagging bibliography items and filling metadata, as well as public outreach in the form of several tutorial videos uploaded on the project's YouTube channel.

More recently, since May 2024 the project has been collaborating with two University of Alberta (UA) graduate students, authors Young and Hasnain, funded as Graduate Research Assistants half by the project's funds and half by the UA's departmental funding. Young and Hasnain have worked primarily on the Interlisp Bibliography, the Medley Primer rewrite alongside other introductory documentation, and publications such as this one.

The project found that its more prolonged collaboration with students through SHFT was more fruitful than hiring freelancers for two reasons. First, students interested in historical software recovery and computer history can be recruited from departments relevant to the project's needs, which provides these collaborators with a higher baseline upon which to build than in the case of a freelancer unspecialized in the project's niche. Second, universities can sometimes assist in providing funding for the students, easing some of the burden on the project's own limited funds; this then facilitates the embedding of external collaborators for longer periods of time, which increases their familiarity with Medley and the project as a whole, and, consequently, their ability to assist the project's efforts.

VI. Conclusion

As technology continues to advance, thousands of pieces of software are inadvertently left by the wayside. Though Interlisp and Medley are largely forgotten, the Medley Interlisp project finds significant value in considering what useful things

might have been left behind in obsolete systems such as these. Viewing historical software systems as an opportunity, the Medley revival effort is predicated on the idea that modern software development could benefit from some of the ideas infused into obsolete systems. Many of these ideas are no longer present in the modern programming canon not because they were bad ideas, but due to things like lack of support over time or shifting focus to newer and 'fresher' technologies without realizing how useful some ideas would remain in the future. To reexamine these ideas or merely document their existences, software recovery efforts like the Medley Interlisp project are needed. Many historical software systems no longer used in the modern day are endangered; they can still be saved by intentional efforts, but may be lost completely if nothing is done.

Above, we outlined the history of Interlisp and the Medley Interlisp project, broadly described the project's roadblocks, and gave an overview of its current progress. We also addressed the project's involvement with external organizations, whether freelancers or university research groups, and outlined the three authors from UA's involvement with the project's efforts. Recovering a seminal software system such as Medley opens the door for all levels of historical research focused on early computing environments. We hope that in documenting the Medley Interlisp Project's efforts, other software recovery efforts may benefit from an example that came before. While we did not have this benefit, the project's approach from the start has, in a way, embodied the exploratory programming paradigm that has undergirded Medley all along (and, fittingly echoes the LISP read-eval-print loop): devise, build, deploy, gather user feedback, refine, deploy, gather user feedback, refine, deploy, gather, refine, deploy, gather, again and again, until we went from having little clue what we were doing, to proving something about why software recovery efforts have been and will continue to be worthwhile for generations of software past and future.

References

[1] B. Liskov, J. McCarthy, and P. Abrahams, "LISP SESSION," in *History of Programming Languages*, Elsevier, 1981, pp. 173–197. doi: 10.1016/B978-0-12-745040-7.50009-0.

[2] W. Teitelman, "History of Interlisp," in *Celebrating the 50th Anniversary of Lisp*, in LISP50. New York, NY, USA: Association for Computing Machinery, Oct. 2008, pp. 1–5. doi: 10.1145/1529966.1529971.

[3] E. Sandewall, "Programming in an Interactive Environment: the "Lisp" Experience," *ACM Comput. Surv.*, vol. 10, no. 1, pp. 35–71, Mar. 1978, doi: 10.1145/356715.356719.

[4] P. Amoroso, "My encounter with Medley Interlisp," Paolo Amoroso's Journal. Accessed: Mar. 10, 2025. [Online]. Available: https://journal.paoloamoroso.com/my-encounter-with-medley-interlisp

[5] "Medley Interlisp project GitHub Issues," GitHub. Accessed: May 22, 2025. [Online]. Available: https://github.com/Interlisp/medley/issues

[6] J. S. Moore, "The Interlisp Virtual Machine Specification," Xerox Palo Alto Research Center, 1976. [Online]. Available: http://www.bitsavers.org/pdf/xerox/parc/techReports/CSL-76-5_The_Interlisp_Virtual_Machine_Specification.pdf

[7] "News and Status Reports," The Medley Interlisp Project. Accessed: May 22, 2025. [Online]. Available: https://interlisp.org/project/status/

[8] "Install and Run," The Medley Interlisp Project. Accessed: Mar. 16, 2025. [Online]. Available: https://interlisp.org/software/install-and-run/

[9] "Access Medley Online," The Medley Interlisp Project. Accessed: Mar. 16, 2025. [Online]. Available: https://interlisp.org/software/access-online/

[10] "Medley running in WebAssembly." Accessed: Mar. 16, 2025. [Online]. Available: http://wasm.interlisp.org/medley.html

[11] Kaplan, Ronald M., "GITFNS." Medley Interlisp Project, May 2022. [Online]. Available: https://files.interlisp.org/medley/lispusers/GITFNS.TEDIT.pdf

[12] "Implement the LOOP Common Lisp macro · Issue #1578 · Interlisp/medley," GitHub. Accessed: May 22, 2025. [Online]. Available: https://github.com/Interlisp/medley/issues/1578

[13] "Interlisp Bibliography," The Medley Interlisp Project. Accessed: Mar. 17, 2025. [Online]. Available: https://interlisp.org/history/bibliography/