Experiences and Reflections of a Computer and Systems Engineering Pioneer

Written by:
Harold (Bud) Lawson

Written: February, 2018

CHM Reference number: X8505.2018

Introduction

I have divided this personal history into three phases; namely computer industry, computer-based systems and complex systems. While the years indicated are approximate and there is significant overlap, it provides a structure to conveying the essence of my career. I have worked with many talented people during my career and we have learned from each other. I thank them all and identify many of them. Further, I take the opportunity to provide some reflections that may have occurred during the experience or in several cases later in my career. I also provide a summary of what I consider to be my most important publications.

Phase 1 (1959 to 1974) – Computer Industry

The period involved is from June 1959 to sometime in early 1974. During this time, I was employed first by Remington-Rand Univac, then by International Business Machines, Inc, later by Standard Computer Corporation (Costa Mesa, California) and finally as a consultant to Datasaab (Linköping, Sweden). Also during this period starting in 1967, I began a parallel academic career, first as Associate Professor and later as Professor at the Polytechnic Institute of Brooklyn (Now the Polytechnic Institute of New York) as well as guest professorships at the University of California at Irvine and Linköpings University in Sweden. This was a very exciting period in the computer industry and it turned out to be a very productive period of my life providing a breadth of experiences and knowledge.
I will attempt to name those persons that I closely associated with during this phase of my career. However, the most important person was the late Rear Admiral Dr. Grace Murray Hopper who was my first boss at Remington-Rand Univac (formerly Eckert-Mauchly Computer Corporation). Amazing Grace has a special place in my heart. From her I learned to question the status quo, to seek deeper meaning, to explore alterative solutions and at least implicitly to always try to find the guiding concepts and principles. This has continuously affected my approach to the many problems and opportunities that I have faced in dealing with complex systems for over 59 years.

Where did it start?

I first became exposed to computer technology as a summer student at the United States Census Bureau in Suitland, Maryland during the summer of 1958. A fellow Temple University colleague, Zigmud Decker, who was doing his second summer student tour, and I shared an apartment in Suitland. Ziggy had luckily in both of his tours had the opportunity to learn programming for the Univac 1103 at the Census Bureau. I, on the other hand, was assigned the task of analysing problems and discrepancies in the reporting of cotton linter production. The summer program administrators did organize a two-day course about computing at the census bureau. This introduction along with many late night discussions with Ziggy about computing got me hooked on the idea of pursuing a career in this up and coming area.
During my senior year at Temple, I was employed as the lab instructor in the Department of Statistics under the direction of Professor Rosella James. That year, Temple decided to purchase an IBM 650 drum memory computer. Rosella arranged that I, along with faculty

members, could take an introductory programming course (learning about the assembly language SOAP). Again this confirmed my interest in computing. Also during the year Rosella and I attended a seminar at the University of Pennsylvania given by none other than Grace Hopper. She spoke about teaching computers to understand programs expressed in English. This really aroused my interest. So, when seeking employment I applied to her Automatic Programming Department at Remington-Rand Univac and was employed starting in June 1959.

Remington-Rand Univac

Compiler Experiences

When I arrived at Univac I was issued a pile of documents including the Univac I, Univac II, Flowmatic, Math-Matic and Assembly Language manuals and documentation of the B-0 Flowmatic compiler. At the same time I was given an assignment to modify a code generator in the B-0 compiler. What a swim or sink introduction! I was really glad that I previously had an introduction to computing otherwise I might have sunk. Through lots of hard work and by asking questions of my more senior colleagues, I received a good start in understanding what compilers where all about. As another part of my initiation into the field, I was given the assignment to design and write a sample payroll application in Flowmatic for the City of Philadelphia (they were considering buying a Univac II). Further, I designed and wrote a multi-variable regression (correlation) routine in Math-Matic.

During this period, the CODASYL committee was developing the COBOL programming language. Our Automatic Programming Department at Univac in Philadelphia as well as a corresponding group at RCA in Camden, NJ developed the initial Cobol implementations. The architecture of the compiler for the Univac II (a machine with 2000 words of memory and 10 tape drives) was developed by our senior people including Bill Finley, Tom Jones, Dick Miner and Dan Goldstein. These gentlemen assigned two vital phases of the compiler implementation to me; namely the initial source program parser as well as the most vital code generation phase. During the latter part of 1959 and during 1960, I worked diligently (many 60-70 hour weeks) on designing and implementing these critical elements. Most interesting was the code generation aspect for which I developed a service network of procedures that generated the machine code. I dealt with each type of package of verb and arguments via simple control sequences that invoked the service network. (A service oriented architecture). Most amazing was that I managed to contain the service network in about 1k words leaving adequate space for the control sequences for all of the common cases. An overlay area was used to swap in (from the compiler program tape file) control sequences for more rarely used cases. I called this phase of the compiler DUZ, named after a popular detergent advertisement at that time (DUZ does everything). Since the Univac II was a decimal machine with only decimal arithmetic (no floating point) it was necessary to develop algorithms for calculating shift distances in decimal alignment. I developed and wrote an internal paper on these generalized decimal alignment algorithms.

A most interesting part of the Univac II Cobol implementation is that it was written in a higher-level language, namely, Flowmatic. This was due to the fact that Grace demanded that we view the compiler as a data processing system. Even the small amount of code for the DUZ service network was the result of Flowmatic compiled code. Some critical machine instructions were incorporated in the Flowmatic source code through a mechanism I introduced into Flowmatic called Supplex code.

The Univac II Cobol compiler was composed of about 60 runs (defined transformations with one or more tape input files and one of more tape output files). It took a long time to compile programs and so there where several checkpoints in the compiler. Thus if a machine failure occurred, it would not be necessary to restart, but instead back up to the last checkpoint.

The day of reckoning was in December 1960 when the CODASYL dignitaries showed up one day at RCA in Camden and the next day at Univac in Philadelphia. For the first time in computer history a source program was compiled and executed on two different machines from two computer manufacturers.
A personal note about my relationship with Grace Hopper: On the 9th of December 1960 my daughter Catherine Louise Lawson (now faculty member at Rutgers University) was born on Grace's birthday. Grace was very happy about this and knitted a suit (she really enjoyed knitting) and also presented Cathy with a gold necklace.
During the spring of 1961 Remington-Rand Univac was acquired by the Sperry Corporation. Plans were being made to provide a Cobol Compiler for newly announced Univac III (which had a different architecture than its predecessors the Univac I and II). At this time we had moved from the primitive facilities at 19th and Allegheny Avenue (an old dusty and dirty Exide battery warehouse) to a new luxurious building in Blue Bell, Pennsylvania. Unfortunately, Grace Hopper fell into disfavour with the new Sperry management (she was not a very "manageable" person). This made the future of the Automatic Programming Department uncertain and several employees including myself left at varying points during 1961.

Reflections

I earlier mentioned the tremendous impact Grace Hopper had upon my career, but my other colleagues at Univac provided significant guidance and counselling as well. A period of my life that is hard to forget with many laughs, serious moments and a hell of a lot of hard work.
I often long for the simplicity of the Univac I and Univac II architectures. Being decimal machines with 12 digit words and a limited but powerful instruction set, it was easy to program even at the assembly language level. This simplicity has certainly remained in my mind since then. On the other hand, one had to be programmer, operator and at times repairman for the machines. I thank Norman Rothberg for teaching me the practical skills needs for operation and even repair during many late hours of the night and early mornings.

Figure 1: Univac II – Fond Memories

Cobol for the Univac II was not widely used since the machine was no longer produced after 1961 and Sperry went over to the Univac III as well as eventually a Cobol implementation for the 1100 series of machines.
In 1999, at the invitation of Len Shustek (a former student of mine and Chairman of the Board of the Computer History Museum), I along with Howard Bromberg (earlier also an employee of Grace Hoppers department) and later manager of the RCA Cobol effort presented a seminar at Stanford University. This was a stimulating event where I presented the architecture of the Univac II Cobol compiler and Howard presented several interesting perspectives concerning the development of COBOL and Grace Hopper. Amongst many well-known attendees was Stanford Professor Donald Knuth who will be mentioned again later. Gwen and Gordon Bell graciously provided a buffet dinner in their home after the seminar. I met up with many old friends and had a chance to reflect on the good old days with Don Knuth.

International Business Machines, Inc.

Further Compiler Experiences

After having very positive job interviews and job offers from RCA in Camden and General Electric in Phoenix, I decided to join IBM at Poughkeepsie, New York and started in August 1961. The goals for my employment included research work in the area of compiler automation technology (compiler-compilers as they became to be known). I first worked with Rainer Kogen exploring the application of the ideas as developed in X-Tran (a string parsing language). However, early in 1962 I was asked to transfer to an ongoing project at IBMs Federal Systems Division in Bethesda, Maryland. IBM does much to earn its alternative designation (I've Been Moved). The SLANG (Systems Language) project was started by Robert Sibley and there were four researchers in the group including the now famous but tragically deceased Don Estridge (mentioned again later).

The SLANG compiler-compiler had the goal of describing and generating defined parsing algorithms (from a notation similar to Backus-Nauer Form) that could be used in quickly building translators for any programming language. A further goal was a general-purpose back end that could generate machine code for many of IBMs machines. The SLANG compiler was first implemented in Fortran and then implemented in typical compiler-compiler fashion in SLANG itself. SLANG operated on the IBM 7090 and could be ported to other machines. Thus it was aimed at supporting compiler development for traditional second-generation machines like 7090, 7070, 7080, 1401 and 1410.

First Professional Publication

While working on the treatment of data declaration processing in the SLANG project and also largely based upon experiences in the Cobol implementation at Univac, I produced my first professional publication and presented it at the ACM National Conference in Syracuse, New York in 1962.
The paper entitled: ***"****The Use of Chain List Matrices for the Analysis of COBOL Data Structures"* is referenced in Donald Knuth's "Art of Computer Programming "Volume 1, as being the first publication of algorithms for multi-linked data structures.

Reflections

The basic concepts of SLANG were pioneering. Later, parser generation and compiler-compilers were further developed by other research groups. For example as a part of the Unix effort at Bell Telephone Laboratories compiler-compilers became more or less a standard well-known approach.
The SLANG project was moved from Bethesda, Maryland to Poughkeepsie, New York (from where I had come) and was then in the Systems Development Division. Also, several new people where employed. In addition to Robert Sibley and Don Estridge other colleagues I recall included Bob Rosenblom, Jim Young, Dick Maeir and Richard Bailey. Further, I promoted the recruitment of three former members of Grace Hoppers Univac team into the SLANG project; namely John Tronoski, Stanley Park, and Vincent Della Valle.
We worked hard with many late nights to gain computer time access. Further playing bridge became an obsession with many of us and we played at every chance we could both at work and in the evenings. Touch football also was a means of recreation.
One event I vividly remember while the project was still in Bethesda is a weekend that Don Estridge enlisted our support in painting the outside of his house. Don bought a lot of beer and we were about 5 people that showed up to paint. The house sidings (shingles) just sucked up the paint and it was very slow going. The net result of the weekend was that we drank all of the beer and painted about one third of Don's house. Don spent the rest of the summer painting himself. You might remember that it was Don Estridge who later became a Vice President of IBM and moved IBM into the pc era in the early 1980s by establishing agreements with Bill Gates. This made Microsoft.

System/360 Experiences

In the early 1960s, a new computer family design was being developed within IBM. After working towards an 8000 series to follow the 7000 series, the architects including Fred Brooks, Gerrit Blaauw and Gene Amdahl developed a proposal for a family of compatible computers with different bus widths, memory sizes and performances. This became the basis for the System/360 series and was announced in 1964. This was a turning point in the computer industry.
The Instruction Set Architecture of the System/360 was implemented in most hardware configurations via the usage of microprograms. The microprograms provided for achieving the compatibility of various members of the System/360 family. At the same time the hardware implementations of various System/360 series members utilized, for the first time, solid-state logic integrated circuits. The instruction set architecture while consistent amongst System/360 models turned out to be quite difficult for the various compilers in generating effective code. This had an important impact upon the operating system OS/360 and related systems software as described later.

PL/I and the Pointer Variable

In an effort to unify programming languages, IBM together with is user groups Share and Guide defined, by committee, a new programming language. Originally called NPL it became PL/I. It was to be used to replace Fortran, Cobol and even Algol. The language became quite large even though there were many good concepts and principles that were used in its development. The compilers for various System/360 models where assigned to groups in Hursley, England

and in Boeblingen, Germany. It was also planned that the SLANG compiler-compiler would be used to provide an implementation.
In 1964, after the implementations were already started, I was asked to join the design control board for the PL/I language. The language had been defined but it did not contain any facilities for treating heterogeneous linked lists. This was an important omission since it was envisioned that PL/I would be used in implementing graphics and system software. In particular, one large IBM customer; namely General Motors Research Labs wanted to write graphics driver programs in PL/I and one of the compiler implementations was to use PL/I to develop (bootstrap) the compiler.
Due to my earlier published works on linked lists for handling multi-linked data structures, I was assigned the task of developing a proper facility for PL/I. It is at this time that the pointer variable concept was invented and then, after much hard work, integrated into PL/I. George Radin, who led the development of PL/I, pointed out in an article concerning the History of PL/I that the pointer variable turned out to be one of the most utilized facilities of PL/I. To document the pointer variable concepts, I wrote an article published in the June, 1967 issue of the CACM. The paper entitled: ***"PL/I List Processing"*** is also referred to in Donald Knuth's Volume 1 on Fundamental Algorithms. In fact Don Knuth in a later article indicated that he viewed the pointer variable as one of the most important concepts of computer science.[1]
While adding the pointer variable to the already large PL/I programming language was accomplished, had the facilities been defined earlier and integrated, the complexity of the PL/I language could have been radically reduced; especially in the area of storage classes.
I pointed this out in an internal publication, but since the implementations where in full swing, there was no opportunity to back out. On the other hand some subsets of PL/I where the pointer variable provided a central facility were implemented by Bob Brittenham and Ben Melkin at IBM as a systems programming language. Even by microcomputer manufacturers provided pointer variable facilities in the language PL/M.

Reflections

It is unquestionable that the pointer variable has had a significant impact upon programmers around the world. When used properly, it provides a vital capability to handle complex data structures. When used improperly is can lead to bugs that are hard to detect.
One of the most successful uses of PL/I for systems programming was its utilization in the Multics Operating System at MIT, most likely the best OS ever developed. Since the introduction of the pointer variable concept, variants have been implemented and used widely in such popular languages as C, C++, Pascal, Ada, Oberon, Eiffel and even Cobol and Fortran.
During the development of the pointer variable concept, I was fortunate to get continuous feedback from Don Knuth as well as Doug McIlroy of Bell Telephone Laboratories who participated in the Multics effort and later led the development effort that resulted in Unix and the C programming language.
That significant attention has been given to the "pointer variable" is indicated in a Google search nowadays yielding many millions of hits. It would be hard to envision implementing systems software and advanced applications these days without these facilities. In summary, this invention has certainly proved to be one of the more significant contributions to the field of computing.
In year 2000 I was honoured as a Charles Babbage Computer Pioneer by the IEEE Computer Society for this invention that has had a long lasting impact upon the computer industry.

[1] Donald Knuth, Structured Programming with the go to Statements, Computing Surveys 6 (4): 261-301, December 1974.

Operating System/360

While working on PL/I, I was asked to represent the PL/I compiler efforts in a control board established to deal with standards (including calling sequences and parameter passing) for the system programs to be provided with OS/360. This was both enlightening and frightening.
Fred Brooks has documented the problems with OS/360 in his famous book “The Mythical Man-Month”. I saw this tragedy develop. Prior to the large-scale systems programming effort at IBM there was a small group that included George Mealy that put forth a consistent set of concepts and principles that were to be followed in implementing OS/360 and the related systems software. As the systems programming efforts, including compilers got into full swing, the problem of complex code generation named earlier became evident. So, many systems programming projects requested exceptions in order to generate more effective code. The control board was established to review such requests. But as a result the complexity of OS/360 starting from a single quite readable design notebook evolved under a period of about 6 months into a wall filled with documentation that virtually nobody read. Entropy and chaos was a fact.

Reflections

My experiences with System/360, PL/I, the Pointer Variable and OS/360 provided a lasting impression in my mind concerning complexity. I have often called this period the beginning of the march into the “black hole of complexity”. This certainly has led me to always value the ability to establish a small set of driving concepts and principles for everything I have “architected” since then.
An interesting observation concerning System/360 is the fact that the Soviet Union made a “political” decision in the late 1960’s to copy (reverse engineer) the design of the System/360. Despite protests from prominent soviet experts, they established the ES EVM (ЕС ЭВМ, Единая система электронных вычислительных машин, meaning “Unified System of Electronic Computers”) project and a variety of hardware implementations were done in various Eastern block countries. Naturally, they needed to master the complexity of OS/360 in order to make use of IBM’s technology. It turns out that they were never able to manage to deal with the complexity and, although over 15000 mainframes were produced, the costs where enormous. Eventually these very expensive efforts where abandoned with the net results of most likely setting Soviet computer technology back at least a decade; perhaps forever. It is also interesting to note that Soviet computing was earlier following an architectural approach that would simplify software development by using higher-level language based approaches to the hardware architecture in their Elbrus series. This approach was pioneered in the USA by the Burroughs Corporation under the direction of Robert (Bob) Barton when he designed the Burroughs B5000. In fact, some in the Soviet called the Elbrus computers for El Burroughs.

Computer Architecture and Microprogramming

After experiences with compilers and programming languages, in 1965, my interests turned to microprogramming and computer architecture. I led a research group at IBM’s New York Programming Center that studied the utilization of microprogramming in implementing programming languages. My colleagues included Lou Levine, Bob Flynn, George Mandler and George Awad. As microprogramming became a more established approach in hardware, there developed interest in higher-level language oriented instruction set implementations along the lines developed by Burroughs. While our research was interesting it was difficult to have an

impact upon IBM that had already marched into the black hole of complexity with System/360 and was not interested in backing out.
So, in 1967, I decided to leave IBM to pursue a career in academia at the Polytechnic Institute of Brooklyn (described below). However, my interests in microprogramming continued to evolve and I became a founding member of SIGMICRO. I worked with Sam Husson, Barbara Liskov and others including the famous computer pioneer Professor Maurice Wilkes who first introduced the concepts of microprogramming.

Standard Computer Corporation

In 1969, I took a leave of absence from Brooklyn Polytechnic to join Standard Computer Corporation in Costa Mesa, CA. Standard had already developed microprogrammable computers. There I was one of the designers (along with Dave Keefer and Burton Smith) of a unique flexible central processing unit called the MLP-900. Due to a change in corporate direction, only one prototype of this advanced machine was built. It was delivered to the Rand Corporation and then moved to the University of Southern California Information Science Institute. At ISI, it was further developed and integrated into the original ARPA network as a computer architecture research resource that was utilized up to the late 1970s.

Reflections

The development of the MLP-900 aroused the interest of Alan Kay who visited us at Standard Computer. He was developing a language he called FLEX and saw the advantages of implementing a FLEX machine via microcode. Of course, his efforts later led to the Smalltalk programming language.
The MLP-900 went onto the ARPA network and was placed in a dual configuration with a PDP-10. In addition to being able to emulate the PDP-10 it could be used to download microcode implementations of other machines and, for example, ANYUK (military real-time computers) were emulated remotely. This certainly is a unique aspect of the utilization of the ARPA network.

Datasaab

The MLP-900 aroused the interest of several computer manufacturers including ICL in England and Datasaab in Sweden where it was planned to license the design. This did not happen, but I came to Sweden in 1971 as a consultant and designed a new processor called the FCPU (Flexible Central Processing Unit). This processor utilized a hardware implementation of Hoare's semaphore variables in order to provide an asynchronous structure for synchronizing independent hardware modules. It may be one of the first processors to use the idea *local synchronous – global asynchronous* units that nowadays are key properties of low power hardware. The properties of the FCPU where reported in a number of publications including a contribution at the Second International Symposium on Computer Architecture in 1975.
The paper entitled: ***"The Advantages of Structured Hardware"*** that I co-authored with Bengt Magnhagen (the hardware project manager) was selected for the "best paper award". There was significant interest in the computer architecture community. The FCPU was utilized in the Datasaab D23 Computer System series. However, its planned true potential for more directly implementing higher level languages was not exploited. There were about nine FCPUs produced and several D23s delivered to customers. However, the microprocessor era entered in the mid-1970s and changed hardware economics. It is interesting to note that the published

prior-art design features of the FCPU have been cited in two patent infringement cases in which I have been involved as a consultant.

Reflections

The leaders at Datasaab: Gunnar Lindström, Bengt Asker and Wigo Wentzel were very supportive in establishing the environment in which the development took place. Bengt Malm, Häkan Niska, Torbjörn Granberg, Bengt Magnhagen, Rolf Flisberg, Lars Blomberg, Gunnar Hesse and others contributed to the design and implementation effort. An important reflection of this experience was related to working with a small group of qualified professionals. We were able to design and implement this advanced central processing unit with a small staff and in a short period of time. Since the processor was divided into asynchronous units, unit testing could be neatly performed without timing constraints and then the units where integrated where the semaphore registers provided for controlled execution. I have often reflected on this design and development experience with small groups and can mention that I think this is why Sweden, as a small country of about 9 million, can develop advanced technological systems in many areas including cars, trucks, jet fighter planes, telecommunication, etc.

Academic Career

As mentioned above, in 1967, I left IBM and accepted an Associate Professor position in the Department of Electrical Engineering at Brooklyn Polytechnic. The contact with Brooklyn Polytechnic was promoted by Bob Flynn who worked in my IBM research group. He was completing his PhD in Mathematics at the time. At Poly, I had the responsibility of initiating a Computer Engineering program within the department for which a number of new undergraduate and graduate courses were created. Further, I designed a Load and Go compiler for a subset of PL/I called PLAGO and led a group of students including Stan Habib, David Doucette, Aron Kirshenbaum and Len Shustek that implemented the compiler. Several of my former students from Brooklyn Polytechnic entered the computer industry, particularly at Digital Equipment Corporation, and some went on to pursue graduate education. In particular, I am proud to have provided guidance to Len Shustek who completed his PhD at Stanford, was the inventor of the Sniffer for local area networks, contributed to very successful new start-ups in Silicon Valley and is now Chairman of the Board of the Computer History Museum as mentioned earlier.

In order to provide a useful introduction to computing based upon PL/I, I teamed up with a former IBM colleague from Vienna; namely Dr. Erich Neuhold and wrote a book entitled: *The PL/I Machine – An Introduction to Programming*, published by Addison-Wesley.

During the time I was at Standard Computer Corporation (1969-70), I was a part time guest professor at the University of California, Irvine where I contributed to their graduate program and advised some students including Larry Rowe, a very successful professor at Berkeley.

After my consulting activities at Datasaab terminated in 1973, I became a guest professor at Linköping University in Sweden. There in addition to providing graduate courses, I assisted in defining the D-line, a new computer engineering degree program (the first in Sweden).

During the next six years I worked as a consultant and part-time professor. In addition to Linköping, I had short-term appointments at the University of Stuttgart, Universidad Politecnica de Barcelona and the Royal Technical University in Stockholm.

In 1979 I was selected for a permanent professor's chair of Telecommunication and Computer Systems at Linköping University. Interestingly, when installed in the presence of Carl Gustaf, King of Sweden, Grace Hopper participated as she was selected by the faculty to receive an

honorary doctorate. Prior to the installation Unisys sponsored a dinner and seminar to honour Grace Hopper where I provided a summary of Univac compiler technology of the 1950s.
In addition to innovative educational activities, I led a research group in exploring asynchronous circuit technology together with Bryan Lyles, a guest researcher in my group that earlier had followed development of the computer architectures (MLP-900 and FCPU) that I developed. In addition, we explored various graphic means of creating programs where Arne Jönsson, Michael Pääbo, Mats Lenngren, Thomas Peterson, and a guest researcher from Keio University, Kee Yamamoto participated. **I** left Linköping University in 1986 in order to pursue full time consulting activities working from my new base in Stockholm.

Reflections on Phase 1

My experiences in the computer industry and academia yielded a great appreciation for the relationship of hardware and software (especially system software). Observing the mismatch between the Instruction Set Architecture of the System/360 and the needs of the systems software provided the impetus to seek better solutions. Of course, even though I pointed to these during my last appointment at IBM, there was no return. The march into the black hole of complexity was impossible to stop.
Well, I tried in several processor designs; namely the MLP-900 and the Datasaab Flexible Central Processing Unit to provide the basis for improving the hardware/software relationship. They were interesting architectures and aroused significant interest in the computer architecture community, but by the mid-1970s, the microprocessor showed up and changed hardware economics.
While the integrated circuit advances of the microprocessor where outstanding, the early microprocessors could only implement a simple instruction set. Eventually, this led to the X86 architecture of Intel as the dominant microprocessor. Building simple applications on top of the X86 architecture (or others such as the Motorola 6800) was quite OK. However, today we are building the world's most complicated software systems on top of the X86 base that was never designed for this goal. A complete miss-match between the hardware and software leading to extremely complex, and not well understood as well as difficult to maintain software systems. The net result being: bugs that are hard to find, viruses, as well as opportunities for hackers to exploit the complexity. One wonders how the world of computing would be today had the so-called language directed architectures became dominant in the industry.
I have written several articles to point out these problems including one calling for the Rebirth of the Computer Industry as a viewpoint article in the Communications of the ACM. However, it may take a major crash of the Internet or something equivalent before society drives the industry into taking fundamental root cause corrective measures that will include a better hardware/software relationship.

Phase 2 (1974-1996) - Computer-Based Systems

From the mid-1970s, I shifted focus and concentrated on advanced applications of computer technology resulting in a variety of successful systems. This has included high-voltage power dispatching for ENHER of Barcelona, the automatic train control system at Standard Radio (still used) by the Swedish Railways, and a limited slip-coupling device (now used in virtually all four wheel drive vehicles) for Haldex. In addition, I assisted the Swedish Defence Materiel Administration in establishing and verifying safety requirements for embedded systems.

High Voltage Power Dispatching

Working with a former graduate student from the Polytechnic Institute of Brooklyn, Miquel Bertran, I developed a structured approach to implementing the complex software necessary to provide the control room function for the high-voltage power supply system of ENHER of Barcelona. In this case, we further developed the idea of monitors (invented by C.A.R. Hoare) and applied them as a fundamental means of structuring the software. Due to this structuring the software system could be developed and maintained by a very small staff. I also contributed to the development of a formal means of specifying and generating program code for the human-machine interface.

Automatic Train Control

During the years 1976-77, I was engaged as a consultant in developing the architecture for the implementation of the world's first microprocessor based Automatic Train Control system. This on-board system for signal interlocking and speed control was developed by Standard Radio of Sweden (owned by ITT) and utilized in the Swedish railways (SJ).
This work resulted in a highly simplified and robust architectural solution that has since 1980 been utilized successfully on most all locomotives in Sweden. Via an engineering approach to the software, based upon viewing the system as being continuous time-driven (as opposed to discrete event-driven), the solution resulted in a surprisingly small amount of software code. The first version occupied just over 10K bytes of read-only memory. Even as successive versions have been implemented the code size is extremely small. The current system that significantly extends the system functionality occupies about 27K bytes of read-only memory.
During the first thirteen years of its existence in virtually all locomotives (over 1000) in Sweden not a single line of program code was changed. Probably a world's record! It has successively been upgraded to new versions for faster trains. The system has been implemented for railways in Norway, Finland, Malaysia and Australia. The Standard Radio developed ATC system solution is now owned by Ansaldo of Italy. Ansaldo utilized the architectural concepts in the implementation of the New Jersey Transit Automatic Train Protection system. The key to providing this small but powerful solution (about 30K bytes of ROM memory in the latest Swedish version) was viewing the system as being continuous in time instead of being discrete. This engineering view paid high dividends and has resulted in an important safety facility for trains.
At Standard Radio, I worked with Sivert Wallin and Berit Brintse who later continued to further develop the technology in a company called Teknogram. In addition to providing continued support to Ansaldo, they used the same architecture to generate train operator training simulators for a variety of customers.
The success of the Automatic Train Control system architecture stimulated personal research related to the generalization of the approach in building real-time control systems that later was exploited in the following projects.

Distributed Control Systems for Vehicles

The architectural concepts developed in the Automatic Train Control project and my further development of the concepts and principles were inputs to a research project initiated by the Swedish National Board of Technical Development (NUTEK). I was a member of the research team that further developed the concepts and principles of the continuous time-driven solution I had introduced. The solution resulted in accommodating both continuous and discrete functions in a distributed vehicle control context. In this effort these two approaches became known as the

Red and Blue application functions. (This identification is now used internationally in discussing automotive distributed control networks). As a part of the research work, I introduced the concept of Software Circuits as an abstraction for developing program logic. It is based upon viewing software via a hardware-oriented paradigm where short well-defined program sequences are provided to perform circuit like transformations of input variables to output variables. This, once again radically reduces software complexity. The project team included Jan Torin, Hans Hansson, Michael Strömberg and Sven Larsson
I also worked closely with Kurt-Lennart Lundbäck of Arcticus Systems in this project. He integrated many of the concepts and principles of deterministic solutions into a real-time operating system called Rubus. Further, his company has developed an approach for model-based design utilizing the concept of Software Circuits called Rubus ICE.

Limited Slip Coupling Device

Along with Arcticus, I became involved in an innovative project, led by Anders Cederberg, to develop a coupling device for vehicles for Haldex in Landskrona, Sweden. Haldex had purchased a patent related to controlling the degree of stiffness between the front and back axels of vehicles. Then they received an order from Volkswagen to develop a computer-based control system to implement this function. Rubus was used as the basis for controlling this combined mechanical, electronic, and software based product. At the time, I had learned of the ISO/IEC 12207 for the life cycle management processes related to software systems. I tailored a version of this so that it could be applied not only to software but to the other technologies as well. (Note: later I participated in the development of ISO/IEC 15288 that provides processes for all types of man-made systems.)
It took some time for Haldex to achieve a successful product, but now this device is used by virtually all automotive manufacturers around the world and provides a very useful safety and efficiency function for four-wheel drive vehicles. The system is now owned and further developed by BorgWarner.

Swedish Defence Materiel Administration (FMV)

In 1986 I resigned from my professor's position at Linköping and moved to Stockholm where I founded my own consulting company, Lawson Konsult AB. Initially, I was contracted by the Technical Director of FMV, Ingemar Carlsson for about half of my consulting time. Based upon experiences with safety critical real-time systems, I developed an approach to assessing safety properties in real-time systems for the Swedish Defence Materiel Administration. In addition to this project, I have been involved in assessing the potential of a new computer architecture proposed by Gunnar Carlstedt as well as in assisting in the mid-life update of the Viggen jet fighter plane developed by Saab. This latter assignment involved producing documentation of the computer control system since the original solution did not provide sufficient information for those involved in updating the control system. This is the backside of small projects done by a small group of competent engineers. The knowledge lies mostly in their heads and can lead to maintenance problems.

Technical Committee on the Engineering of Computer-Based Systems

I have made significant professional contributions related to spreading knowledge on the Systems Engineering of Computer-Based systems. This has been accomplished by publications

as well as active participation in organizations in Sweden and internationally including the IEEE Computer Society Technical Committee on the Engineering of Computer-Based Systems (ECBS) and INCOSE. I was a founder of the Technical Committee on the Engineering of Computer-Based Systems and served as its chairman (1999-2000).

Reflections

My experiences with embedded real-time systems from this phase convinced me that many commercially available event-driven real-time operating systems add significant complexity and non-determinism into applications. There are better solutions for a variety of embedded system applications. The experience with ATC (Automatic Train Control) was a significant development that then was exploited in the automotive context and in other commercial products from Arcticus Systems.

During the Nutek sponsored research project we had cooperation with Professor Herman Koptez at the Technical University of Vienna. Herman shared the view concerning time-driven deterministic solutions and he eventually created the company TT-Tech that successfully markets products based upon this approach.

Academic Career (Continued)

As noted above, during the period 1974 to 1979, in addition to active consulting at ENHER and with ITT I was both a part time and guest professor at Linköping University, at the University of Stuttgart, at Politecnica de Barcelona, and at the Royal Technical University in Stockholm (KTH).

In 1979 I was appointed as a full time Professor of Computer Systems at Linköping University. There, in addition to both undergraduate and graduate education efforts, I led research efforts in the area of asynchronous circuits as well as software engineering. In 1983, I together with Erik Sandewall established, IDA the first Computer Science institute in Sweden that has become one of the foremost in Europe. Upon leaving IDA in 1986, a Lawson Prize was established to honour significant international contributions from the faculty or staff.

I can mention that during 1984 I had a sabbatical leave from Linköping and was guest professor at the University of Malaya. During this time, I together with the Dean of Engineering Tengku Azzman and with the support of the Prime Minister Dr. Mahathir established the Malaysian Institute of Microelectronic Systems (MIMOS). The goal was to improve knowledge and capabilities in electronics and software engineering so that Malaysia could lift itself up on the value chain instead of simply being a provider of inexpensive assembly labour for microelectronic companies. Today, MIMOS is a company employing over 600 people and has placed Malaysia as a leading developer and user of ICT technologies. One month in 1984 was also spent as a guest professor at Keio University in Yokohama, Japan where I continued a project with Kee Yamamoto, a former Linköping University guest researcher.

During my visit in Malaysia I came upon a new idea of providing graduate education for foreign students in Sweden. Thus upon returning, I successively developed proposals of how to provide a program that would attract foreign students and faculty in providing Masters Degree education in English. The proposal attracted significant attention of the University Chancellor and many of the leading universities in Sweden. So, I took a leave of absence in 1985-86 and further developed the concept that became known as Swedish International University. It was a virtual university developed as a catalogue and presented courses from participating institutions; namely, Uppsala, Lund, Royal Technical University, Chalmers, Linköping, Agricultural College, Stockholm School of Economics (Handelshögskolan) and Lulea. I had significant support from the Swedish Employer Organization (SAF) and the concept aroused significant interest.

Unfortunately, the funds necessary to sustain the concept where never provided. All organizations, including the Minister of Education thought it was a great idea, but no champion arose to provide the necessary funding. It is interesting to note that today the ERASMUS program provides some similar possibilities.

Phase 3 (1996 – Present) – Complex Systems

With a solid background in the computer industry as well as experiences in designing and developing several successful embedded systems, I was well prepared to enter this third phase where my focus has widened to the general area of Complex Systems. It is certainly true that hardware and software systems always exist in some wider system context from which requirements are provided. So, this was a natural step in my career.

International Standards

I joined the ISO/IEC JTC1 SC7 WG7 effort aimed at developing an international standard for System Life Cycle Processes at its beginning in 1996. My participation as Head of the Swedish delegation was at the request of Dr. Raghu Singh, at that time convener of WG7 and was sponsored by the Swedish Defence Materiel Administration and the Swedish National Board for Technical Development (NUTEK).
After various practical difficulties in this project, a restart was made in 1999. At that time, Stan Magee became the new convener and Stuart Arnold was appointed as the sole editor of the emerging ISO/IEC 15288 standard. Further in an election involving the fourteen member countries, I was elected to be the architect of this standard. From this point, Stuart Arnold and I worked very closely in pinning down the concepts and principles upon which ISO/IEC 15288 is based. Further, we worked on spreading the concepts in WG7 and elsewhere and defended them during the following years that led to its initial publication in 2002.
The experience in WG7 offered the opportunity both to learn about standards and the standardization process as well as to work on applying the standards of WG7. After learning about the ISO/IEC 12207 standard on Software Life Cycle Processes, I helped three companies in tailoring the standard to meet their needs for life cycle processes.
Cambio AB - A medical information system provider. The processes of 12207 were tailored as a means of defining the details of agreements between Cambio as a supplier and the hospitals and clinics to which they supply their products and services. It defined roles and responsibilities over the range of software related processes.
Gambro AB – A medical equipment supplier specializing in kidney dialysis equipment. The tailoring of 12207 was required for Gambro to attain Federal Drug Administration approval of its software development life cycle. This was successful and the FDA reviewer at that time stated that it was the best set of software life-cycle processes he had experienced. It is also interesting to note that this stimulated the development of a specialized standard for software in medical devices that is based upon 12207.
Haldex AB – Supplier of a Limited Slip Coupling Device for four-wheel drive vehicles. As described above, the concepts and principles of the train control application were used in this product. In addition, the 12207 standard was tailored in order to deal with the system as well as software aspects. The system being composed of mechanical hardware, electronic hardware, and software elements. (This was prior to the existence of 15288).
While working with 12207 and during the earlier developments related to 15288, I cooperated with Johan Bendz at the Swedish Defence Materiel Administration (FMV) on how these standards could be of assistance in implementing Change Management. Johan also joined into

the SC7 standardization activities and has had a convener role in the ISO/IEC 42010 standard on Architecture Description.
Since the 15288 standard can be applied to any type of hard and/or soft man-made system, Stuart Arnold and I decided to write a joint paper in order to convey the message of a changing scope for systems engineering. It points to the fact that the standard can and is showing to have an impact upon business management as well as systems engineering.
The Swedish Defence Materiel Administration (FMV) was the first organization in the world to implement ISO/IEC 15288. I served as a mentor to the implementation project and worked closely with Eddie Lindquist.

Consulting Partner at Syntell

In 1999 I started my cooperation with a Logistics and Systems Engineering company in Stockholm called Syntell AB. We have co-operated in spreading the knowledge of systems engineering and in particular that ISO/IEC 15288 standard in Sweden and internationally. I have worked closely with Tom Strandberg, Daniel Falk and others. The managing director of Syntell Mats Bjorkeroth also asked my to join the board of directors and I served in this role for five years. I have helped Syntell open new markets for their services, in particular in Russia where there was an increasing interest in applying the 15288 standard where Asmus Pandikow assisted in this effort.

Continued Academic Connection

As mentioned above, I had a parallel academic career, beginning in 1967, when I joined the Polytechnic Institute of Brooklyn as an Associate Professor in the Electrical Engineering Department. I have continually had contact with a variety of universities in the USA, Europe and the Far East.
In the early 1990s I assisted the Mechanical Engineering department led by Sören Andersson at the Royal Institute of Technology (KTH) in making the transition to Mechatronics. I held a graduate course, advised students and helped them establish contact with Professor Herman Koptez and the University of Vienna. I also assisted graduate students in their research including the current Professor of Embedded Control Systems, Martin Torngren.
In 2002, Professor Dinesh Verma requested that I become an Academic Fellow at Stevens Institute of Technology and to develop a new graduate course on Systems Thinking. In response to this request, I developed and have delivered the academic course at Stevens as well as at other universities and as professional development training in organizations in various countries.
As the course material matured, successive versions where produced and in 2010 I published a book entitled: *A Journey Through the Systems Landscape* as a Print-on-Demand book published by College Publications of Kings College, UK. The book is aimed at developing a discipline independent capability to think as well as to act in terms of systems. This is accomplished by unifying concepts of principles of systems thinking and systems engineering. The book has been translated to French by Brigitte Daniel Allegro and to Russian by Victor Batovrin.
The course, based upon the book, can and has been taught for scientific, engineering, military, medical and management related participants with a wide diversity of backgrounds. The course develops a common conceptual view that can be applied for any type of man-made system (hard, soft, or mixtures thereof). The study of the dynamics of systems thinking is based upon work of Peter Senge, Peter Checkland, and others. Through systems thinking, it is possible to identify system problems and opportunities. After identification of problems and/or opportunities,

the acting part is accomplished by prudent decision-making and identifying required structural changes in systems. The acting part is presented in the context of the life-cycle management of systems based, for example, upon the ISO/IEC 15288 standard. Coupling the thinking and acting together provides a paradigm for change management.

The Systems Series

After publishing *A Journey Through the Systems Landscape* and based upon requests from other authors, I decided to start a series of books that would focus on a wide variety of aspects related to systems. In this effort, I arranged cooperation with the School of Enterprises and Systems at Stevens Institute of Technology and the Bertalanffy Center for the Study of Systems Science. Thus, respectively Jon Wade and Wolfgang Hofkirchner joined my editorial team. I also acknowledge the enthusiastic support for multiple volumes provided by Mats Persson.

Thus far, nine volumes have been produced and several more are on their way. Of particular interest is the Volume 7 *Software Engineering in the Systems Context* that is co-edited by Ivar Jacobson and myself. Due to the increased central role of software in today's systems this volume is very timely and provides perspectives on the role and relationships between Software Engineering and Systems Engineering.

Some of My Publications

I have published over 100 papers at various conferences, in professional journals, as technical reports. Further, I have provided several contributed chapters and published books. Here I present those that I personally consider to be the most important.

- "The Use of Chain List Matrices for the Analysis of COBOL Data Structures", Proceedings of the ACM Conference 1962, Syracuse, New York.
  (First publication of multi-linked data structure algorithms as cited by Donald Knuth in his classic book The Art of Computer Programming, Volume 1: Fundamental Algorithms)
- "PL/I List Processing", Communications of the ACM, Volume 10, Number 6, June 1967.
  (First publication of the pointer variable concepts)
- "Programming Language Oriented Instruction Streams", IEEE Transactions on Computers, Volume C-17, Number 5, May 1968.
  (An early presentation of the relationship between programming languages and processor instruction set architectures)
- "Functional Characteristics of a Multi-Lingual Processor", IEEE Transactions on Computers, Volume C-20, July 1971. (Co-author: Burton K. Smith).
  (Description of the first truly general-purpose microprogrammable processor, the MLP-900 that was an ARPA Net resource in the early 1970s)
- "Advantages of Structured Hardware", Proceedings of the Second Annual Symposium on Computer Architecture, Houston, Texas, January 1975. (Co-author: Dr. Bengt Magnhagen).
  (Description of the Datasaab FCPU (Flexible Central Processing Unit) that utilized asynchronous control of the microarchitecture - Selected as the Best Paper Award at the Symposium)

- "Function Distribution in Computer System Architectures", Invited paper appearing in the Proceedings of the Third Annual Symposium on Computer Architecture, Clearwater, Florida, January 1976.
  (Describes the importance of taking a holistic view of computer architectures in respect to the distribution of functions between hardware and software levels)
- "Philosophies for Engineering Computer Based Systems", IEEE Computer, Vol. 23, No. 12, pp. 52-63, December, 1990.
  (Discusses the importance of identifying and utilizing driving concepts and principles in developing and sustaining complex computer-based systems and includes 3 case studies)
- "CY-CLONE - An Approach to the Engineering of Resource Adequate Cyclic Real-Time Systems", Real Time Systems, The International Journal of Time-Critical Computing Systems, Vol. 4, No. 1, February, 1992.
  (Based upon experiences with the Automatic Train Control system, this extends the use of a deterministic approach in the context of multi-processor systems – Can be used as a basis to provide safety critical solutions using multi-core microprocessors)
- Parallel Processing in Industrial Real-Time Applications, Prentice-Hall series on "Innovative Technologies", ISBN 0-13-654518-1. 1992.
  (The result of a sponsored project to introduce engineers in Sweden to the utilization of parallel and distributed processing in real-time applications. Includes contributions by Bertil Svensson and Lars Wanhammar)
- "BASEMENT: A Distributed Real-Time Architecture for Vehicle Applications", Real Time Systems, The International Journal of Time-Critical Computing Systems, Vol. 11, No. 3, November, 1996. (Co-authors: H. Hansson, M. Strömberg, and S. Larsson).
  (Describes the project involved in developing a distributed architecture for vehicles and includes the introduction of the concept of "software circuits")
- "Salvation from System Complexity", IEEE Computer, Vol. 31, No. 2, Feb 1998, pp 118-120.
- "Infrastructure Risk Reduction", Communications of the ACM, Vol. 40, No. 6, June 1998, pp120.
- "From Busyware to Stableware", IEEE Computer, Vol. 31, No. 10, Oct 1998, pp117-119.
  (All three of these brief contributions point to the problems that have arisen as the result of using the limited capability microprocessors from the 1970's in today's complex computer system applications)
- "Rebirth of the Computer Industry". Communications of the ACM June 2002/Vol. 45, No. 6.
  (Describes in detail the problems caused by the transition to microprocessor based solutions in respect to the complexities that have evolved resulting in unreliable software solutions. The article points to a path for rectifying these problems within the computer industry. Unfortunately, this has not happened and probably will not happen in my lifetime – We need some real catastrophe's to initiate such a transition)
- "Twenty Years of Safe Train Control in Sweden", Proceedings of the International Symposium and Workshop on Systems Engineering of Computer Based Systems, Washington, DC. April, 2001. (Co-authors: S. Wallin, B. Bryntse, and B. Friman).
  (Documents the result of this very successful Automatic Train Control system, the world's first microprocessor based solution)
- "Viewing Systems from a Business Management Perspective: The ISO/IEC 15288 Standard", The Journal of Systems Engineering, Vol. 7, No. 3, September, 2004. Co-author: Stuart Arnold

(Describes the change role of Systems Engineering from traditional product orientation towards the needs of organizations and enterprises)

- "A Journey Through the Systems Landscape", College Publications, Kings College, UK, ISBN 978-1-84890-010-3. 2010.

(A book that introduces the use of Systems Thinking, Change Management and Systems Engineering as proper approaches to dealing with complex systems. Provides a number of useful concepts, principles and paradigms aimed at providing personal as well as group (including organization) competence)

- "Software Engineering in the Systems Context"

(A book edited by Ivar Jacobson and myself that presents various perspectives in dealing with the relationship between Software and Systems Engineering. In addition to their contributions to the book many well known Software and Systems Engineers have contributed very useful chapters)

Final Reflections

Well – it has been a long and interesting career. Certainly it has been non-deterministic since life for me has been a happening with changes in my place of work, the orientation of my work, the countries I have worked in and the family I have shared my life with. I have learned a lot and have dedicated my final years to sharing the knowledge I have accumulated. But it is never too late to learn so in parallel I keep learning from colleagues, my consulting activities and participants in my academic, professional development courses as well as my coordinating editor role of the College Publications Systems Series.

I have presented seminars entitled: Experiences and Reflections of a Computer and Systems Engineering Pioneer for a variety of academic and professional organizations.

