Citations to Richard C. Waters' Research

At the end of 2023, I went to Google Scholar and looked at the citations to my papers as a proxy for the impact of my research. The Google Scholar data has the virtue of being comprehensive because it looks at a wide range of sources, but it is messy in several ways. For one thing, Google Scholar sometimes concludes that citations that are actually to the same paper are to different papers. Merging those together slightly reduced the total number of papers that have citations to them.

Another problem is that it does little if anything to filter out self-citations (cases where an author cites their own earlier paper in a later paper). For the data presented in Figure 1, I looked at all my papers and patents with 10 to 30 citations to them and eliminated the self-citations. A number of them had several self-citations and a couple that appeared to have 10 or more citations to them were revealed to have less than 10 non-self-citations. I then looked at a random sample of the rest of the items, which revealed that while a number had several self-citations, none had enough to make a significant difference.

When making Figure 1, I eliminated all the papers and patents with less than 10 citations to them. A number of those items are ones I think are quite good, but it is fair to say that, whether or not they are good, they probably had little, if any, impact.

The net effect of the above reveals me to have an h-index of 36 and an i10-index of 62 (both a bit less than shown on Google Scholar due to the data cleaning above). Not bad as researchers

Area	**Cites**	**Individual Papers and Patents Cited**
Research at MIT		
Mechanical Arm Control	60	60[36]
Lisp Pretty Printer	31	18[41] 13[46]
Lisp Utilities	30	19[47] 11[52]
Loop Analysis	217	149[37] 55[35] 13[34]
Series	79	79[50]
KBEmacs	680	289[43] 194[40] 76[39] 49[42] 41[44] 20[53] 11[38]
Programmer's Apprentice	870	744[20] 57[11] 24[15] 23[13] 22[17]
AI and SE	515	250[19] 161[16] 31[12] 30[23] 26[22] 17[18]
Program Translation	164	164[45]
Design Apprentice	32	32[70]
Requirements Apprentice	422	369[10] 53[9]
Reverse Engineering	152	77[67] 75[31]
Research at MERL		
Precursor to Diampnd Park	46	32[25] 14[24]
Audio Interactive Tutor	153	132[55] 21[54]
Spline	508	261[5] 182[4] 40[64] 13[63] 12[56]
Spline Patents	294	92[2] 73[3] 66[61] 45[73] 18[57]
Diamond Park	487	235[62] 126[65] 126[1]
Tree Insertion Grammar	363	171[29] 97[27] 79[28] 16[30]
History of Grpah Drawing	67	67[7]
Miscellaneous Patents	279	92[58] 59[72] 41[69] 31[68] 26[71] 18[60] 12[59]
Total Citations	5,449	To my 62 papers and patents with at least 10 citations each

Figure 1: Citations to publications authored by R.C.Waters as of 12/31/2023.

go, particularly considering that I have done very little research since becoming CEO in 1999 so we are talking about a research career of only 27 years, but certainly not great.

For Figure 1, I have grouped the 62 papers into 20 research areas. Each row shows the total number of citations to the papers and patents in an area followed by the several papers and patents in the area (each a reference in square brackets), each proceeded by the number of citations to it. The rows are listed in approximately chronological order. Each area is described below.

Mechanical Arm Control – The paper cited here is dated 1979, but it is only a minor update of an internal MIT note from March of 1973 [33].

When I applied to MIT for the 1972-73 year and didn't get in, I went to Harvard instead, but immediately took advantage of being able to take classes at MIT. I offered myself up to basically do anything for anyone and got connected to Professor Berthold Klaus Paul Horn who tasked me with looking into the dynamics of controlling a multi-link robot arm taking Coriolis forces into account. I produced [33] in short order, which I later found out was just in the nick of time to change people's minds and get me admitted to MIT for the 1973-74 year.

At MIT, I continued to work on robotics for a while but became disillusioned by how hard it was to get a robot arm to do anything interesting. However, the fast method of calculating things I developed for [33] was useful to many people and after I finished my PhD, Bertold encouraged me to re-release my earlier note in a more widely disseminated form, which I did in [36]. No one is more surprised than me at how many citations this paper has received. What I developed was a small thing, but it has been widely used.

Lisp Pretty Printer – Before coming to MIT, I only worked with programming languages like Fortran and PL1. At MIT, I met Lisp and fell in love with its ability to easily manipulate programs as data and through macros, to enable complex extensions to the language. Since then, whenever possible, I used Lisp for all my programming. I also did several things to improve the language and the Lisp programming environment.

An early thing I did was create a "Pretty Printer" that made it easy to print programs and data structures attractively given a specified width and number of lines. My tool rapidly became popular and was eventually adopted as part of the definition of Lisp [48].

Lisp Utilities – I implemented several utilities I used to facilitate program development in Lisp, some of which were taken up by others. The one I personally felt was most helpful supported the automatic regression testing of programs [51]. It did not get much uptake, and therefore is not in Figure 1 but I used it to help maintain everything I wrote.

Loop Analysis – The heart of my PhD thesis was converting Fortran programs into a dataflow diagram representation where they could be manipulated in a syntax-free manner. The key novelty in my work was converting almost all loops into an expressional form. My key insight was realizing that while the typical program of the time appeared to be full of labels and branches that could be arbitrarily hard to reason about, the full power of labels and branches was almost never used. Rather, they were only used in very clichéd ways.

The papers in this area describe how this analysis was done. These methods were a key basis for much of the rest of my work at MIT, see below.

Series – The most direct use of my insight about loops was the construction of an extension to Lisp, allowing the vast majority of loops to be written in an expressional notation. The central concept is of a "Series," which is a series of values spread out in time rather than space. Loops are written as the composition of fragments operating on Series. I found Series intensely useful and used it in all my programming. For example in the more than a thousand pages of code for KBEmacs, there is (if I remember correctly) only one loop that looks like a loop. Every other loop is written using Series.

In an attempt to gain more interest from people, I evolved the system over many years from an early pragmatically useful version called LetS [14] to a final more aesthetically attractive version that could be run either interpreted or compiled, but was slightly less pragmatically useful. A key goal was to get Series incorporated into Lisp as my Prett Printer had been and I came very close to doing that.

Unfortunately, a researcher named Crispin Perdue came up with a concept that at the interpreted level was superficially different, but equivalent to Series. In the end, our two approaches ended up as only appendices [8, 49] to the Lisp manual rather than either being fully included. I was annoyed at that, because I felt that the automatic conversion of expressional notation into highly efficient loops, which only worked for Series, made Series vastly superior.

In [14] I showed that many languages had concepts that moved in the direction of Series but none did so comprehensively. In the present, the situation remains the same. For example, it has been interesting to see how Series-like expressions in Microsoft Excel have gradually grown more mainstream, no longer needing a special command to enable them, while still being far from comprehensive.

If I had not been committed to working with Chuck Rich toward the Programmers Apprentice, I might have focused on making LetS or Series into a standard feature of many programming languages in the way others got object orientation into many languages. However, that would have been a lot of effort. In Lisp, LetS and Series can be implemented as macro packages. In other languages, you would have to modify the compiler, which is a bigger challenge and requires buy-in from many people.

The biggest regret of my career has been the languishing of Series. It would not be too late to try to revive it, but it would take a great deal of time, effort and probably money.

KBEmacs – The Knowledge-Based Editor in Emacs (KBEmacs) was my vehicle for demonstrating my work toward the Programmers Apprentice. Using it, you could edit a program both textually and using higher level commends based on the program's logical structure. In particular, you could insert chunks of code from a library of clichés. To me, there were three key parts of this: (1) the main operations occurred at the level of a dataflow representation of the program, rather than at a textual level, which was greatly facilitated because loops were also expressions and control flow was relegated to a minor issue (2) but because KBEmacs could automatically convert both ways between text and dataflow you could still freely edit at a textual level using Emacs, and (3) the use of clichés made the creation of programs quicker.

Given that creating KBEmacs was my primary effort during the time I worked at MIT after my PhD, it is appropriate that it is my most cited individual work from that time. Given that KBEmacs never reached the level of a prototype anybody else could use, it is also appropriate that its influence rapidly waned after I stopped working on it. I do not believe anything today could be said to be based on it. This I regret, but not enormously, for the reason below.

A center of activity in the world of programing automation at the time was program transformation systems. These were text-to-text systems that could do interesting things and could make use of clichés, but they were inherently hobbled by operating on text (or parse trees) with all its syntactic idiosyncrasies. It was clear to Chuck and me that what they should be doing is taking a KBEmacs-like approach of fundamentally operating on dataflow diagrams. We felt that a logical way to go was to essentially merge KBEmacs and the program transformation approach to produce a fundamentally more powerful system than the then current transformation systems.

We talked a lot about potentially starting a company to make this idea a reality. But in the end, we decided that creating a startup was not for us. We wanted more from life that just working 24x7 to make a startup work and knew that startups where the founders do not pour all of themselves into them for years fail. Because I still feel this is sound reasoning, I do not regret that we did not go the startup route. Because I think that that was the only way KBEmacs could have turned into something significant, I do not much regret the lack of influence of KBEmacs.

Programmer's Apprentice – While I worked on KBEmacs, Chuck worked on a different aspect of the Programmer's Apprentice we were seeking: creating a formalized dataflow representation he called a "Plan" and a reasoning system he called a "Plan Calculus" for manipulating programs in logically rigorous ways. Our long-term goal was to use have a system like KBEmacs that used his Plans as its internal representation rather than bare-bones dataflow diagrams and used the Plan Calculus to manipulate them, while still supporting textual editing when a programmer desired. Unfortunately, developing the Plan Calculus system proved to be much harder than Chuck initially thought and it never reached the point where it could be incorporated into KBEmacs.

Nevertheless, while we never got near to being able to demonstrate what we hoped the Programmer's Apprentice would be, we had a clear view in mind and wrote a number of papers about the concept and our progress toward it.[1] These papers are my most cited work and I hope that the ideas in them influenced many researchers. However, I am now so long out of that world that I have no idea to what extent our ideas have survived the test of time.

AI and SE – Over the years, Chuck and I also wrote a number of thought pieces on the general topic of how Artificial Intelligence could be used to support Software Engineering and edited a volume of papers in this area written by others [16]. I was the primary author of "Automatic Programming: Myths and Prospects" [19] and consider it the best written paper I ever crafted. Given it's large number of citations, I think it was recognized as insightful.

Program Translation – On my own, and with my students, I worked on several spin offs from KBEmacs. One was essentially just an observation. Since everything important in KBEmacs operated at the dataflow diagram level separated from the syntax of any particular programming language, and I had made KBEmacs work on the new language Ada to show that it was not something overly tied to Lisp, KBEmacs could trivial translate a program from Lisp to Ada and vice versa.

Design Apprentice – My Student Yang Meng Tan and I wrote an idea piece on how one might apply Programmer's-Apprentice-like ideas to program design decisions. He switched to Chuck as his primary advisor and his PhD thesis [32] related more to Chuck's work than mine.

Requirements Apprentice – My student Howard Reubenstein and I wrote some idea pieces on how Programmer's-Apprentice-like ideas could be used for requirements acquisition. His PhD thesis [26] demonstrated some steps in that direction.

Reverse Engineering – Reverse engineering of programs was a hot topic in the early 1990s. I teamed up with Eliot Chikofsky, who had a lot of experience running conferences, to put together

[1]While it seems not to have been cited by anyone, the definitive description of our progress toward the Programmer's Apprentice is our 1990 book [21].

the first workshop focused on reverse engineering [66]. The conference did not happen until after I began work at Mitsubishi Electric Research Labs (MERL), but the effort was born while I was at MIT. Two publications related to the conference have gotten a significant number of citations.

Precursor to Diampnd Park – When Chuck and I got to MERL as two of the first three american employees, our research changed completely. I vividly remember us being told that we could basically do whatever we wanted as long as it was not more work on Software Engineering tools. (Mitsubishi Electric could not imagine selling a Software Engineering tool.)

We gathered most of the people at MERL together to create a demonstration system featuring a couple of people on a local network interacting with an artificial agent. Computer graphics was used to display the people as simple avatars in a Spartan virtual environment with the agent as a bipedal robot in the environment. The people could move themselves around with a joystick. The motion of the robot was controlled by a dynamic simulation provided by Marc Rabert. The people could speak to each other using audio over the local network. They could talk to the robot using speech input (from Dragon Systems) and the robot could talk back using speech output (Dectalk). Simple language understanding of a small set of commands allowed the robot to take actions and generate reasonable outputs. Background noises were simulated including foot falls. Getting all this to work together was at the cutting edge for its time.

Audio Interactive Tutor – While I was working on the demo above, I had an idea for what I called The Audio Interactive Tutor (TAIT). I had used language learning tapes form Pimsler several times and found them useful, but was frustrated by the fact that the only judge of whether what I was responding was correct was me and I ended up practicing a lot of things I knew well in order to practice those that I didn't. Given the very restricted set of correct responses at any one time and the equally limited set of likely mistakes, it should be fully practical for a system to recognize whether a person was making a correct response and comment on common errors. In addition, it could keep track of what a person was and was not learning and automatically set up an efficient practice schedule.

I presented this at a language teaching conference and it was well received, but afterwords someone came up to me and said that he had heard multiple AI types talk about how AI could help language learning but nothing every came from it because there was never any follow through. Unfortunately for TAIT, there was indeed no follow through. This is another thing I regret. I am sure it could straightforwardly have been done, but I was busy with other things. Years later, I tried to interest Pimsler in the idea, but nothing came of that.

At MIT, I never filed a patent because MIT was not actively patenting at the time and for years, you couldn't patent software anyway. However, MERL had a focus on patenting, so I filed a patent on TAIT [55]. It is interesting to note that 86% of the citations to TAIT are to this patent. This gives me hope that the ideas in TAIT were used in later research.

Spline – Enthused with our success creating the multi-person agent interaction demo above we resolved to turn it into a real useful system. This required the creation of a sufficiently powerful substrate to support the communication between users. The closest thing to what we wanted was the military simulation system SIMNET; however, it had several deficiencies in comparison to what we wanted. I set out to create a better system, which I called the Scalable PLatform for INteractive Environments (SPLINE), and then just Spline (not an acronym), and in the end the Interactive Sharing Transfer Protocol (ISTP).

Spline supported several things that no system had been able to simultaneously support before. like SIMNET it could support a large environment with a large number of users. However, going beyond SIMNET, it supported those users talking to each other via audio in very near real time, it allowed the environment to be changed during operation (as opposed to between sessions) allowing evolution during non-stop operation, and it allowed any user to make changes.

Because we could never interest Mitsubishi Electric in using or selling Spline, effort on it eventually ended at MERL and we did no further work on Distributed Virtual Reality systems. However, there is no doubt that Spline was influential and key aspects of the way it worked can be seen in the multiplayer role playing games and virtual environments of today. All told, the design of Spline is probably my most clearly impactful research.

Spline Patents – I have separated out here the patents for Spline. The large number of citations to them attest to the number of patents by others either building on the Spline patents or trying to work around them. This is a further indication of the impact of Spline.

Diamond Park – We created Diamond Park (for which [62] is the definitive reference) to show what Spline could do. I led the Diamond Park effort, but my leadership was mostly managerial since other people did almost all the work. My only hands-on effort was being the director and cinematographer for the video we made showing off Diamond Park. Other than as an illustration of Spline and an advertisement for Mitsubishi Electric at COMDEX in 1995, Diamond Park had little if any impact, since it was never released.

Tree Insertion Grammar – In parallel with the above, I had a multi-year collaboration with Yves Schabes who was a computational linguist. Linguistics was my minor for my PhD and I have always continued interest in it. One day I ran into Yves trying to prove something about a particular class of grammer he had in mind and I contributed the proof he wanted and then later constructed a much simpler proof. This led to a series of papers, which received some notice in the linguistic community. I have no idea whether they had any real impact, but it was a fun sideline for me with Yves doing 95% of the work.

History of Grpah Drawing – A key person at MERL for many years was Joe Marks. I always enjoyed working with him because he bubbled over with intersting ideas and enthusiasm. He was involved with a project to automatically draw graphs in human readable ways and had the idea to investigate how long graphical representations had been used. He enlisted a MERL intern from Albania named Eriola Kruja to lead the effort and roped me into the project that lead to this paper. I did at most 1% of the work, but it was again a fun sideline.

Miscellaneous Patents – Even after I was in management, I (sometimes by myself, but more often with others) continued to have ideas for patents. Some were granted and some were not. Some got no notice and the ones shown here got significant numbers of citations. It is interesting to me that some that were only published and not granted [59, 60, 72] were nevertheless cited a fair bit. To me this indicates that while rejected by the Patent Office, people agreed with me that they were interesting ideas.

References

[1] Anderson D.B., Barrus J.W., Howard J., Rich C., Shen C. & Waters R.C., “Building Multiuser Interactive Multimedia Environments at MERL,” *IEEE MultiMedia*, 2(4):77–82, Winter 1995.

[2] Anderson D.B. & Waters R.C., *System For Sending Small Positive Data Notification Messages Over a Network To Indicate That a Recipient Node Should Obtain a Particular Version of a Particular Data Item*, US patent 5,842,216, disclosed 1/96, filed 5/96, granted 11/98.

[3] Barrus J.W. & Waters R.C., *System For Designing a Virtual Environment Utilizing Locales*, US patent 5,736,990, disclosed 6/95, filed 8/95, granted 4/98.

[4] Barrus J.W., Waters R.C. & Anderson D.B., “Locales and Beacons: Efficient and Precise Support for Large Multi-User Virtual Environments,” *Proc. IEEE Virtual Reality Annual International Symposium*, (Santa Clara CA, March 1996), 204–213, IEEE Computer Society Press, Los Alamitos CA, 1996.

[5] Barrus J.W., Waters R.C. & Anderson D.B., “Locales: Supporting Large Multiuser Virtual Environments,” *IEEE Computer Graphics and Applications*, 16(6):50–57, November 1996.

[6] Cohen H. & Weil G.R., “Tasks of Emotional Development: A projective test for children and adolescents,” Lexington books in psychology, D.C. Heath, 1971.

[7] Kruja E., Marks J., Blair A. & Waters R.C., “A Short Note on the History of Graph Drawing,” *proc. 9th International Symposium on Graph Drawing*, (Vienna Austria, September 2001), 272–286, Springer-Verlag, Berlin, 2002.

[8] Perdue C. & Waters R.C., “Generators and Gatherers,” Appendix B in *Common Lisp: the Language*, Second Edition, 956–959, Steele G.L.Jr., Digital Press, Burlington MA, 1990.

[9] Reubenstein H.B. & Waters R.C., “The Requirements Apprentice: An Initial Scenario,” *Proc. Fifth International Workshop on Software Specification and Design*, (Pittsburgh PA, May 1989), 211–218, IEEE Computer Society Press, Washington DC, 1989.

[10] Reubenstein H.B. & Waters R.C., “The Requirements Apprentice: Automated Assistance for Requirements Acquisition,” *IEEE Transactions on Software Engineering*, 17(3):226–240, March 1991.

[11] Rich C., Shrobe H.E. & Waters R.C., “An Overview of the Programmer’s Apprentice,” *Proc. Sixth International Joint Conference on Artificial Intelligence*, (Tokyo Japan, August 1979), 2:827–828, Stanford University Computer Science Dept., Stanford CA, 1979.

[12] Rich C. & Waters R.C., *Abstraction, Inspection and Debugging in Programming*, MIT/AIM-634, June 1981.

[13] Rich C. & Waters R.C., “Computer Aided Evolutionary Design for Software Engineering,” *ACM SIGART Newsletter*, (76):14–15, April 1981.

[14] Waters R.C., *LetS: An Expressional Loop Notation*, MIT/AIM-680, October 1982. MIT/AIM-680a, February 1983.

[15] Rich C. & Waters R.C., “Formalizing Reusable Software Components,” *Proc. Workshop on Reusability in Programming*, (Newport RI, September 1983), 152–159, ITT Corporation, Hartford CN, 1983.

[16] Rich C. & Waters R.C. (editors), *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann, Los Altos CA, 1986.

[17] Rich C. & Waters R.C., *Formalizing Reusable Software Components in the Programmer’s Apprentice*, MIT/AIM-954, February 1987.

[18] Rich C. & Waters R.C., “Artificial Intelligence and Software Engineering,” in *AI in the 1980’s and Beyond: An MIT Survey*, 109–154, Grimson W.E.L. & Patil R.S. (editors), MIT Press, Cambridge MA 1987.

[19] Rich C. & Waters R.C., “Automatic Programming: Myths and Prospects,” *IEEE Computer*, 21(8):40–51, August 1988.

[20] Rich C. & Waters R.C., “The Programmer’s Apprentice Project: A Research Overview,” *IEEE Computer*, 21(11):10–25, November 1988; and its reprinting as Rich C. & Waters R.C., “The Programmer’s Apprentice ,” Chapter 7 in *Artificial Intelligence at MIT: Expanding Frontiers*, Volume 1, 166–195, Winston P.H. with Shellard S.A. (editors), MIT Press, Cambridge MA, 1990.

[21] Rich C. & Waters R.C., *The Programmer’s Apprentice*, Addison–Wesley, Reading MA and ACM Press, Baltimore MD, 1990.

[22] Rich C. & Waters R.C., “Knowledge Intensive Software Engineering Tools,” *IEEE Transactions on Knowledge and Data Engineering*, 4(5):424–430, October 1992.

[23] Rich C. & Waters R.C., “Approaches to Automatic Programming,” in *Advances in Computers*, Volume 37, 1–57, Yovits M.C. (editor), Academic Press, San Diego CA, 1993.

[24] Rich C., Waters R.C., Schabes Y., Freeman W.T., Torrance M.C., Golding A.R. & Roth M., “An Animated On-Line Community with Artificial Agents,” *IEEE MultiMedia*, 1(4):32–42, Winter 1994.

[25] Rich C., Waters R.C., Strohecker C., Schabes Y., Freeman W.T., Torrance M.C., Golding A.R. & Roth M., “Demonstration of an Interactive Multimedia Environment,” *IEEE Computer*, 27(12):15–22, December 1994.

[26] Reubenstein, Howard B., *Automated Acquisition of Evolving Informal Descriptions*, MIT/AI/TR-1205, June 1990.

[27] Schabes Y. & Waters R.C., “Stochastic Lexicalized Context-Free Grammar,” *Proc. Third International Workshop On Parsing Technology*, (Tilburg, The Netherlands, August 1993), 257–266, Association for Computational Linguistics Special Interest Group on Parsing, 1993.

[28] Schabes Y. & Waters R.C., *System For Decreasing the Time Required To Parse a Sentence*, US patent 5,475,588, disclosed 6/93, filed 6/93, granted 12/95.

[29] Schabes Y. & Waters R.C., "Tree Insertion Grammar: A Cubic-Time Parsable Formalism That Lexicalizes Context-Free Grammar Without Changing the Trees Produced," *Computational Linguistics*, 21(4):479–513, December 1995.

[30] Schabes Y. & Waters R.C., "Stochastic Lexicalized Tree-Insertion Grammar," in *Recent Advances in Parsing Technology*, 281–294, Bunt H. & Tomita M. (editors), Kluwer Academic Publishers, Norwell MA, 1996.

[31] Selfridge P.G., Waters R.C. & Chikofsky E.J., "Challenges to the Field of Reverse Engineering," *Proc. Working Conference on Reverse Engineering*, (Baltimore MD, May 1993), 144–150, IEEE Computer Society Press, Los Alamitos CA, 1993.

[32] Tan, Yang Meng, *Formal Specification Techniques for Promoting Software Modularity, Enhancing Documentation, and Testing Specifications*, MIT/LCS/TR-619, June 1994. (Later publised as a book *Formal Specification Techniques for Engineering Modular C Programs*, Kluwer Academic Publishers, Boston MA, 1996.)

[33] Waters R.C., *Mechanical Arm Control*, MIT vision flash 42, March 1973.

[34] Waters R.C., *A System for Understanding Mathematical Fortran Programs*, MIT/AIM-368, May 1976. (Revised August 1976.)

[35] Waters R.C., *Automatic Analysis of the Logical Structure of Programs*, MIT/AI/TR-492, December 1978.

[36] Waters R.C., *Mechanical Arm Control*, MIT/AIM-549, October 1979. (Updated version of [33].)

[37] Waters R.C., "A Method for Analyzing Loop Programs," *IEEE Transactions on Software Engineering*, 5(3):237–247, May 1979.

[38] Waters R.C., "A Knowledge Based Program Editor," *Proc. Seventh International Joint Conference on Artificial Intelligence*, (Vancouver BC Canada, August 1981) 2:920–926, American Association for Artificial intelligence, Menlo Park CA, 1981.

[39] Waters R.C., "Program Editors Should Not Abandon Text Oriented Commands," *ACM SIGPLAN Notices*, 17(7):39–46, July 1982.

[40] Waters R.C., "The Programmer's Apprentice: Knowledge Based Program Editing," *IEEE Transactions on Software Engineering*, 8(1):1–12, January 1982.

[41] Waters R.C., "User Format Control in a Lisp Prettyprinter," *ACM Transactions on Programming Languages and Systems*, 5(4):513–531, October 1983.

[42] Waters R.C., *KBEmacs: A Step Toward the Programmer's Apprentice*, MIT/AI/TR-753, May 1985.

[43] Waters R.C., “The Programmer’s Apprentice: A Session With KBEmacs,” *IEEE Transactions on Software Engineering*, 11(11):1296–1320, November 1985.

[44] Waters R.C., “KBEmacs: Where’s the AI?,” *AI Magazine*, 7(1):47–56, Spring 1986. (Reprinted in *Proc. Second Kansas Conference: Knowledge-Based Software Development*, (Manhattan KS, October 1986), Kansas State University, Manhattan KS, 1986.)

[45] Waters R.C., “Program Translation via Abstraction and Reimplementation,” *IEEE Transactions on Software Engineering*, 14(8):1207–1228, August 1988.

[46] Waters R.C., *XP: A Common Lisp Pretty Printing System*, MIT/AIM-1102, March 1989. MIT/AIM-1102a, September 1989.

[47] Waters R.C., “Automated Software Management Based on Structural Models,” *Software–Practice & Experience*, 19(10):931–955, October 1989.

[48] Waters R.C., “Pretty Printing,” Chapter 27 in *Common Lisp: the Language*, Second Edition, 748–769, Steele G.L.Jr., Digital Press, Burlington MA, 1990.

[49] Waters R.C., “Series,” Appendix A in *Common Lisp: the Language*, Second Edition, 923–955, Steele G.L.Jr., Digital Press, Burlington MA, 1990.

[50] Waters R.C., “Automatic Transformation of Series Expressions into Loops,” *ACM Transactions on Programming Languages and Systems*, 13(1):52–98, January 1991.

[51] Waters R.C., “Supporting the Regression Testing of Lisp Programs,” *ACM Lisp Pointers*, 4(2):47–53, June 1991.

[52] Waters R.C., “System Validation via Constraint Modeling,” *ACM SIGPLAN Notices*, 26(8):27–36, August 1991.

[53] Waters R.C., “Cliché-Based Program Editors,” *ACM Transactions on Programming Languages and Systems*, 16(1):102–150, January 1994.

[54] Waters R.C., “The Audio Interactive Tutor,” *Computer Assisted Language Learning*, 8(4):325–354, December 1995; and the early version in Waters R.C., *The Audio Interactive Tutor*, MERL TR 94-04, April 1994.

[55] Waters R.C., *Audio Interactive Tutor*, US patent 5,540,589, disclosed 3/93, filed 4/94, granted 7/96.

[56] Waters R.C., *Time Synchronization In Spline*, MERL TR 96-09, April 1996.

[57] Waters R.C., *Beacons for Locales*, US Patent 5,920,862, disclosed 7/95, filed 11/95, granted 7/99.

[58] Waters R.C., *Situation Awareness System*, US Patent 6,396,535, disclosed 12/98, filed 2/99, granted 5/02.

[59] Waters R.C., *Invisible Beam Pointer System*, filed 5/02, not granted. Published 2003/0210230A1.

[60] Waters R.C, *Identifying Faces from Multiple Images Acquired from Widely Separated Viewpoints*, filed 9/04. Not granted. Published 2006/0056667.

[61] Waters R.C. & Anderson D.B., *System For the Reliable, Fast, Low-Latency Communication of Object State Updates Over a Computer Network by Combining Lossy and Lossless Communications*, US Patent 6,006,254, disclosed 7/97, filed 8/97, granted 12/99.

[62] Waters R.C, Anderson D.B., Barrus J.W., Brogan D.C., Casey M.A., McKeown S.G., Nitta T., Sterns I.B. & Yerazunis W.S., "Diamond Park and Spline: Social Virtual Reality with 3D Animation, Spoken Interaction, and Runtime Extendability," *Presence: Teleoperators and Virtual Environments*, 6(4):461–480, August 1997.

[63] Waters R.C., Anderson D.B. & Schwenke D.L., *The Interactive Sharing Transfer Protocol Version 1.0*, MERL TR 97-10, October 1997.

[64] Waters R.C., Anderson D.B. & Schwenke D.L., "Design of the Interactive Sharing Transfer Protocol", *Postproc. WET ICE '97 – IEEE Sixth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, (MIT, June 1997), 140–147, IEEE Computer Society Press, Los Alamitos CA, 1997.

[65] Waters R.C. & Barrus J.W., "The Rise of Shared Virtual Environments," *IEEE Spectrum*, 34(3):20-25, March 1997.

[66] Waters R.C. & Chikofsky E.J. (editors), *Proc. Working Conference on Reverse Engineering*, (Baltimore MD, May 1993), IEEE Computer Society Press, Los Alamitos CA, 1993.

[67] Waters R.C. & Chikofsky E.J. (editors), *Special Section on Reverse Engineering, Communications of the ACM*, 37(5):22–93, May 1994.

[68] Waters R.C., Jones T.R., Perry R. & Seiler L., *Method and Apparatus For Multi-Phase Rendering*, US Patent 6,359,619, disclosed 6/98, filed 6/99, granted 3/02.

[69] Waters R.C. & Russell F., *Video Camera Controlled Surround Sound*, US Patent 6,741,273, disclosed 6/99, filed 7/99, granted 5/04.

[70] Waters R.C. & Tan Y.M., "Toward a Design Apprentice: Supporting Reuse and Evolution in Software Design," *ACM SIGSOFT Software Engineering Notes*, 16(2):33–44, April 1991.

[71] Waters R.C., Thornton J., Raskar R., Kato M. & Dietz P., *Self-Correcting Rear Projection Television*, US Patent 7,227,592, Disclosed ?/03, filed 9/03, granted 6/07.

[72] Wittenburg K., Waters R.C. & Smaragdis P., *Cellular Telephone Based Surveillance System*, Disclosed 5/04, filed 5/04, not granted. Application published 2005/0255826.

[73] Yerazunis W.S. & Waters R.C., *Method for Smooth Motion in a Distributed Virtual Environment*, US patent 5,793,382, disclosed 12/95, filed 6/96, granted 8/98.