Hicks, Judith. "The Educational Theories of John Ruskin: A Reappraisal." *British Journal of Educational Studies* 22 (1, 1974): 56–77.

Mashaal, Maurice. *Bourbaki: Une Société Secrète de Mathématiciens.* Paris: Belin, Pour la Science, 2000.

Senechal, Marjorie. "The Continuing Silence of Bourbaki: An Interview with Pierre Cartier." *Mathematical Intelligencer* 20 (1998): 22–28.

Colin McLarty

DIJKSTRA, WYBE EDSGER

(*b.* Rotterdam, Netherlands, 11 May 1930; *d.* Nuenen, Netherlands, 6 August 2002), *computer science, logic, mathematics.*

In 1972 Wybe Dijkstra became the first Dutch computer scientist to win the Turing Award, at the young age of forty-two. He counts as one of the founders of the discipline of computer science itself. He wrote the first Dutch textbook on programming, between 1952 and 1955. His work aimed at developing a theory of computing without computers. He was a member of the Royal Netherlands Academy of Arts and Sciences and a foreign honorary member of the American Academy of Arts and Sciences. He received a large number of prizes and distinctions.

Biographical and Career Details. Dijkstra grew up within an intellectual environment. Both his parents had taken university degrees: his mother was a mathematician, and his father was a chemist. In 1948 he finished *gymnasium,* the highest level of high school in the Netherlands (pupils receive education in Latin and Greek). He was groomed for a scientific career. His parents thought it would be a pity not to devote his life to science, and he followed their advice. He studied theoretical physics at one of the oldest Dutch universities, in Leiden.

In 1952 he started his career at the Mathematical Center in Amsterdam. He finished his university studies in 1956, and received his PhD in 1959 on "Communication with an Automatic Computer." In 1962 he became a professor at the Technical University in Eindhoven. In 1973 he became research fellow for the Burroughs Corporation. In 1984 he moved to Texas, to the University of Texas in Austin, retiring in 1999. He was married and had three children. He died of cancer in 2002 at age seventy-two.

Professionalization of Programming. Maurice Wilkes built one of the first electronic digital stored-program computers in the world, the EDSAC (Electronic Delay Storage Automatic Calculator) at the University of Cambridge. He also developed one of the first programming courses in Europe, and in 1951 Dijkstra became one of his first students. This course was Dijkstra's first encounter with electronic digital computing machines. Dijkstra's entrance into the field of computing via knowledge of and experience with programming influenced Dijkstra's further career: although he participated in the logical design of some early digital electronic machines, he never involved himself with the material construction of a computer. People from all over the world joined Wilkes's courses, which means that Dijkstra belonged to the international community of computer experts from the start. It also means that Dijkstra entered the field in the context of scientific computing.

Dijkstra started working at the Center for Mathematics and Computer Science in Amsterdam (the former Mathematical Center), not yet having finished his studies. This center was subsidized by the national government, aiming at making mathematics useful for society. Numerical analysis and statistics were the core business, and this work involved a lot of computation. Therefore, the center had a computation department, which was headed by Adriaan van Wijngaarden. The first professor in computing science in the Netherlands, Van Wijngaarden was one of the key figures in the development of the computer language ALGOL 68. This computation department emerged as one of the leading institutions in the pioneering era of Dutch and European computer science in the context of scientific computing. Van Wijngaarden convinced Dijkstra to become a programmer, arguing that "computers are here to stay" (Dijkstra, EWD1308, p. 1).

In 1953 Dijkstra developed a programming manual for the first Dutch electronic digital computer, which was still being built at the time, the ARRA (Automatisch Relais Reken Apparaat, or Automatic Relay Calculator). Thus, he developed his first thoughts about programming without a machine at his disposal. Given the fact that early computers were rare, and that these were rather laboratory experiments than proper working machines, this was not so exceptional. For example, Arthur W. Burks and John von Neumann developed ideas about coding in 1946–1947, before the IAS (Institute for Advanced Study) machine had been built.

Dijkstra wrote, together with Van Wijngaarden, the first programming textbook, which was well entrenched in current knowledge and practice of the early 1950s. It included a discussion of the computer itself (the ARRA), a section on flowcharting, a library of subroutines, examples of programs, and a discussion of interpretative programming. In this textbook, one chapter is fully devoted on reliability of the results. This is a topic that Dijkstra continued working on throughout his career, especially with regard to software.

In his very first manual about "the programmer's task" in 1953, he initiated the professionalization of the programming activity (without coining it as such). He described the task of the programmer in terms of five steps, each in line with ideas about programming in the emergent scientific computing community. The first step was the mathematical formulation of the problem; the second was the mathematical solution. The third step was the construction of the numerical process that would produce the right result; the fourth step was the actual programming in terms of operations; and the final step was the coding. Flowcharting was used in the fourth step: information technology (IT) was a central concept to represent the structure of the problem, independent of the code that was specific to the machine.

Dijkstra called himself the first programmer in the Netherlands. Whether or not that was the case, he definitely was among the first people in the Netherlands who wrote about programming as a separate activity. In 1962 he described it as follows: "I should like to draw your attention in particular to those efforts and considerations which try to improve 'the state of the Art of Programming,' maybe to such extent that at some time in the future we may speak of 'the state of the Science of Programming'" (Dijkstra, EWD32, p. 1). Throughout his career he put his work in the perspective of professionalization of programming, in theory as well as in practice.

In 1967 he wrote in *Informatie,* the leading Dutch IT journal, "Het einde van een ambacht" (The end of a craft), in which he made a plea for properly (that is, academically) educated programmers. Dijkstra was a very good teacher himself, and used his teaching activities to spread his ideas. Those who knew him generally say that he was at his best in front of a class, explaining his latest insights. In his view, the key to a good computing science program was to consider it as a branch of mathematics. It is important to realize at this point that at the technical universities in the Netherlands there were programs in "technical mathematics," which taught numerical analysis, modeling, and early computing science. He himself had an appointment at such a program. So, mathematics was a natural context for him. He taught numerical analysis himself in the first years of his professorship.

Another aspect of academic professionalization is the definition of the object at the core of a discipline. For Dijkstra, the core object of computing science was the "abstract mechanism," a notion that he introduced in a number of EWD papers (EWD51, for example) and finalized in his 1976 *A Discipline of Programming.* In this book he writes: "I view a programming language primarily as a vehicle for the description of abstract mechanisms" (p. 9). An abstract mechanism is an algorithm that can be executed by an automaton and that produces an unique result when the input is given. Dijkstra was only interested in the formal aspects of the algorithm, and not in the physical machine that performed the execution. All his work was actually devoted to defining foundations for computing science as *science.* In contrast to many others of his time, he did not conceive computing science as a mixture of disciplines (electrical engineering, mathematics, and so on), but as proper mathematics. In 1961 he had already written that the mathematician "has theorems, we have subroutines" (Dijkstra, 1961, p. 4).

An academic discipline needs an object and a method. Together with a Dutch colleague Wim Feijen, he wrote *A Method of Programming* (1984, in Dutch), which was translated into English in 1988. In this book programming is presented as a "formal branch of mathematics."

Operating Systems. From the start, Dijkstra not only programmed but also reflected on the activity itself. He tried to find what he called "general" technologies and "general" statements, by which he meant technologies and statements that were independent of the specific computer and the specific program. Dijkstra used this word *general* throughout his career.

His first attempt to find such a general technology was the interrupt. One of the major problems at the time was that input and output devices were much slower than the clock of the computer. When the computer had to print a result, while the printer was still being occupied printing a previous result, the computer had to wait until the output device had finished typing (for example). That was considered to be inefficient. An interrupt told the computer to continue executing its orders, and to store the result that should be printed. The computer continued executing its program. When the printer was ready printing, the interrupt told the computer, and a connection was made between the stored result and the printer. Generally, this was coined as a synchronization problem, and the interrupt prevented waiting time. Already in 1955, he had written *Het communicatieprogramma van de ARRA* (Communication program of the ARRA), which shows his early interest in what would, in the early twenty-first century, be called the operating system of the computer.

Between 1956 and 1959 he worked on his PhD dissertation, "Communication with an Automatic Computer," published as a book in 1959, in which he developed an interrupt mechanism for the first Dutch commercial computer, the Electrologica X1. The book contained only the programming aspects of the mechanism (the hardware side of the interrupt was built by Bram Loopstra and Carel Scholten). The interrupt was at the vanguard of research at that time. For example, Frederick Phillips Brooks Jr. and Dura Sweeney patented an interrupt system for the Stretch Computer, built at IBM during 1956–1959. However, Dijkstra did not cite their work in his dissertation. In 1968 Donald Knuth argued in *The Art of Computer Programming* that most fundamental techniques until then (and the interrupt system was one of them) had been independently developed by a number of different people. About Dijkstra, Knuth wrote in 1968: "An interrupt system which enabled buffering of input and output was independently developed by E. W. Dijkstra between 1957–1958. His thesis mentions buffering techniques, which in this case involved very long circles of buffers since the routines were primarily concerned with paper tape and type-writer I/O. Each buffer contained either a single character or a single number" (p. 227).

Dijkstra continued his work on parts of what would be called operating systems. The first paper on semaphores, "Multiprogrammering en de X8," was circulated in Dutch in 1962 (EWD51). Multiprogramming meant that the central processor was able to divide its time among a variety of jobs. (This is different from concurrent programming, in which several processors carry out the same job). In this paper he introduced his idea about *seinpalen* (semaphores), explaining that he used a metaphor derived from the railways because that was how he conceived of the regulation of concurrent sequential processes. How do several machines, either concrete or abstract, know that they can send or receive jobs or data? Generally, semaphores regulated the synchronization between loosely connected sequential processes, for example, the synchronization between an abstract mechanism (the program) and the input/output devices.

This work would finally result in a very influential paper, "Cooperating Sequential Processes" (EWD123, 1965). He proved the correctness of the logic of the semaphores. This paper was widely read, and many of the concepts that he introduced in that paper were quickly adopted by the pioneering computing science experts. In the early 1970s, the first books on operating systems came out, among them in the Netherlands by professor Arie J. W. Duijvestijn in 1973 (who had been the first programming expert at Royal Philips Electronics NV and was professor at the Twente University) and in the United States in 1973 by Per Brinch Hansen, who had developed a multiprogramming system for the RC 4000 computer at the Regnecentrale in Denmark. These works show that Dijkstra's concepts (semaphore, deadlock, critical regions, message buffers) were highly influential at the time. An example is the banker's algorithm, an algorithm that prevented deadlock during the execution of concurrent processes by one single processor. A deadlock situation is, for example, the case when two processes wait for each other during the execution of the program. Dijkstra introduced it internationally in EWD123, although a more primitive version of it had already been introduced in EWD108 in Dutch. This banker's algorithm was almost literally adopted in Brinch Hansen's book on operating systems (pp. 42–45). Dijkstra also introduced (internationally) the structure of hierarchical levels in his 1968 paper "The Structure of the 'THE'-multiprogramming system" ("THE" stood for Technical University Eindhoven).

Programming Languages and Compilers. The Mathematical Center was involved in the development of ALGOL 60, and was one of the central actors in the development of ALGOL 68. These were algorithmic languages, partly developed in competition with IBM's FORTRAN. The Mathematical Center provided a good environment for working on programming languages and compilers. Dijkstra was not directly involved in the development of ALGOL 60, as he was working on a compiler for the Electrologica X1 computer. However, his book *A Primer of ALGOL 60 Programming* was reprinted almost yearly and was the standard textbook on ALGOL in the 1960s.

A compiler translates the high-level language code into machine language code. Together with Jaap van Zonneveld, Dijkstra wrote the first ALGOL-compiler in the summer of 1961. The code of this compiler has recently been documented by Frans E. J. Kruseman Aretz (2003). This work proved to be very influential. He developed several concepts, the most famous of which is probably the "stack." The stack referred to a specific way of organizing memory during the executional process, following the principle of "first in, first out." It used recent developments by the German computer pioneers Friedrich Bauer and Klaus Samelson. This work became very well known, as is shown by textbooks in computing science *avant la lettre.* These early books within a discipline are mostly collections of important papers as long as a core body of knowledge is still lacking. In 1967 Saul Rosen, a professor in computer science at Purdue University, edited a collection of important articles on programming systems and languages, and Dijkstra's article on the stack was one of them:

> "Recursive Programming" by Professor Dijkstra is an early and important contribution to the art of writing compilers. The problems involved in permitting recursive calls on subroutines are attacked and handled in a simple elegant fashion. Almost everyone who has been involved in writing an Algol compiler has used some of the ideas developed in connection with the Algol compiler written by Professor Dijkstra and his colleagues at the Mathematical Centre at Amsterdam. (p. 181)

In 1968 Dijkstra published one of his most widely known articles, "GoTo Statement Considered Harmful."

It was about the statement "GoTo" in high-level programming languages (the original title of this paper was "A Case against the GoTo Statement" EWD215). In this article he addressed one of his central concerns: the gap between the static program text and the dynamic process of its execution. Dijkstra's striving for logic and proof of correctness originated from his conviction that the human mind was not very good at thinking through the process of execution of a program (executional abstraction). In many of his papers, he addressed this fundamental problem. In the course of his career, he increasingly considered programming to be a mental activity.

This problem, the confusion between a program and its execution, is also the key to his famous notion of "separation of concerns" (EWD447, 1974). According to Dijkstra, in the daily practice of programming, the separation between the preparation of the program and its execution was unclear. That caused a lot of problems. In his view, being well-defined rather than being implemented was a vital characteristic of a programming language. In this paper he asks which should take priority:

> On the one hand we have the physical equipment (the implementation), on the other hand we have the formal system (programming language). It is perhaps a question of taste—I don't believe so—to whom of the two we give the primacy, that is whether it is the task of the formal system to give an accurate description of (certain aspects of) the physical equipment, or whether it is the task of the physical equipment to provide an accurate model for the formal system—and I prefer the latter. (EWD447, 1974, p. 3)

This closed the circle in a way: after having freed programming from being dictated by the electronics of the machine, now the construction of the machine should enable the implementation of a well-defined and consistent programming language. That was what Dijkstra found at Burroughs when he started working there as an independent researcher. The computer there was one of the few expressly designed to implement ALGOL.

Algorithms. Already in 1957 Dijkstra conceived of his shortest path algorithm, published it in 1959 in *Numerische Mathematik.* A concise history of the shortest path algorithm is given by Helená Durnová (2004). Dijkstra has become known for several other algorithms, of which the banker's algorithm was already mentioned.

Context, Further Career, and Working Style. One of Dijkstra's most important Dutch colleagues, and competitor at the same time, was Willem van der Poel, who also joined the programming courses with Wilkes, and who constructed one of the first computing machines in the Netherlands. Van der Poel became a professor at Delft University of Technology, and was among other things chairman of Working Group 2.1 of the International Federation for Information Processing (IFIP WG 2.1) Dijkstra also participated in IFIP WG 2.1, and later in 2.3.

IFIP was established in 1960. Working group 2.1 was established to work on ALGOL—it still exists under the name of algorithmic languages and calculi. Van Wijngaarden led this working group when they started working on ALGOL 68. Charles Lindsey gave a very neat personal history about the reasons why this group of people, while they worked on ALGOL 68, finally split up: Dijkstra and Niklaus Wirth, who developed the high-level programming language PASCAL, founded (among others) a new working group 2.3, called programming methodology.

Dijkstra and Van der Poel developed quite different conceptions of programming. The key difference is that Dijkstra abstracted from the computer, while Van der Poel remained loyal to the machine. The boundary between hardware and software was Van der Poel's object of research.

Dijkstra was never good at citing others. "For the absence of a bibliography I offer neither explanation nor apology," he wrote in his preface to *A Discipline of Programming* (1976, p. xvii). On the one hand, Dijkstra has left thousands of pages of beautiful text, but on the other hand, it is sometimes difficult to trace his own intellectual inspiration.

From the 1970s, he was the principal motivator of the Tuesday Afternoon Club of the Technical University Eindhoven. This was a weekly seminar where recent work was discussed. This was continued in Eindhoven as well as at the University of Austin after he had moved there. These meetings were inspiring for people who belonged to Dijkstra's inner circle. Increasingly however, Dijkstra's behavior became off-putting now and then; some people experienced him as very offensive in social gatherings. Dijkstra established his reputation through his writings; his style is very elegant to read. Some people, however, tried to avoid meeting Dijkstra, while at the same time tried very hard to get his most recent EWD paper. Dijkstra had four PhD students: Netty van Gasteren, Nico Habermann, David Naumann, and Martin Rem. Habermann worked with him on the THE-multiprogramming system.

Paul Ceruzzi argues in his *History of Modern Computing* that Dijkstra was actually the exponent of a community that aimed at constructing a theoretical basis for computing science. This is a fair statement in the sense that Dijkstra was one of the software pioneers who addressed fundamental issues in such an elegant and rigorous way that he strongly influenced the early community. It is also true in the sense that many of his terms, such as "structured programming" or "separation of concerns," have almost become common sense language in computer science and software practice.

However, people from industry, and people from data processing were not that responsive to Dijkstra's ideas. Throughout his career he made a plea for proving correctness of small pieces (mechanisms) of a big program. People in industry felt that Dijkstra did not understand their problems: problems of large scale and problems of efficiency. In a world where time and money mattered, Dijkstra's programming methodology could not always work. Dijkstra, for his part, did not take the industrial and business context very seriously:

> One of the standard objections raised from the floor is along the following lines: "What you have shown is very nice for the little mathematical examples with which you illustrated the techniques, but we are afraid that they are not applicable in the world of data business processing, where the problems are much harder, because there one always has to work with imperfect and ambiguous specifications." From a logical point of view, this objection is nonsense: if your specifications are contradictory, life is easy, for then you know that no program will satisfy them, so make "no program"; if your specifications are ambiguous, the greater the ambiguity, the easier the specifications are to satisfy. (EWD447, 1974, p. 2)

BIBLIOGRAPHY

Dijkstra corresponded extensively with colleagues in academia and industry by means of numbered papers, reports, commentaries, and so forth, which are known as EWDs. Many of his publications started out as EWDs, but most EWDs were never published; however, they are readily available through archival sources. The Center for American History at the University of Texas, Austin, houses Dijkstra's original manuscripts. All the papers by Dijkstra, including all the references, are available from the "E. W. Dijkstra Archive" at http://www.cs.utexas.edu/users/EWD/welcome.html.

WORKS BY DIJKSTRA

Unpublished Manuscripts

"Some Meditations on Advanced Programming." EWD32, 1962.

"Multiprogrammering en de X8." EWD51.

"Cooperating Sequential Processes." EWD123, 1965.

"Een algorithme ter voorkoming van de dodelijke omarming." EWD108.

"A Case against the Go To Statement." EWD215.

"The Humble Programmer." EWD340.

"On the Role of Scientific Thought." EWD447, 1974.

"From My Life." EWD1166.

"What Led to 'Notes on Structured Programming.'" EWD1308.

Published Works

Het communicatieprogramma van de ARRA. MR21. Amsterdam: Stichting Mathematisch Centrum, 1955. Available from the "E. W. Dijkstra Archive."

With A. van Wijngaarden. *Programmeren voor Automatische Rekenmachines.* Amsterdam: Mathematisch Centrum, Rekenafdeling, 1955. Several further editions of this textbook were coauthored with Th. J. Dekker.

Communication with an Automatic Computer. Rijswijk, Netherlands: Excelsior, 1959. Dijkstra's PhD dissertation, in book format.

"On the Design of Machine Independent Programming Languages." MR34. Amsterdam: Mathematisch Centrum, 1961. Available from the "E. W. Dijkstra Archive" at http://www.cs.utexas.edu/users/EWD/welcome.html.

A Primer of ALGOL 60 Programming. London: Academic, 1962.

With Charles Antony Richard Hoare and Ole-Johan Dahl. *Structured Programming.* London: Academic, 1972.

A Discipline of Programming. Englewood Cliffs, NJ: Prentice-Hall, 1976.

With Wim H. J. Feijen. *Een methode van programmeren.* The Hague: Academic Service, 1984. Translated by Joke Sterringa as *A Method of Programming* (Reading, MA: Addison-Wesley, 1988).

OTHER SOURCES

Apt, Krzystof. "Edsger Wybe Dijkstra (1930–2002): A Portrait of a Genius." *Formal Aspects of Computing* 14 (2002): 92–98.

Bauer, Friedrich, and Klaus Samelson. "Sequentielle Formelübersetzung." *Elektronische Rechenanlagen.* 1, H. 4. (1959): 176–182.

Brinch Hansen, Per. *Operating System Principles.* Englewood Cliffs, NJ: Prentice-Hall, 1973.

Ceruzzi, Paul. *A History of Modern Computing.* Cambridge, MA: MIT Press, 2003. First published 1998.

Durnová, Helená. "A History of Discrete Optimization." In *Mathematics throughout the Ages II,* edited by Eduard Fuchs, 51–184. History of Mathematics 25. Prague: Research Centre for the History of Sciences and Humanities, 2004.

Goldstine, Herman H., and John von Neumann. "Planning and Coding for an Electronic Computing Instrument." Reprinted in *Charles Babbage Institute Reprint Series for the History of Computing,* vol. 12, pp. 145–308. Los Angeles: Tomash, 1982–1992.

Knuth, Donald E. *The Art of Computer Programming.* Vol. 1, *Fundamental Algorithms.* Reading, MA: Addison-Wesley, 1968.

Kruseman Aretz, Frans E. J. "The Dijkstra-Zonneveld ALGOL-60 Compiler for the Electrologica X1 (historical note SEN, 2)." CWI-report Note SEN-N0301. Amsterdam: Centrumvoor Wiskundeen Informatica, June 2003. Available from http://www.cwi.nl/ftp/CWIreports/SEN/SEN-N0301.pdf.

Lindsey, Charles H. "ALGOL 68 Session." In *History of Programming Languages II,* edited by Thomas J. Bergin and Richard G. Gibson, 27–96. New York: ACM Press, 1996.

MacKenzie, Donald. *Mechanizing Proof: Computing, Risk, and Trust.* Cambridge, MA: MIT Press, 2001.

———. "A View from Sonnenbichl: On the Historical Sociology of Software and System Dependability." In *History of Computing: Software Issues,* edited by Ulf Hashagen, Reinhard Keil-Slawik, and Arthur Norberg, 95–136. New York: Springer, 2002.

Rem, Martin, and Hans Schippers. "Edsger Dijkstra en zijn THE systeem" [Edsger Dijkstra and his THE-system]. In *TU/E Gedreven door nieuwsgierigheid* [TU/E driven by curiosity], edited by Harry W. Lintsen and Hans Schippers, 225–231. Zutphen, Netherlands: Walburg Pers, 2006.

Rosen, Saul. *Programming Systems and Languages.* New York: McGraw-Hill, 1967.

Adrienne van den Bogaard

DING WENJIANG (V. K. TING) (*b.* Huangqiao Village, Taixing, Jiangsu Province, China, 13 April 1887; *d.* Changsha, Hunan Province, 5 January 1936), *geologist, educator in geology.*

Ding Wenjiang was one of the founders of geological undertakings in China, especially of the renowned Geological Survey of China that began in 1916. Ding's geological studies were mainly carried out in Yunnan, Guizhou, and Guangxi provinces in the second and third decades of the twentieth century; they contributed greatly to the understanding of Palaeozoic stratigraphy and geological structures in southwestern China. He was later a research professor of geology at Peking University (1931–1934).

Early Life. Ding was born into a local gentry family. His father, Ding Zengqi, married Miss Shan and had four children. Ding Wenjiang was their second son. His exceptional intelligence was shown in one of his examination papers when was eleven years old. In his paper he wrote about the accomplishment of the Emperor Han Wu Ti (140–87 BCE) in developing the southwestern areas of China. His interest in this region seemed to predict his later geological career.

Ding was educated first in Japan (1902–1904) and then in the United Kingdom (1904–1911). He stayed in Tokyo learning Japanese only one and a half years and then went to England where he studied at Cambridge University (1904–1906) and Glasgow University (1907), majoring in zoology and geology. He received two bachelor's degrees from Glasgow in 1911 at the age of twenty-four.

Geological Survey of China. In 1913, Ding was appointed chief of the Section of Geology under the Ministry of Industry and Commerce of the Peking government of China. From early on he recognized the urgent need to train young Chinese geologists to do research. Through negotiations with the authorities in the Geology Department of Peking University, which had not accepted students since 1903, Ding and his colleagues, Zhang Hongzhao and Weng Wenhao, were able to use the building and equipment of the department to establish in 1913 the Geological Institute of China, which was actually a training college in geology. Some thirty students were enrolled and received three years of serious training. In this temporary educational institute, Ding taught geology and paleontology and was especially rigorous in field training and mapping. In 1916, eighteen students graduated. They were the first generation of Chinese-trained geologists and became the backbone of the newly established Geological Survey of China.

Ding was director of the Geological Survey of China from 1916 to 1921. In that post he initiated systematic prospecting of mineral resources and regional geological mapping. He established the National Geological Library and the National Geological Museum, both in Beijing, and authored various geological publications, including the *Bulletin of the Geological Society of China*, which was initially published annually and became a quarterly in 1948. The journal was renamed as *Acta Geologica Sinica*, affiliated with The Geological Society of China since 1952.

Palaeontologia Sinica. Of special note was Ding's role in the development and publication of the multivolume *Palaeontologia Sinica,* one of the most important palaeontological publications. Ding organized it with the help of Johan Gunnar Anderson of Sweden. Ding was the chief editor from 1921 (its first year of publication) until his death in 1936. Another contribution of note was the first issue of the Special Report of the Survey, titled *A General Statement on the Mining Industry of China*, by Ding and Weng Wen Hao, published in 1921. In it the authors point out that unsuccessful prospecting for oil in northern Shensi (Shaanxi) Province was probably the result of insufficient drilling, not lack of oil. This supposition proved correct, and the area later became one of the biggest oil and gas basins in North China.

Geological Society and Peking University. In 1922, Ding helped establish the Geological Society of China in Beijing, one of the earliest natural science organizations in China. He was president of the society in 1923 and was reelected in 1929.

Ding was a renowned geological educator. The Geology Department of Peking University, founded in 1909, but closed in 1912, was restored in 1917. In 1920, Ding invited Amadeus William Grabau from the United States