In 1965, four undergraduates at the University of Waterloo wrote Watfor, a fast student-oriented Fortran compiler for the school’s IBM 7040, largely because the available Fortran compiler was slow and offered weak diagnostic and debugging tools. This article describes the birth and evolution of the Watfor family and explores how it fits into the University of Waterloo’s unique-within-Canada cooperative education program and pedagogical philosophy.

High-level computing programming languages were invented in the late 1950s, and as they evolved, it was perhaps inevitable that debates would arise regarding the relative merits of these new tools. For a variety of reasons, different groups came to prefer different languages or implementations, although as David Nofre recently pointed out, these debates evolved subtly and could be easily influenced by nontechnical factors and desires.1

At the same time, the widespread and explosive increase in interest in modern computers created a demand for computer programmers that easily outstripped supply. University and college graduates represented one logical source of young talent, but virtually every school faced the same problem: programming languages ill-designed for the mass instruction of potential computer scientists, engineers, scientists, and mathematicians. Although IBM and other computer manufacturers generally provided their customers with compilers to translate source code written by programmers into computer-readable object code, the compilers were often unsuitable for novice programmers. Typically, the compilers produced object code that ran quickly and efficiently, but the compiling process itself could be so slow that only a few students could be accommodated per day, not dozens or even hundreds as some universities wished. And when it came to the inevitable debugging process, beginner programmers did not have much help because most compilers provided little useful diagnostic information.2

Ultimately, many university computing centers would create their own student-oriented compilers and programming languages, a clear example of technological users transforming an artifact, actively participating in the development, and shaping the Fortran language toward a more educationally friendly version.3

Within this context, the University of Waterloo developed Watfor for their IBM 7040 mainframe in 1965 and 360 Watfor for their System/360 in 1967. The latter, eventually known as Watfiv, proved to be widely, even wildly, popular around the globe and led to an entire family of University of Waterloo-created educational software packages that continued into the 1980s.

At the University of Waterloo, these events were crucial in helping establish the identity and reputation of the university, which quickly became known as a center of innovation and leading-edge computer education with strong ties to industry. Watfor and Watfiv helped create and reinforce this image and helped elevate the school among the older, more established universities in Canada.

The Unconventional University of Waterloo

The University of Waterloo was founded in 1957, initially as a technical offshoot of the existing arts-focused Waterloo College. Located in Waterloo, Ontario, Canada, about
an hour west of Toronto, its most dominant characteristic was an unconventional cooperative education program in which students would alternate their academic terms with on-the-job, apprentice-like work terms. The co-op program was a product of the Waterloo Plan, a mid-1950s proposal by local civic and industrial leaders to create a new kind of university that would unite industry and academia, effectively joining theory and practice by sharing in student education. This radical notion was dismissed by every other Canadian school, but supported by political and business leaders who feared that the Canadian technical workforce was falling behind that of the Soviet Union. A small cohort of just 75 co-op engineering students arrived in July 1957, but within 10 years, nearly 6,000 students were enrolled in four faculties. On what had been farmland just a few years earlier, a new campus grew with 15 academic buildings that contained offices and lecture halls, libraries and laboratories, completed or under construction.

Much of the University of Waterloo's eventual reputation can be traced to key faculty members hired in that first decade. Most relevant to this story is Ralph Stanton, who headed the mathematics department, having taught previously at the University of Toronto, one of Canada's largest and best-regarded universities. Stanton was a well-connected figure and respected in industry. In 1959, he recruited J. Wesley Graham to Waterloo. Graham had been one of his top students at Toronto and had finished a master's degree there before joining IBM Canada, where he rose to manager of its Applied Sciences Division.

At Waterloo, Graham helped establish a foundation for digital computing. He taught numerical analysis and served on the university's first computing committee, which set the agenda for computing and selected Waterloo's first computer, a relatively small, inexpensive IBM 1620. IBM installed its most basic model in April 1961 as a rental, but Waterloo slowly upgraded it over the following years and purchased it outright in 1963. Use of the machine was initially restricted to graduate students and faculty research, but undergraduate Fortran-related courses were offered in 1962.

University Computing in the Early 1960s

Many organizations struggled in the early 1960s to integrate and understand modern computing, and universities faced their own unique challenges. Educational and research demands, for instance, differed significantly from those of the financial and manufacturing worlds. Workshops, conferences, and articles were devoted to exploring the differences between these situations and analyzing the specific problems found in university computing.

Often, the first problem was to find a logical place to house a computer within the existing university structures. Engineering, science, and mathematics departments might be the only major computer users, but should they also be responsible for operating the computer on behalf of the entire campus? A common, but not universal, view was that academic departments should rightly focus their energies on research and teaching, not the more prosaic and conservative duties of computer maintenance and system programming. Thus, computing centers were often established as or eventually evolved into nonacademic service units.

Still, finding computing staff could be difficult and the line between academic and nonacademic activities could be blurred or nonexistent. Computing center directors were often double-duty faculty members, and they frequently hired their staff programmers from the ranks of their students. In 1963, 12 out of 15 Canadian university computer centers were under the direction of a professor of engineering, physics, or mathematics. At the University of Waterloo, Graham served simultaneously as head of computer science and director of the Computing Center throughout the mid-1960s. Despite the intention of a strong division between operations and academics, having the two closely linked meant that pedagogical needs could be identified and met more easily.

A primary concern of virtually every computing center of the era was throughput, often measured in the number of jobs processed per hour. A single job consisted of a single task, measured from start to finish, although it might involve several programs or peripherals. A typical student job consisted of a short Fortran source code listing submitted on a deck of punched cards. A computer system processed such a job by first loading a Fortran compiler into memory, followed by the source listing. The compiler would read the human-readable source code line by line and translate it into a machine-readable object code file. The system would
then load and run a linker to combine the object code with any necessary software libraries, creating a program ready for execution on the computer. The system would then execute the final program, using the data from the original source deck and output the results. An error at any stage would likely terminate the job immediately. Once the job finished, the computer would begin anew with the next job in the queue.

While most student jobs were short—perhaps 50 lines of source code or less—the volume was high. In the early 1960s, at a university computing center using a standard manufacturer-supplied Fortran compiler, processing several hundred such jobs could take hours or even days. For instance, on the IBM 1620—by far the most common computer at North American academic institutions in 1963—compiling a “zero-length” program that deliberately did nothing could take 10 minutes.11 The same program on the much faster IBM 7040 still took about a minute.12 On the vastly more powerful IBM System 360 Model 65 from the mid-1960s, a similar zero-length program compiled and run 30 times—simulating a class of students—still took at least 10 minutes of computer time.13 A medium-sized campus in the early 1960s might need to turn around 300 to 500 undergraduate jobs per day and still have computer time left over for research by professors or graduate students plus systems and maintenance tasks by the computing center. On a stock IBM 1620 with a standard IBM-supplied Fortran compiler, this was simply impossible. At least one of the three—teaching, research, and maintenance—would have to give way.14

Further complicating matters, two characteristics of most undergraduate programmers are worth noting. First, as a group, students were more likely to make mistakes, and with their inexperience, it might take dozens of attempts to remove all the errors from their code. Fortran was a relatively straightforward language to teach and learn, but some instructions, such as the FORMAT statement, were notoriously difficult to master. Compilers could contain idiosyncratic features or bugs that would also elude novices, and almost no compilers provided meaningful diagnostic information if there were any mistakes in the code. Fixing even a single bug could take days with a slow compiler. In these circumstances, the bulk of a university’s computer time was wasted compiling programs.

The second characteristic of typical student programmers was that they only needed to run their final programs once.

For such jobs, the big headache is getting them debugged and working right. As soon as you get the right answers once, the hell with production—you’re through with it and you start the next one. In other words, many short compilations and very few executions.14

After a particular assignment or problem was solved, a student would move on, and there was no need to keep a copy of the object code because it would never be needed again. Fortran compilers were normally optimized to minimize execution time of the final object code, but for universities with high numbers of novice programmers with short programs that would run just once, a more critical optimization was to minimize the compilation time.

The lack of a fast student-oriented compiler with useful diagnostics was an inescapable problem for the professors and computing centers charged with mass instruction of undergraduates.

One early compromise was a scaled-down “pidgin Fortran,” known as Gotran.15 Created by IBM’s Applied Programming Division in 1960 specifically for the IBM 1620, it was not technically a compiler, but a load-and-go interpreter based on Fortran I. During operation, it translated source code directly, assembling the complete executable line by line, without producing an object code file or unloading the interpreter from memory. Once it finished translating the final statement, the computer could execute the program immediately, hence the term “load-and-go.” However, because it needed almost all the standard 20,000-word memory to shoehorn the interpreter and an executable into memory at the same time, it could only offer a reduced and simplified language set. It did not include the FORMAT statement, for example, and compensated by automatically determining the appropriate output based on the variable type.

Gotran proved to be a good option for beginners. It was faster and easier to learn than the standard Fortran and could still be run on a minimally equipped IBM 1620.16 A simple program that solved one quadratic polynomial took as much as 35 minutes with the standard Fortran compiler, but it only required five minutes with Gotran.17 Nonetheless, improvements could be made.
Gotran’s limited language set inspired the University of Wisconsin in the US to create a full Fortran load-and-go compiler for their upgraded 40,000-word IBM 1620. Known as Forgo, the fast Fortran II compiler also provided constructive diagnostics to the programmer. Intended primarily for students, the feature offered useful accounts of possible errors in the code, making it easier to find and correct mistakes.

For universities and colleges with similar hardware and teaching demands, including the University of Waterloo, Forgo was a popular choice. Although Forgo was available through the official IBM 1620 Users Group, one anecdote has it that a copy of the Forgo source code was “borrowed” by a Waterloo student visiting the University of Wisconsin sometime in 1962.

Wes Graham quickly recognized Forgo’s value and adopted it immediately in plans for a new series of formal programming courses at the undergraduate level. In the first full academic year, 1962–1963, about 300 students took the courses; by 1964–1965 more than 800 students enrolled. At its peak at Waterloo, Forgo was being used to process as many as 2,000 student jobs per day. Graham happily began promoting Forgo as a solution to the throughput problem, favorably comparing the inexpensive IBM 1620 to a much more costly IBM 7090. The dramatic speed improvements made mass programming instruction convenient and economical, which were essential to Graham if computer education was also to be meaningful:

Learning to use a computer for engineering purposes involves more than sitting in on a FORTRAN course and trying one or two problems on the computer. Experience is essential. ... over the course of an entire undergraduate programme.

The IBM 1620 was an entry-level computer for many schools. As enrollment and research demands increased during the early and mid-1960s, many colleges and universities upgraded. In Canada, most schools, including the University of Waterloo, chose the IBM 7040 as a successor. Although the IBM 7090 was more powerful and desirable, it was too expensive for Canadian universities, with the exception of the University of Toronto.

Responding to user needs and wishes, IBM continued to improve Fortran, eliminating machine-specific dependencies, extending its capabilities, and formalizing its peculiarities. In late 1962, IBM released Fortran IV, initially for the Stretch supercomputer and later for the 7090 and 7040 line of mainframes. Although it was intended to be a fast compiler that would also hold to the original ideal of efficient object code, not every user was satisfied. Despite IBM’s improvements, Fortran IV was not well suited for students or beginner programmers; schools that had upgraded their hardware began again their quest for more appropriate tools. One alternative, Quiktran, an interactive time-sharing version of Fortran from IBM, might have suited, but it required expensive additional hardware investments. Digitek Corporation’s commercial Fortran compilers were said to be much faster than IBM’s compilers, but they were also more expensive at $3,000 to $5,000 per installation and $1,600 per month thereafter. By the summer of 1964, many academic computing centers had launched their own projects to develop a Forgo-like Fortran compiler for their students (see Table 1).

The University of Waterloo installed its IBM 7040 in October 1964, but Graham knew of no solutions to the compiler problems...
throughput problem for the new computer. Initially, access to the IBM 7040 was restricted to faculty and graduate students, who were soon logging more than 300 hours a month on the machine, much of it lost to compile time.27

**Watfor: Waterloo Fortran**

Concerned with poor utilization, Graham decided that Waterloo would write its own student compiler. On 15 April 1965, a Working Group on Writing a Fortran IV Compiler was established, led by lecturer Peter Shantz, with Graham as the faculty and Computing Center representative. Graham hired three undergraduates—Jim Mitchell, Richard Shirley, and Bob Zarnke—for the summer to carry out most of the programming. The team hoped to have a working version fully compatible with IBM’s 7040 Fortran IV compiler (known as IBFTC) finished in three months. Perhaps more realistically, they intended to have a debugged, documented version ready to share with other computing centers by October. At some early point in the project, Shantz coined the name Watfor, as a play on Waterloo Fortran, and a fourth undergraduate student, Angus German, joined the programming team.28

At the initial meeting, there was some uncertainty that this was a feasible summer project, but after a few weeks of reading and communicating with members of other similar projects, development began in earnest.30 The primary goal was a fast compiler that maintained runtime efficiency, while simultaneously generating useful diagnostics during the compile and execution phases.

The team would also work toward several educationally driven extensions of the general IBM Fortran language. The most innovative contribution by the Waterloo group was mixed-mode arithmetic and logical expressions, intended to simplify programming by allowing mixed operations between real, integer, complex, or double precision variables.31 Progress by the end of May was significant; the general flow of the compiler was complete and the team had drawn up a list of bugs in IBFTC that would be fixed in their own version.32

By the end of the summer, Watfor was nearly complete and many of the design goals had been met. Like most other student-oriented compilers, to improve throughput Watfor did not generate an object file. It was found to be faster to recompile source decks than to load and link precompiled object files.33 During object-code execution, Watfor remained in memory, ready to load the next student program source deck as quickly as possible. Following in the footsteps of Forgo, Watfor provided runtime diagnostics and checks, such as verifying index values within the specified bounds of an array. When a student’s object code finished executing, Watfor printed a source listing with any applicable warnings, the program results, and the compile and executing times.

In late August, with the cooperation of IBM’s Toronto Data Center, Watfor was run in a head-to-head speed test with IBFTC on IBM Toronto’s IBM 7044.34 Three tests were conceived to evaluate each compiler’s ability to handle typical student jobs. The first test of 14 short programs took almost 13 minutes with IBFTC, but only about 10 seconds with Watfor—a phenomenal rate of more than 5,000 cards per minute. The second test was to compile and run a single large and complex program. Although the IBFTC-generated code was considerably more efficient, Watfor could compile the program so much faster that for a single run it was quicker overall. A final large-scale test of almost 100 programs of typical length and complexity for an undergraduate was not even allowed to run its course. Watfor finished in under four minutes and shortly thereafter the test was terminated when it was clear that IBFTC would not finish for hours.35

The first official version of Watfor was unveiled on 14 September 1965.36 It had taken only five months for four third-year undergraduate mathematics students, working 10 to 12 hours a day, to write what was believed to be the fastest Fortran IV compiler available for the IBM 7040. Waterloo faculty
and graduate students put it to use immediately, and within a few months, it was available for undergraduates on a limited basis.

Although Graham had Waterloo’s local teaching circumstances in mind with Watfor, he was not shy about promoting and advertising its success. In one press release in October 1965, the compiler was framed in terms of the multimillion-dollar savings created by Watfor’s impressive speed, noting that the “giant 7040 computer now has the capacity of five such machines.”37 Graham also sent an announcement to the IBM user community Share and the foremost journal of academic computing news Communications of the ACM.38 Any IBM 7040 computing center could request a copy of Watfor by sending a blank 1,200-foot reel of magnetic tape to the University of Waterloo, which would return a copy of Watfor with a full set of documentation. Cooperation and sharing of resources in this fashion was common among IBM users, particularly when it improved efficient use of resources.59 Within a few months, five other Canadian universities and 11 institutions in the US were running Watfor.60

The response from other centers was immediate and enthusiastic. For example, the director of the University of Pennsylvania Computing Center wrote to Graham:

I would like to gild and prominently display your WATFOR tape…The staff is now exercising the compiler and getting more and

more excited about it. This Computer Center will never be quite the same again.41

The University of Iowa reported that the average compilation of student jobs now took one hundredth the previous time.42 The State University of New York at Buffalo Computing Center sent a copy of their newsletter urging all SUNY faculty members to switch to Watfor immediately.43

Not just universities and colleges were interested in Watfor. By the summer of 1966, of the 66 centers using Watfor, almost one-third were considered “industry.”44 Because student-oriented compilers were generally and explicitly designed to be compatible with the Fortran standard, even experienced programmers stood to benefit. Debugging and testing their own programs were best handled with a fast student-oriented compiler, but once their programs were ready for production, the slower IBM compiler was selected to generate the more efficient object code.45

360 Watfor and a New Plan
As Watfor’s reputation spread, requests poured into Waterloo for a new version that would be compatible with IBM’s recently announced System/360 computer. Many computing centers, both academic and in industry, were upgrading to IBM’s powerful new system but hitting the same problem as before: slow Fortran compilers with poor diagnostics. E.D. Kingsburg of Imperial Oil Canada’s Systems and Computer Services Department wrote to Graham that, “The IBM 360 compiler leaves a lot to be desired” and encouraged a 360 version of Watfor.46 F.K. Dietzler, a systems programmer for Goodyear Tire and Rubber Company in the US, also encouraged Graham and observed, “It is abundantly clear that IBM intends to do nothing about it…[and] we would consider support to an organization such as yours.”47

In May 1966 when Waterloo put in its order for a System/360 Model 75, Graham also put lecturers Paul Cress and Paul Dirksen and a team of undergraduates to work on the 360 Watfor. Unfortunately, Waterloo’s own System/360 was on backorder, so much of the initial design and coding was done by hand during the week and cards were driven one hour to IBM’s Data Center in Toronto for testing on the weekends.58 Cress and Dirksen hoped to have the new version ready by the end of 1966, but the 360 Watfor was not finished until April 1967, perhaps just in time.
A month earlier, at a Share conference in San Francisco, the two were mobbed at a short session organized to discuss the near-ready compiler.49

Technically, 360 Watfor was highly derivative of the original version. It was fast, offered comprehensive error diagnostics and was as compatible as possible with Fortran IV for the System/360. As with its antecedents, the load-and-go compiler remained resident in memory during the compile and execute phases and did not generate object files to reduce overhead and process jobs quickly. The internal program flow was similar to that of the original Watfor, although some lessons had been learned and the compilation sequence was more sophisticated.50 Error handling and diagnostics were both improved, and 360 Watfor could generate more than 300 different compile or runtime error messages.51

The new 360 Watfor was also created with a broad set of users in mind. For instance, it could be run on any of the different System/360 operating systems. To serve universities and colleges with smaller budgets, the compiler was designed to operate with minimal hardware requirements on cheaper models and configurations. To suit beginner programmers, it was as permissive as possible with respect to Fortran syntax. For example, although standard Fortran limited variable names to six characters, Watfor simply truncated any longer names, warned the programmer, and continued compiling.52 However, because a significant number of industrial computing centers and programmers were expected to use 360 Watfor, features were added that experienced programmers would find useful, including a mode that disabled most of the diagnostic features to increase execution speed of the final object code.53

As expected, 360 Watfor performance was exceptional. When it was nearly complete, a round of tests on a borrowed System/360 Model 40 showed that an IBM 1103 printer straining at its maximum speed of 1,100 statements a minute could not keep up with the compiler's output. A second series of tests with the printer disconnected indicated that the compiler was easily capable of processing many thousands of student jobs per day.54

The biggest difference between the two versions of Watfor was not technical but the manner in which it was distributed. In early 1967, Graham decided that the Computing Center could no longer afford to provide and maintain the widely used compiler for free. In only 18 months, more than $20,000 had been poured into debugging, distribution, documentation, and maintenance of 7040 Watfor—far more than had been budgeted in the summer of 1965 to write the compiler. Graham prepared a pricing plan with two options. For $30, anyone could request a copy of the compiler and a few demonstration programs, but they would not receive any paper documentation, support, or upgrades. For $300, operators received a copy of the compiler source and object decks and all the guides, manuals, and documentation related to installation, programming, and compiler design. For the additional cost, the Computing Center would send them any updated versions of the compiler and advice was available by mail and telephone.55

The $300 cost was far less than the monthly rental cost of the System/360, and without any reasonable alternative, hundreds of universities purchased the support and service plan. This source of funding allowed Waterloo to continue to develop Watfor and other similar educational software packages.

In 1968, 360 Watfor was succeeded by Watfiv (which stood for Waterloo Fortran IV). As IBM’s line of System/360 computers found worldwide acceptance and success, so did Watfor. Compared to other Fortran compilers, including IBM’s, Watfor and Watfiv were considered superior for software development and widely recommended.56

Other “Waterloo” programming languages soon followed. When a suitable version of LISP could not be found for their System/360, Waterloo developed its own for artificial intelligence research. One local legend has it that Joseph Weizenbaum, the AI pioneer, once visited Waterloo and bet that Waterloo’s LISP could not solve an “impossible” pet problem of his. He lost the bet and dutifully paid Graham a quarter.57 In the early 1970s, when employers in the cooperative education program insisted that students learn Cobol, a student-oriented version was created called Watbol.58

To help organize this effort, in 1972 Graham established the Computer Systems Group (CSG), a nonacademic department of the university. CSG’s mandate was to create new software packages to expedite the educational process and maintain existing packages, while operating as a nonprofit organization.59 By 1982, the annual income
from software sales and licensing by CSG and several other similar groups established by Graham had exceeded $1.5 million. Graham reflected that same year:

> When we are able to solve a problem, frequently the identical solution is required at other universities, so we export the software. When we do that, we receive income. The income is modest, but coming from so many places, it adds up.60

The income was typically directed back into the university to purchase new hardware and to sponsor scholarships in the Faculty of Mathematics, thus contributing enormously to the university’s reputation and attracting exceptional faculty and students.

In the mid-1970s, CSG also developed the Waterloo Interactive Direct Job Entry Terminal (Widjet) that again improved student access to mainframe computing. Students could sit at a screen or keyboard terminal that was connected to a DEC PDP-11 multi-user time-sharing minicomputer and directly edit their programs. When the student was ready, PDP-11 transparently passed the program to the mainframe to be compiled, and the results were transparently returned back to the terminals and displayed to the students. The system was relatively inexpensive and eliminated any need for punched cards. Unsurprisingly, Widjet was well received by other universities and marketed by both IBM and DEC.61

In the late 1970s, CSG also developed the Waterloo Systems Language (WSL, pronounced whistle), a portable C-like programming language for the IBM 370 mainframe. It was written to avoid licensing conflicts with Bell Labs, which owned the intellectual property rights to C at the time. The WSL compiler and other Waterloo software were subsequently adapted for early microprocessors, and Graham helped three of his former students spin these efforts off to form a separate business called Waterloo Communications, otherwise known as Watcom. There, Graham’s protégés developed a variety of educational software packages throughout the 1980s and 1990s, concentrating on compilers and programming languages for use with personal computers. As Watcom’s chairman of the board, Graham worked actively to establish a relationship of shared benefits between the university, CSG, and Watcom. This model was used by many of the other highly successful, high-tech spin-off companies associated with the University of Waterloo.62

**“WAT” For Ever**

Watfor’s legacy is considerable. One estimate from within Waterloo suggests that over a quarter of a century, millions of students worldwide learned to program using Watfor.63 Many programming textbooks written at this time encouraged its use, including Graham, Dirksen, and Cress’s own book, *Fortran IV with Watfor*.64 Their popular and well-reviewed text outsold all other books of its kind in 1970, according to the publisher Prentice-Hall.55

Waterloo’s success was recognized in various ways. Student-oriented compilers and software represented a narrow, unappreciated branch of the computing world that depended largely on the initiative of university computing centers. Aside from the spartan Gotran from IBM, computer manufacturers showed little interest in developing a Fortran compiler for their systems that would be suitable for the academic environment. In 1972, Cress and Dirksen shared the ACM Grace Murray Hopper Award, given to the outstanding young computer professional of the year on the basis of a recent major contribution. Their citation read, “For the creation of the Watfor Compiler, the first member of a powerful new family of diagnostic and educational programming tools.”66 More recently, histories of computing have acknowledged Watfor’s contribution to the history of programming languages and computing education.65 Charles Davidson, who helped create Forgo at the University of Wisconsin, once noted that “the world owes [Waterloo] a great debt for the marvellous job they did ... in providing and supporting processors for many phases of the instructional (and production debugging) process.”68

Yet from a technical perspective, Watfor itself was never particularly innovative or revolutionary. Internally, the System/360 version was based on the IBM 7040 version, which owed a great deal to Forgo, which was itself inspired by Gotran. Solving the throughput problem and improving error messages for student programmers were not new ideas in 1965, although it can be said that the succeeding versions of Watfor accomplished these goals well. The original Watfor represented a grounded understanding of the throughput problem with a sprinkling of new techniques. The 360 version...
was simply the logical next step, just as Watfiv/S was created in 1974 to accommodate recent developments in structured programming (such as the WHILE and IF-THEN-ELSE constructs) and multiprogramming features. From this internal viewpoint, Watfor remains evolutionary, not revolutionary.

Watfor's story begins with the University of Waterloo's pedagogical philosophy: “Education is best accomplished when there is a lively interaction between the academic community and the larger community which it serves.” When work began on the compiler, Waterloo was less than a decade old and its most notable characteristic was its cooperative program that recently had been expanded to include computer science students. For the co-op program to function, tight connections with industry were necessary; to be hired, students needed experiences employers would value, including a solid grasp of industry-relevant programming languages such as Fortran. As Graham had pointed out in 1964, undergraduates could not be efficiently prepared to contribute to an increasingly computerized world without student-oriented compilers. This commitment helped drive Graham and Waterloo toward Watfor.

The jump from the IBM 7040 to the IBM System/360 provided additional lift. Graham would argue, in retrospect, that the accelerating enrollment at Waterloo all but guaranteed that a new version of Watfor would be written for the System/360. Yet, as Waterloo and other schools that had created and distributed student-oriented compilers discovered, the choice would not come cheap. Graham’s “service and materials fee” neatly solved this part of the problem. Even the smallest college could find $300 if it could save tens of thousands more.

The entrepreneurial solution was unlikely to emerge at any other university in Canada but was unsurprising at Waterloo, given its links to industry. As many in the computing field learned in the 1960s, software was almost never as cheap as expected; the ongoing costs of support, maintenance, and enhancement could easily outstrip the initial development budget. Success came from recognizing the benefits and costs of the enterprise and creating the necessary support structures. Watfor’s technical merits are respectable, but the self-funding model required a marketing and sales finesse and sophistication that few other schools had. Ralph Stanton and Wes Graham in particular had extensive contacts and experience with the computer industry. The Computer Systems Group that Graham launched in 1973 to organize the development of educational software packages indeed operated much like a computer software company, not a university department.

By the early 1980s, the University of Waterloo’s undergraduate computer science program was well regarded internationally, thanks to its leading-edge computer education environment. The university had an infamous reputation for aggressively courting industry, often accepting large donations of computing hardware in exchange for a reciprocal commitment to develop a compatible software package or programming language. Other Canadian universities openly criticized this approach, arguing that Waterloo should concentrate on long-term research and academics, rather than applied software; a computer science department’s reputation should be based on the intellectual caliber of research, not the size of corporate grants or software sales. The response from the chairman of the Department of Computer Science was completely characteristic of Waterloo: “We must be in touch with what is going on in business, or we will quickly become obsolete.”

Perhaps both as blessing and curse, the “Wat” prefix became a catchall naming mechanism at Waterloo. Aside from the various compilers and educational software applications, examples include Watcow, the Computer Systems Group’s custom-built and portable Waterloo Computer On Wheels; Watcat, a computerized library card catalog; and Watcard, the student ID card. Although few now remember or appreciate its roots, the prefix has been, and continues to be, used by countless laboratories, research groups, student clubs, and even environmental committees. In retrospect, “Wat” might seem obvious, but Watfor inaugurated the tradition.

In 2005, the University of Waterloo Department of Computer Science was renamed the David R. Cheriton School of Computer Science. Somewhere along the way a tongue-in-cheek departmental coat of arms from the 1980s was misplaced and few remember it now (see Figure 3). The design specifically honored the ancestry of Watfor with a banner containing three short words of unclear order. Many undergraduates might have read “WAT EVER FOR,” but the preferred interpretation was “WAT FOR EVER.”
"Wat For Ever:" Student-Oriented Computing at the University of Waterloo

Figure 3. University of Waterloo Department of Computer Science coat of arms. (Courtesy of Peter Ponzo.)

References and Notes


5. Unfortunately, Graham died in 1999, not long after he retired, and could not be interviewed. However, his enormous personal archive was donated to the University of Waterloo over the following years and proved invaluable to this project.

6. Technically, it was not the first computer at the University of Waterloo. The school had tested the Bendix G-15 for a week in 1960, and the 1620 was on backorder, so Graham and the committee settled for a rented IBM 610 for about a year.


20. At one point more than 400 schools were using Forgo. Davidson, “The Emergence of Load-and-Go Systems for Fortran,” p. 37.

21. Ralph Stanton was visiting Wisconsin for a lecture series and was accompanied by a promising undergraduate named Angus “Gus” German. German discovered FORGO and surreptitiously copied a deck of punched cards to bring back to Waterloo. Ponzo, Computer Science at Waterloo, p. 25.
30. Although their notes refer to several publications related to Algol compilers, there was never any doubt at Waterloo that they would be writing a Fortran compiler.
33. S.C. Hope to Mr. James, letter, UW Archives GA133, folder 88, 11 May 1966.
34. The IBM 7044 was slightly faster, but the two machines were fundamentally compatible.
36. “Computer Science. Memorandum to All Faculty and Graduate Students,” UW Archives GA133, folder 1132, 7 Sept. 1965.
38. More than a three-paragraph Watfor news item appeared in the same Nov. 1965 issue that published a description of the design and capabilities of Purdue University’s similar PUFFT for the IBM 7090. S. Rosen, R.A. Spurgeon, and J.K. Donnelly, “PUFT—The Purdue University Fast FORTRAN Translator,” *Comm. ACM*, vol. 8, no. 11, 1965, pp. 661–666. Shantz and his team would write a similar article describing the main design considerations and features of Watfor. P.W. Shantz et al., “WATFOR—The University of Waterloo Fortran IV Compiler,” *Comm. ACM*, vol. 10, no. 1, 1967, pp. 41–44. However, by the time the journal received the Watfor article in July 1966, the compiler was already well known to the 7040/44 community and Graham had a new team working on a version for the IBM System/360.
41. B. McClure to S. Hope, letter, UW Archives GA133, folder 1225, 1 Nov. 1965.
42. J. Whitely to S. Hope, letter, UW Archives GA133, folder 1220, 23 Nov. 1965.
43. A. Ralston, “Newsletter #9,” UW Archives GA133, folder 1220.
50. For example, whereas the IBM 7040 version used a symbol table with fixed-length words, the symbol table of the new version was managed with linked lists. P.H. Cress et al., “Description of /360 Watfor: A Fortran-IV Compiler”
"Wat For Ever:" Student-Oriented Computing at the University of Waterloo


57. Ponzo, \textit{Computer Science at Waterloo}, p. 32.


61. Cowan et al., “Development of Educational Software Using the DEC PDP-11.”


63. Ponzo, \textit{Computer Science at Waterloo}, p. 32.


66. See ACM Award Citation for Paul E. Dirksen and ACM Award Citation for Paul H. Cress, http://awards.acm.org/hopper.


72. Davidson notes that the University of Wisconsin lacked the necessary resources to forge ahead as Waterloo did. Davidson, “The Emergence of Load-and-Go Systems for Fortran,” p. 37.


Scott M. Campbell is the director of the Center for Society, Technology, and Values and a lecturer in the Department of Systems Design Engineering at the University of Waterloo, Canada. His research interests include the history of computing in Canada, and he coauthored a book on the topic with Zbigniew Stachniak, \textit{Computing in Canada: Building a Digital Future} (Canada Science and Technology Museum, 2009). Campbell has a PhD in the history of technology from the University of Toronto. Contact him at scott.campbell@uwaterloo.ca.