On Mimix

This paper introduces Mimix, a future computer system for research, writing, collaborating, and teaching. Mimix is inspired by the ideas of its forebears such as the Memex, NLS, and Xanadu. I hope to achieve with Mimix what those platforms could not: a modern, deliverable system which can be used by anyone to augment his or her own intellectual work.

"If I have seen further, it is by standing on the shoulders of giants." -- Isaac Newton

Name & Inspiration

Mimix (pronounced "mimics") is named after Vannevar Bush's 1945 proposal for the Memex, a machine with similar goals. Bush's vision was constrained by the technology of the day and could not be realized as he described it with a microfilm storage system. The Memex's purpose was *memory extension*. It would keep track of the researcher's path through other people's documents as well as his or her own writings, annotations, and the links between them. In 2018, we still don't have such a system.[1]

Another key driver for the Mimix idea was Douglas Engelbart's NLS or oNLine System, first demonstrated at The Mother of All Demos in December of 1968, a month after I was born. By this time, the technology existed to achieve the goals of Bush's Memex (or this Mimix, for that matter), but NLS was constrained by a terrible user interface that Engelbart himself found difficult to manipulate.[2] What was not clear in 1968 is how humans and computers might interact with such a system.

Before NLS, hypertext inventor Ted Nelson envisioned a non-linear writing system he called Xanadu. It was notable for the realization that people need to connect their topical references visually as well as textually, often on the same screen. In the real world you might put all your stuff for a Hawaii trip in one place so that you can identify it visually. You don't run around and tag everything #hawaii. Unfortunately, Nelson became bogged down in creating a 3D version of his interface and also failed to deliver a product that people could use.[3]

A final inspiration for Mimix is the Lisp machine, a family of computers appearing around the same time as these other innovations. Lisp is relevant to Mimix because it's ideal for working with lists. Lists (and lists of lists) are a terrific way to represent a trail of research, notes, and writing. Furthermore, Lisp machines are designed to be all-encompassing with the operating

[1] Software has to be delivered to be real. This was a given as a mainframe programmer at IBM. If your code could not be delivered and debugged and delivered again to a customer, it wasn't ever going to leave the lab.

[2] The event is startling in its prescience but also humorous in its failures. Along with the Mother of All Demos, Engelbart had also invented "Death by Demo" as Bill Gates and many other famous presenters would come to call it. As soon as you show something to other people, all the brokenness appears.

[3] I don't think 3D is the answer here. Reading and writing are still essentially 2D activities.

system often written in Lisp itself. By definition, they have no fixed memory limit. These ideas will become important in the practical execution of the Mimix machine.

Reading & Research

Mimix is designed for **parallel reading** in which two or more documents appear on-screen at the same time.[4] The system allows drawing links between any parts of two documents.[5] You can connect a paragraph or table from a scanned paper book to a new paper you're writing now. This is a fundamental ideal in Mimix. It's your reading + your notes + your writing.[6]

Mimix is optimized for **nonlinear writing**, the kind often done with note taking, creative writing, and academic material.[7] Mimix lets you create a complex document like a screenplay[8], dissertation, or medical report and build structure on the fly. You write the parts in any order and relate those parts visually and textually later. While you're reading and writing, Mimix is building **metadata**[9] which helps show the relationships between your writing and your sources.

Mimix lets you add links between any documents you read, your own writing, and your collection of notes and pictures. You can connect elements of interest in any two documents, no matter who originally wrote them.[10] You can see all the linked relationships in your documents at once, at any level of detail. Using metadata, Mimix can show the people, dates, and keywords related to your links as well the links themselves.

---

[4] Parallel reading is a fundamental concept in both Bush's Memex and Nelson's Xanadu.

[5] Graphically drawing a link between pages is from Xanadu. Nelson was deeply disappointed with the watered-down HTML link structure we have instead.

[6] Keeping your reading and writing together was key to both Memex and Xanadu.

[7] Nelson formalized the idea of nonlinear writing, which is really the only kind most people do. We pretend that a document begins at the beginning and ends at the end, but this is rarely the case. Like a piece of music, riffs and melodies appear to us first. We need to see (and hear) how these relate to compose something of value. The reason we cling to this linearity in writing is that's how movable type worked on a printing press. Several authors have tried to get around this. The first I encountered was William Faulkner whose stream of consciousness writing style made the reader "jump the page." Ram Dass's *Be Here Now* used typography and graphics to encourage a non-linear flow.

[8] I have three screenplay ideas in various levels of completion and, like most writing tools, all of the software I've tried for writing them is terrible.

[9] Many of the uses of metadata presented here are from my previous invention known as Searchadillo or Project Koko. Koko was a tool which analyzed a user's query, consulted a variety of API's, performed semantic analysis on the result, and created a customized user interface based on the semantic context of the results. Koko used a DSL called IL, Interface Language, in order to transform queries and results into a uniform structure which could be analyzed. It is the subject of the US Patent "System and method for sending various application programming interfaces to a customized user interface."

[10] Another idea from Nelson that we still don't have today.

All of the documents you read, your exact path through them, and all of your notes and links are recorded in a **stream**.[11] You can review, playback, or extract materials from any part of any stream.

Important words in your writing are recognized automatically in order to build metadata. Every occurence of these words and the relationships between them are cataloged and instantly accessible. An author of fiction might use this feature to visually organize everything known about a single character. A drug researcher might consult metadata about chemical structures on the same screen where a paper is being drafted that mentions them.

Your metadata, links, and streams can be used to create unique **views** of your data automatically, such as automatic **timelines**, instant **biographies** (or bibliographies), and automatic **slideshows**.[12] Mimix can help show the relationships between the things you've read and written and what others have written and can suggest new documents that might belong in the current view.

Extreme Metadata

The concept of extreme metadata is pivotal to Mimix. First of all, it's meta because it isn't your data per se, but it does contain links back and forth from your data. It's extreme because it's bigger than any personal metadata of yours and also used in more ways than you might expect.

[11] The idea of stream recording all of a user's interactions is straight from Bush's Memex. He proposed a microfilm recording with each page that was read or written being recorded linearly on film, the book scanner, film recorder, and dual screen playback system incorporated into the device. A Memex session would result in an indelible record of the author's research and compositions which also showed the relationship between these. A previous invention of mine called TIDE, the Trellis IDE, invoked this idea for software development. Trellis was an early framework I wrote for reusable components built in Microsoft ASP and used to run my web business. TIDE was to record all the developer's actions and suggest resources like code or documentation, even during debugging. State would be maintained and could be inspected or rolled back. TIDE used a DSL call TEL, Trellis Execution Language, which was itself XML.

[12] The idea of variable views of the same data created on the fly can be sourced to Douglas Engelbart. His NLS system transformed drawings into lists and lists into outlines or narrative text. In Mimix, automated timelines could be created by scanning for atoms which are dates and organizing the surrounding material in chronological order. An author who is writing the history of the French language could consult a timeline of all the books he or she has viewed. An automated biography works the same way for people. Every author mentioned has canonical information available as well as his or her own notes. These can be compiled into an instant and automated biography. An automated slideshow is simply a visual presentation composed of elements from the user's reading and writing so far. The order of importance of the items and the level of detail they require can be inferred from weighted metadata.

Mimix recognizes semantic **atoms**[13], terms that mean just one single thing. If you're writing a book about the history of nuclear power there are many names which appear repeatedly. Einstein is one, but there are lesser known names like Leslie Groves who are also important. It would easy to generate a list of the most notable people in nuclear power. There can't be more than 10,000. It could be done even by scanning a few documents as you researched them.

What if we started with a list of the 100 million best known people in history? These would be proper name atoms. Anytime you read or write about one of these people, you're reading and writing about the same person as countless others. Naturally, your work is related to everything else they've written about that person, too. Just typing the word "moon" onto a blank page invokes a million facts, yet today's software makes you go to several different places to find them. Shouldn't the facts about the moon come to you as you're writing, since they're already known and easily found?

Now a programmer will say here that theoretically you could be referring to any moon, and of course that's true. But we can figure it out. Anyone on earth who just says "moon" is talking about Earth's moon. But if I were to write the phrase "moons of Jupiter," I'm talking about something else. Several new moons around Jupiter were recently discovered. If I type even that phrase "moons of Jupiter" into a blank page I am referring to all of these old and new discoveries whether I know about them or not. I shouldn't need to search to find out that new ones were discovered.[14] That's already known. It's canonical metadata, in Mimix terms.

If I were to begin to write about the moons of Jupiter, why do I have to go and cull all of these facts from different websites, programs, and books? Why do I have to ask for the relevant names, people, timelines, and stats about these moons? Doesn't the phrase "moons of Jupiter" already contain all that information?[15] Of course it does, and a modern computer should know that and be able to retrieve and relate that to my writing, even as I pen those first three words.

When we write about Einstein or Stravinsky or anyone else, there's a whole world outside of what we've read and written so far that is also germane.[16] When you mention an atom such as a

---

[13] From the Greek *atomos* for "indivisible." The term has the same meaning in Mimix as in Lisp.

[14] The need to separately search for data is an artifact of publishing and paper indexing systems like the library card catalog. A more modern paradigm would be to expect data to come to you when it is already canonical. Instead of looking at "published" data as being outside of our own work, it makes more sense to integrate existing sources into the new work from the start.

[15] James Burke's *The Day the Universe Changed* is an eye-opening look at how we continually redefine reality, science, and language as humanity makes new discoveries and inventions.

[16] A term invented by Shakespeare. It's fascinating to think how many words and phrases he contributed to English. If I were writing on a Mimix machine, I'd hope to include lots of great examples of how his verbal art changed this language. Einstein later regretted his work on the bomb, which is surely germane to any mention of it. When Stravinsky conducted the Rites of Spring for the first time, people walked out of the theatre. Music was changed forever.

real person, a chemical, or a movie title, Mimix can see the relationship between your writing and reading and other writing about the same thing. By recognizing words like Stravinsky as an atom, Mimix can find and expand upon these relationships, including relating the composer to other atoms which you might want to review: the people, places, dates, works, and events to which he is related.

Even the 10 million most important names will have duplicates, so we must create separate Mimix atoms for Madonna the singer and Madonna the religious figure. This is easy to do. An atom's internal name in Mimix is separate from its display name. There can be as many John Smiths as we need and all can be kept separate.

We would need separate atoms for Paris Hilton the singer and the Hilton Paris Opera hotel since we'd be interested in different types of information about them. Writing about Paris Hilton would call up metadata about her as a person and artist: her bio, discography, and tour dates. Writing about the Hilton Paris Opera hotel would show metadata from France, such as hotel amenities and which Métro station to get off at. (It's Haussmann Saint Lazare).[17]

Paris has great architecture and if you mention the Hilton Paris Opera in a story about architecture, Mimix could offer metadata about the other architectural aspects of the neighborhood and they are legion. The story of why the station is named Haussmann is worthy of a book in itself. By design, Mimix leads to discovery of new, factual information from other sources.

But trying to find architecture information about a person would not make sense unless he or she were somehow involved in architecture.[18] Paris Hilton is a person involved in music who shares a name with a hotel brand. Renzo Piano is an architect who shares a name with a musical instrument. Writing about *that* Piano would cause Mimix to include architectural documents, references, timelines, etc. as well as people-type information in the metadata views, but nothing about music.

[17] This turns out not to be as easy to discover as you might think. You have to know a bit about the Paris Métro. The beauty of it is that there are so many stops so close together you can walk to the Hilton Paris Opera from several stations. Even the Opéra station of the same name will work. But the hotel recommends Haussmann Saint Lazare and Mimix would, too. An arriving guest is likely connecting by rail from the airport, another city, or another metro station, so we should suggest the simplest connection from those trains. If you're in the neighborhood, go see the Opéra Garnier and the terrific Fragonard Perfume Museum nearby.

[18] If I were writing in Mimix about people such as John Von Neumann or Fred Brooks and used the word architecture, I should get back an entirely different kind of metadata, namely stored program architecture and System/360 architecture. I do wish I were composing this text in Mimix right now.

Mimix's use of metadata outside your own data is important, especially canonical[19] metadata such as facts about people, places, and material objects. A huge amount of everyday writing refers to things which are known facts. When mentioning the Walt Disney Company we invoke a whole slew of facts about their history and operations. Even simple blogging relies on quick access to factual material about names, places, dates, and events.[20]

Any kind of academic writing or new invention requires exhaustive research of the existing domain, if only to weed out known problems and solutions. The design of Mimix greatly simplifies one of the most difficult tasks of writing: collecting and organizing your observations from various source materials. This leads to your new contributions, your "Ah-ha!" moments.[21]

Mimix extracts and separates atoms using context references and canonical data sources.
Names are only part of the semantic atom. Like the Greek *atomos*, Mimix atoms are "indivisible." But also like Greek atoms, underneath lies a much more complex structure.[22] In Mimix, all of these are atoms: 2019 Mirai, Toyotas, Toyota, Japanese cars, and cars. Each term refers to an increasingly larger group but still only means one thing.

Atoms are applied at the meaning that makes sense in the user's work. Cars means all cars everywhere and not just Toyotas. Toyotas means Toyotas of any year or model. Reading or writing about Toyota the company would produce different (and more wide-reaching) metadata than research or writing about the 2019 Mirai.[23]

Mimix ships with common lists of atoms for many everyday uses. It would be trivial to include not only the top 10 million people but also the top 10 million places, plants, animals, bacteria,

---

[19] This term is important in Mimix. It refers to data which is accepted to be correct and unalterable. Walt Disney's birthday or the correct spelling of portmanteau are facts. They only need to be stored once and can be consulted repeatedly. More importantly, all references to canonical atoms refer to the same thing. In other words, every time we write Walt Disney we refer to the person with the same birthday, published works, etc. There is a built-in relationship to atoms and their canonical metadata without the user explicitly having to find it. Whether data is accepted as canonical or not is up to the user, but one could imagine it would be easy to make many lists which are indisputable, such as the nuclear and chemical properties of the elements in the periodic table or the people, places, and dates involved in historic events. These are canonical data. It is also possible to consult two canonical sources for the same information, such as the price of bitcoin on two different exchanges or who shot JFK. The atoms bitcoin and JFK are themselves canonical and refer to only one thing each.

[20] It would be better if more blogging relied on factual information.

[21] Credit to Vannevar Bush again here for recognizing that you need your own materials organized along with other people's in order to get to "Ah-ha!"

[22] Far from being indivisible, the current standard model of physics describes 17 subatomic particles which comprise the atom and whose relationships are still not entirely defined.

[23] A Mimix machine would be able to show you that Mirai is Toyota's hydrogen fuel cell vehicle without me having to write this separate note.

viruses, molecules, books, movies, songs, ad infinitum. Being plain text, this information is small and being well known and well established, already canonical.[24]

Users with special needs could add custom metadata about their area of work: a list of 10 million great works of art, for example, could be shared among schools of all levels. Big companies like HP or Samsung already offer extensive information about every part they've ever produced and much of it is online, but finding, organizing, and accessing those facts is arduous.

Because every part of Mimix recognizes semantic atoms, the system can point out relationships between them and organize information in a way that complements your work. If you're developing software, Mimix might display a help system for not only the language you're writing in but also for the unique code you're writing. It should not be necessary to search for help or other code references when it's clear what you're writing about. This is also true of medical research, TV scripts, investigative journalism, and many other types of writing.

Using **weighted metadata**, Mimix can tell which atoms in your work are the most important and which are outliers. This is beneficial not only for organizing the structure of your work but also to discover new and potentially important insights which may be buried under more common material.

Using your stream, Mimix can suggest **alternate streams** through your data which you might not have considered. These can include new automatic timelines, slideshows, and biographies, as well as system-generated annotations and references from your other work.

Mimix can collapse any stream into a **simplified view** with the most important information organized in several different ways. Conversely, the system can create automatic **expanded views** of simple information you've entered, using the facts found in canonical metadata.

Fulltime Encryption

Because Mimix is expected to hold important private information, the entire system is encrypted. One way to implement this is inside a fully encrypted virtual machine or docker container. An implementation of this kind fits well with the Lisp machine idea presented later.

[24] The numbers here are arbitrary and purely illustrative. I have no idea how many important people, books, or chemicals there are and surely there will be multiple answers to those questions. The idea is that the user can load some set or sets of canonical data that he or she likes. If you were writing about the Musée du Louvre, you could accept their metadata as canonical. Today's systems are optimized for video and audio which are relatively huge compared to the size of the textual information I'm talking about. Vannevar Bush said of the original Memex, "The Encyclopædia Britannica could be reduced to the volume of a matchbox."

The Mimix system itself would provide the bridge to resources outside the VM environment. The VM environment would provide isolation from the host system.[25]

Format Free Text

In Mimix, data is stored as text without formatting. This has several advantages. First, formatting can be stored separately as another kind of metadata.[26] It can be applied to text or edited independently. Several kinds of formatting can be swapped in and out over the same text. Formatting can be reused between documents without affecting their content. There is no need to "markup" the Declaration of Independence four times to use it in a pamphlet, a video, a keynote, and a book or to do so 30 times in a class of 30 students writing a paper on the same topic. By keeping the formatting separate, all those instances can refer to the same text.[27]

Format free text allows collaborators to have their own formatting and annotations which are also just metadata. When reading or writing about the same text, each individual will often want to keep his or her own notes and reorganize the provided materials. Format free text enables these kind of collaborative yet unique views.

Finally, format free text makes it possible to export information from Mimix to other common file formats and systems. Any markup needed by the new system is created at the time of the export.

---

[25] IBM's VM operating system was fantastic. We could instantly create and access storage, retrieve canonical documents and have them printed anywhere, view any of IBM's source code, install any product or language, and setup test machines, just for starters. A forum system encouraged collaboration across job titles. Today's uncoordinated PC apps and websites look like a joke in comparison. These tools drew resources from all over the IBM universe without burdening the user about how they were provisioned.

[26] There have been many attempts to detach content from formatting but all of them that I know of require some kind of markup in or around text. Current formatting systems are of two types, both anachronistic. Systems that insert codes into documents are derived from WordStar codes which worked the same way. Systems that surround text with tagged markup like HTML are derived from IBM's SGML. Today, we have enough storage to keep all the versions of a text and its markup separately. When needed for display, the text can be parsed and the desired effects applied. One way to parse text like this is with regular expressions, though I'm not a fan. Unconstrained by the need to mingle word processing and data storage, the separated formatting information can be used to "format" the data in novel ways, such as to display a slideshow or to recall some automated metadata. Interestingly, prior to codes and markup systems, text was also format free. IBM punch card machines used fixed length fields to declare plain text input data which was then formatted by a language like COBOL into phone bills and Social Security checks. COBOL had extensive commands for formatting because text was always represented internally without it. Lisp is the same way. HTML, in contrast, has so much inline formatting that it's difficult to read or extract the actual words. As an example, try taking the atoms from CNN's home page, from a Google search result, or from a page in Wikipedia.

[27] The idea of all users of a text referring to the same document is from Ted Nelson.

Streamsharing

Mimix allows users to share any aspect of a stream with another user with a great degree of granularity.[28] You can choose to share a stream or anything that appears in it. You can share individual documents, pages, paragraphs, sentences, annotations, links, pictures, or metadata. It is not necessary to leave the reading/writing environment to spontaneously present or share with someone and there is no requirement that others join or maintain the same view. The power of Mimix comes into play precisely because each view is independent.[29]

Streamsharing is not document sharing, app sharing, or screen sharing. It opens up new possibilities in education, presentation, and collaboration. During streamsharing, each person may pursue his or her own path through the stream, including creating new materials and adding new references. By design, this creates a new stream belonging to the viewer.

When teaching, each student has all the teacher's original materials as well as his or her own notes and documents, all accessible from the same screen. Students can work independently and still access the teacher's materials. Students can share their own streams with other students using the same granular tools.

When presenting, the audience can annotate, edit, or extract atoms or metadata from any part of the stream without the presenter's involvement. Instead of slides and a list of links, audience members receive the presenter's full stream all at once. This becomes a new stream that the recipient adds to going forward.

When collaborating, participants can access shared materials without the burden of having their own notes and edits displayed to others. In a large scale organization, each user could have his or her own view of tools and reference materials. Corporations, public sector, and academia could allow users to organize and annotate their own materials from a shared stream.[30]

---

[28] Sharing a session or parts of it was a key aspect of the original Memex. Being a microfilm reel recorded in linear time, it was easy to give someone else a printout of any frames or hand them the entire spool.

[29] It isn't possible to leave the Mimix reading/writing environment for this is the only environment provided, a design decision inherited from Lisp and NLS. The independent views likewise come from Lisp in that each Mimix user is seeing his or her own memory space, although elements can be interchanged with other users. Engelbart was the first to show the idea of live collaboration without leaving the active computing workspace.

[30] Some of the ideas here are from a project I proposed while at IBM called the Prophet Interface Model. Introduced around the time that IBM and Microsoft were working on a unified graphical interface, the idea was to give users a portable interface which they could move between machines. With Prophet, users would be able to customize their own graphical interface to the IBM mainframe universe and access it from any connected device. Finding support only in the online forums and not from above, I picked up the phone and called John Backus, the inventor of FORTRAN and an IBM Fellow. The mainframe made it easy to find him with a single command and call him on our internal phone system. When I complained that the company didn't seem interested in pursuing the project, he said, "Well, don't take it personally.

Moving Away from Apps, Documents, and Windows

Today's tools for research and writing rely mostly on a paper metaphor. The simplicity of linear writing on paper is its Achilles heel. As stacks of papers pile up, we lose track of their contents and organization. While still offering paper and document affordances, Mimix moves away from these concepts by representing data underneath in one of three primitive forms: atoms, metadata, and streams.

Atoms represent the fundamental content containers in Mimix. The name Walt Disney is an atom. Ultimately, all data in Mimix resides in atoms.[31] Once created, atoms cannot be changed, but a new atom can be created with the same or different content.

Data about other data is metadata. Metadata about Walt Disney includes things like his birthday, filmography, and so on. Metadata resides in a list separate from the atom but applied to it. By creating a new list and applying it, we can change the atom's metadata. The appearance of an atom together with its applied metadata is used to create the view presented to the user.

The fact that different metadata can be applied to the same atom gives us flexibility in how we interpret an atom semantically as well as visually or organizationally. Walt Disney was controversial for several reasons, the best known of which is his stance on equality. Was Walt bigoted or inclusive? A Mimix user could load metadata that supported either side of the argument, as he or she chose. The result would be a different view of the source material. In any case, the various metadata all refer to the canonical atom of Walt Disney.[32]

The third kind of information in Mimix is a stream, a list of views over time. Creating or removing atoms and applying or removing metadata alters the view and is recorded in the stream. The

I'm working on the next generation of computer languages [Functional Programming] and they don't listen to me, either."

[31] A Mimix atom is a Lisp symbol.

[32] The overwhelming evidence is that Walt was not only egalitarian and inclusive but perhaps the greatest social engineer of all time, a theory advanced in Stephen Fjellman's landmark book, *Vinyl Leaves: Walt Disney World and America*.

stream is temporal and chronological.[33] Because the underlying atoms are never changed, we can extract data or play back the stream in its original view or with a different view.

We can create all kinds of traditional screen and paper-like representations from this stream of atoms and metadata. A long passage from a scanned book could be included in an original manuscript, moved around, and reformatted several times without ever changing its content. It could be associated with any number of notes or other papers and these included in a view or not, as desired.[34] "Documents" can be highlighted, written on, cut, combined, and paginated, but under the hood we are just applying new metadata to the atoms. A professor creating a curriculum outline could request a different view: a set of pre-filled slides, notes, and links. A technical writer or linguist could construct a reference manual or glossary automatically.

This holistic approach to information management makes Mimix more like an operating system or a self-contained machine than just an application. Mimix cannot rely on the outside operating system's file organization as it doesn't provide the atoms and views we require. We also cannot ask users to install, setup, backup, or secure a system like Mimix or to manage its metadata. Those features must be automated and isolated from the user and the hosting environment.

Solving the problems of the atom/metadata/stream system results in many ancillary benefits. We get automatic package control and system setup, automatic versioning of all files, and built-in redundancy. Using Mimix is more like using a mainframe than a PC, even though it happens on your own machine. Mimix manages the entire operating environment to ensure you always have access to all your streams. Mimix can find and correct many errors automatically, including data and factual errors, setup errors, missing files, etc.[35]

Built on Lisp

Mimix is built on Lisp and inherits many qualities of Lisp machines like the one commercialized by Symbolics in 1980. The self-contained nature of Mimix, including acting as its own operating system, is one of these Lisp machine qualities.

---

[33] A stream is temporal in that it is bounded by time two ways. It gets created in time by recording the user in real time and it can be collapsed or purged by the user at a later time. Streams are chronological in that they support a play, fast forward, and rewind metaphor. You can move through a stream in the order in which it was recorded or scrub through the recording as you would a video. Because the stream is not a video, there is no waiting between the playback of the real time steps you conducted to create it. More importantly, the stream's atoms and metadata are live during playback. You can alter them and explore from any point.

[34] The goals here are pure Ted Nelson.

[35] Self-correcting systems are nothing new but are rarely found in consumer-facing software.

Another feature drawn from Lisp is a global environment and global package system which together allow for any combination of data isolation and sharing. Lisp includes built-in data sharing between what in other languages would be called applications, objects, windows, models, or datasets.[36] This makes Lisp ideal for Mimix atoms and metadata which must be shared between many views.

The Lisp package system provides the isolation needed to allow the user to inherit and extend upon others' content streams. This "inherit and extend" methodology is foundational to Lisp and a natural fit for the research and writing use cases of Mimix. The same atoms and metadata you use can be copied between packages, allowing users to replicate and build on others' streams of work.

Lisp is ideal for hosting domain specific languages since it has a built-in macro parser. In order to describe atoms, metadata, and streams, we invent a Lisp DSL called MSL[37], Mimix Stream Language. MSL is a superset of Common Lisp and is built with Lisp macros and functions.[38] MSL provides high level operations for creating and destroying atoms, building and applying metadata, recording and sharing streams, and accessing and editing these elements.

Serialized State

A Mimix machine's **state** can be recorded by writing down all the MSL operations used to get there. If started from zero, a Mimix machine would have no packages, no atoms, no metadata, and no code other than built-in MSL. It would contain only one stream, that of the empty view where the user could begin working. Its state would also be the empty view.

With built-in MSL functions, the user could load an MSL file and recreate any previous state, including all or part of its atoms, metadata, and streams. A **snapshot** of the state can be taken at any point and edited or shared with others through an (encrypted) plain text file of MSL. The incoming elements can be added to the current environment or replace their previous definitions if the user chooses.

---

[36] Lisp is beautifully free of any of these ideas, though it supports them all. This allows new languages built on Lisp to include as many of these features as they want and implement them in any manner that benefits the programmer, without the burden of any additional language cruft.

[37] MSL's Hello World program is `(mimix "Hello World.")` which creates and returns a new atom with that data.
.

[38] A subset of Lisp wouldn't allow you to "jump the box" and have full access to Common Lisp when you need it. A superset of Lisp doesn't constrain you because anything that isn't MSL is Lisp. You can destroy all your data if you'd like. IBM's VM system had this feature, too. What resulted was not programmers wiping themselves out but all kinds of great tools and innovative ways to combine the layers. Users were given free reign to expand and customize the provided tools. MS-DOS famously allowed you to screw up your own stuff which gave people the freedom to start the PC revolution. Today's dumbed-down web and mobile apps have kept actual and potential developers in a box designed by platform providers.

Because the full system state is serialized, it can be backed up and verified against a hash. Dependencies can be identified by scanning MSL and often automatically corrected using hash-verified backups of previous states. Because the state is a plain-text object, it can be viewed and edited. The state recording (stream) is temporal and chronological in order to make the most sense to a human reading it.[39]

The Mimix machine's state is the result of playing back a series of MSL stream recordings or MSL code.[40] It can therefore be virtualized by spawning any number of independent states within the same Mimix machine and switching or sharing between them. Each swappable state as seen by the user is simply the result of playing back one stream vs the other.[41] No actual data can be destroyed in any view or stream because data is atomic.

States can also be exported along with all their dependencies. Because the export is (encrypted) plain text MSL, the importing system can make selective edits and incorporate the incoming stream in the way that serves the user best.

[39] These ideas are drawn from three areas. One is my prior invention called Sierra Red. Prototyped as a universal MLS for the real estate industry, it was built with a DSL called Vendor Language. It is the first implementation of a Turing complete domain specific language in XML. Vendor Language processed incoming data in many different formats (none of them XML) from different providers. The content and arrival times were unreliable. It examined incoming data field-by-field as it arrived, compared it with our rules and existing data, transformed it into XML, and added it to our database in the right place. VL was human readable so we could fix problems by editing our own code. A one-to-one correspondence between incoming fields and VL code greatly simplified debugging. The result was a serialized state. If the system stopped, it could be safely continued from the next line of VL without starting over. As proposed for Mimix, the state was the built-up product of the execution of the DSL. The second influence is IBM's SMP/E, a language and a coordinated series of tapes sent to mainframe customers that ensured they had the correct software and patches applied in the correct order. It could install, update, and remove software and order any missing tapes by itself. If an error occurred, the new tape could roll back the system to a known good state. It did all of these things while the machine was serving tens of thousands of users and without interrupting them. The software would simply not allow customers to build a non-working system. Once the changes were ready to go, SMP/E could swap out the new build for the old one with users seeing no loss of data. The third influence in the Mimix serialization system is from the blockchain, where hashed records are considered canonical and provably identical.

[40] There is no difference between a stream recording and MSL code except perhaps the order of operations and the means used to create it. You can record a stream and edit it later as MSL code or write code and edit it in real time with the editing tools.

[41] A new state can be bootstrapped by simply running the MSL code from some previous stream recording. There's no need to install the complete system. It can be selectively reviewed before starting the new instance. The plaintext nature of MSL allows extracting and examining the atoms and metadata inside the stream, including the use of custom tools the user might write. Dependencies can be detected and fulfilled automatically by pulling from the originating stream.

Conclusion

Mimix is inspired by important pioneers in computer science.[42] It addresses the unmet needs of 100,000,000 writers, researchers, teachers, students, innovators, inventors, and thinkers of all kinds. It is achievable with today's technology.

My goal in presenting this paper is to attract the people who can help me create and fund Mimix. Your contributions are warmly welcomed.

Thank you in advance,

-- David Bethune

Key West, Florida
August 7, 2018

[42] Those pioneers include so many whom I haven't mentioned but who strongly influenced this paper and inspire me to pursue this project. Firstly, Paul Graham for whom *On Mimix* is titled. I can't say enough about your writing and work. Thank you. I must also acknowledge Seymour Papert whose LOGO programming environment (really a Lisp) for the TI-99/4A changed my life as a child. Thank you to all those who work to advance computer science for the benefit of mankind.