Formal Methods: My 50 Years of Work[*]

Dines Bjørner
DTU: Technical University of Denmark
Fredsvej 11, DK-2840 Holte, Danmark
bjorner@gmail.com www.imm.dtu.dk/~{}db

It all began at the IBM Lab. in Vienna, 1973–1974. I will reminisce of how VDM, the Vienna Development Method, emerged. We were to develop a compiler for PL/I. In Feb. 1974 IBM HQ curtailed this and related projects. So three of the co-developers of VDM went abroad. Peter Lucas to the US, Cliff Jones via Brussels back to the UK, and I to Denmark. With students and later, at Dansk Datamatik Center (DDC), we then developed compilers for CHILL and Ada according to the IBM Lab. Vienna ideas. I will survey those developments. They led to the design of the RAISE (Rigorous Approach to Industrial SE) Method – with RSL (the Raise Specification Language). I will survey that development. As UN Director of the UN University's Intl. Inst. for Software Technology, in Macau, 1991–1997, we pursued Formal Methods and their theoretical underpinnings, and applied Formal Methods to the development of systems for railways (in China), finance ministry (Vietnam), country wide telephone system (The Philippines), multi-script documents (for Mongolia), etc. I will survey these projects – emphasizing how Domain Modeling emerged. I will then outline recent work, since my retirement, at 70 in 2007, in Domain Science and Engineering.

All this should take 30 mins.

I will end the talk by by revealing my emergence as a scientist from [merely] being an engineer/researcher – and by engaging the audience in a discussion of why our colleagues [may not] take on to Domain Science & Engineering.

Contents

[*]Invited talk for FROM 2024, the Working Formal Methods Symposium, The West University, Timişoara, Romania, 16–19 September 2024. The presentation was eventually cancelled due to illness.

Submitted to:
FROM 2024

1 Introduction

Allow an 86 year old man[1] to, first, reminisce, Sects. 4–10, and, finally, put forward some personal observations, Sects. 11.1.

But, to a beginning, some remarks.

I have been blessed with both a long life, and a reasonably productive life. Choices, as to jobs and changes of job venues, appear to have been reasonable choices – but not always traditional. People for and with whom I worked were all quite unique – as You shall read.

The main sections of this paper, Sects. 4–10, should be seen on the background of the characterizations of a number of computing-related terms. I outline these in Sect. 2.

• • •

The paper is not a computer science but a software engineering paper. It is a paper that touches upon some aspect of the history of computing. One may characterize it as a 'memoir' paper.

It is written for the following reasons: (i) historians of computing may find a few "nuggets" to complement their studies, and (ii) to touch upon an issue of 'researcher vs. scientist', cf. Sect. 11.1. It was fun to write !

2 Some Delineations

The delineations of this paper are mine. You may not agree with them. But You shall have to accept them for the course of this paper. This vocabulary developed, as from the early 1980s, as a result of my interaction with members of the IFIP Working Group WG 2.3,[2] and my reading of [65, *Michael A. Jackson*].

Computer Science: *The study and knowledge of the phenomena that occur within computing devices*[3]. I am not a computer scientist.

[1]This paper was written in the summer of 2024 – and I was born on 4 October 1937.

[2]Notable WG 2.3 members in this respect were: Sir Tony Hoare, Edsger Diijkstra, John Reynolds, Michael Jackson, Brian Randell and Cliff Jones.

[3]*Automata Theory, Formal Languages, Complexity Theory, ...*

Computing Sciences: *The study and knowledge of how to construct those phenomena*[4]. I am primarily a computing scientist (cum software engineer).

Programming Methodology: To me, a synonym for computing science[5].

Method: By a method we understand *a set of principles and procedures for applying a set of technique and tools in order to achieve the construction of some artifact.*

Methodology: *The study and knowledge of one or more methods.* I have, since my Vienna days, been and am "nuts about" 'methodology'.

Formal Methods: *A formal method has its main techniques and tools be based on mathematical insight.*[6] I am definitely a formal methods person.

IT: *A material universe of hardware (electronics, etc.): computers, modems, printers, screens, tactile instruments.* IT focus on larger memories, smaller size, faster execution, less heat generation, less power consumption, etc.; that is physical properties. I was in IT between 1962 and 1965.

Informatics: *An intellectual universe of computer and computing science.* Informatics focus on appropriate computing, correctness, etc.; that is on mental concepts. I have been in Informatics since 1965.

The Triptych Dogma: *Before software can be designed we must understand its requirements. Before requirements can be prescribed we must understand its domain. So we must study, analyze and describe the domain.* I have spent the last 30 years studying domain science and engineering. The formulation of the 'Dogma' came about around 2015.

Machine: By the machine we understand *the hardware, i.e., a computing system.* This definition follows that of M.A. Jackson [65].

Domain: By a domain we shall understand *a rationally describable segment of a discrete dynamics fragment of a human assisted reality: the world that we daily observe – in which we work and act, a reality made significant by human-created entities.*

Requirements: *By requirements we shall generally understand a condition or capability needed by a user to solve a problem or achieve an objective [64].*

By software requirements we specifically shall understand a document which prescribes desired properties of a machine (computer): ***What*** *the machine shall (must; not: should) offer of functions and behaviours, and what entities it shall "maintain"*

[44, Chapter 9] makes precise my understanding of what 'software requirements' are.

Software: *By software we understand not only codecode that may be the basis for executions by a computer, but also its full development documentationdevelopment!document: the stages and steps of application domain descriptiondomain!description, the stages and steps of requirements prescriptionrequirements!prescription, and the stages and steps of software designsoftware!design prior to code, with all of the above including all validationvalidation!document and verificationverification!document (incl., testtest!document) documents. In addition, as part of our wider concept of software, we also include a comprehensive collection of supporting*

[4]*Program design (functional, imperative, logic, parallel), Domain Engineering, Requirements Engineering, Software Design, ...*

[5]The distinction between 'computer science' and 'computing science' was "a feature" of IFIP Working Group WG 2.3: Programming Methodology

[6]A common programming tool is that of a programming language – which, to be formal, must have a formal syntax and a formal semantics.

documents:support!document training manualstraining!manual, installation manualsinstallation!manual, user manualsuser!manual, maintenance manualsmaintenance!manual, and development and maintenance logbooks development!logbookmaintenance!logbook So: Software, as documentation, comprises many parts

[28, *SE3*].

3 Pre-History: 1982–1973

Two weeks before my graduation as an M.Sc.E.E., January 1982, I went to the compulsory military service conscription test. On a Thursday morning. I had expected to be conscripted and planned then to join the Royal Danish Navy. But, due to my cleft pallet, I was, however rejected. On my way home I passed the IBM HQ in Copenhagen. Remembering that I had read in one of their advertisements in the Technical University student monthly, that they were hiring for their newly established Nordic Laboratories, in Stockholm, Sweden, I went in. The next Tuesday I could write in the Nordic Lab. directors' guest book: *"Come what come may, time and the hour runs through the roughest day" [Shakespeare, Macbeth, Act 1, Scene 3].* I was hired. Started March 1, 1962. Worked for some half year in designing stochastic and adaptive control, according to my boss: Karl Johan Åstróm's research, for Sandviken's 18-9 Chromium Nickel steel works. Not really my area of interest, though. And it must have shown through. I was then transferred to help design the hardware logic for a data communications "gadget", the IBM 1070, for the IBM 1400 series of computers. Transferred to complete the design at IBM's Systems Design and Development centre in San Jose, California, in Dec. 1963. Was doing pretty well, and was asked to also design a data communications front-end to the IBM 1800 computer. Did that well. And embarked on a three year leave-without-pay to get a Ph.D. back at my alma mater. During my Ph.D. studies I met, at various summer schools such notables as Donald E. Knuth, Christopher Strachey, Niklaus Wirth, Tony Hoare, Edsger Diijkstra, Robert Floyd, Calvin C. Elgot, and many others. I became "hooked" on, first computer, then computing science. After my Ph.D. the IBM Nordic Lab. kindly facilitated that I was assigned to Gene Amdahl's supercomputer project at the IBM Advanced Computer Systems Lab., Menlo Park, California. Worked there for six months. Then IBM HQ closed down the ACS/1 effort and I was kindly transferred to IBM Research in nearby San Jose. After a while I got to work with John W. Backus, after he had had no-one working for him in some 10 years. I implemented various versions and generations of his variable-free functional programming, FP and FFP, languages. Moved on to work with Edgar F. ('Ted') Codd on pre-versions of SQL. Also as the first one working for him in some 10 years ! After a year and a half, on a mid December 1972 Thursday morning, I received a telephone call from Peter Lucas, the IBM Vienna (Austria) Laboratory. So I began there May 1st 1973.

• • •

My thesis work was [7, *A Theory of Finite State Transducers*] and resulted in two publications: [8, *A Flow Mode, Self-Steering, Cellular Multiplier-Summation Processor*] and [13, *The Synthesis of Finite State Syntax Directed Top-Down and Bottom-Up Transducers*].

During my almost three years at IBM Research I worked on diverse topics: First some [11, 12, *Flowchart Machines* and *Folded Syntax- and Recursive Flowchart-Machines*]; then [10, *Finite State Automaton Definition of Data Communication Line Control Procedures*]; then [14, *On the Definition of Higher Level Language Machines*]; then [15, *Finite State Tree Computations (Part I)*]; and finally [6, *The GAMMA-0 Relational Data Base Interface Specifications of Objects and Operations*]. I also edited [9, Data Description & Access. *1st ACM SICFIDET Workshop*].

4 The Origins: The IBM Vienna Laboratory, 1973–1975

We, Peter Lucas, Hans Bekič, Cliff Jones, Wolfgang Henhapl and I, were to develop a formal description of PL/I (without concurrency). The IBM Vienna Laboratory had, in the 1960s, developed a number of formal definitions of PL/I (with concurrency) `ULD` [3, 4, 5, 1, ULD I,II,III]. They were based on Peter Landin's SECD concepts [67, 69, 68]. Time had come to express programming language more abstractly, less mechanically. The recent work of Dana Scott and Strachey and Strachey's colleagues at Oxford [89, 90, 78, 77, 79, 83, 80, 81, 82] became a major inspiration. Dana Scott visited the IBM Vienna Lab. regularly – his wife having Viennese roots.

So we embarked on the development. In the early stages focus was on the specification language. It became known as Meta IV – in allusion to there having been three versions of the VDL, the Vienna Definition Languages, for the ULD I-II-III. The approach we took in development became known as VDM, the Vienna Development Method. We would each be given a subset of PL/I to define. Each day, typically from an hour and a half before noon till we left in the late afternoon, we would work (scribble) on print-outs of yesterday's work. We would correct mistakes and add new texts. These would be handed over for transcription, by secretarial staff, at IBM Austria's HQ. Next morning, at 8:00 am new print-outs would be on our desks. We were assigned, on a rotating basis, to review a colleague's work – shifting colleagues' throughout the week. We would then mark up what we considered problematic points. At 10:00 am we would meet in the coffee room and discuss some such problems before handing over the corrections to our colleagues.

Around September 1974 we issued [1]. It was at that time, at the home of Peter Lucas, celebrating the event, that our boss, Kurt Walk, asked: "*and what is the name of the specification language and of the development approach ?*" that Meta-IV and VDM was decided upon. Later the name 'Meta-IV' was changed into 'VDM SL' [74].

IBM, having curtailed, in February 1974, the larger corporate-wide 'Future Systems Machine' project that our PL/I development project was part of, was flexible enough to let its formal PL/I definition continue to a natural end. As it indeed did. So the group dissolved. After a few iterations several of us left: Henhapl to the Technical University of Darmstadt, Germany; Cliff Jones via the IBM Systems Research Institute in La Hulpe, near Brussels, Belgium, to Oxford University (to work for his D.Phil. – having Tony Hoare as his advisor); Peter Lucas to IBM in the US; and I – via a one year guest professorship at Copenhagen University – to The Technical University of Denmark.

• • •

What made the IBM Laboratory in Vienna such a special place ?

For indeed, of all the places I have worked, the 1960-70s IBM Vienna Laboratory stands out.

When my wife and I visited, our first visit to Austria, the Lab., in March 1973, to get acquainted with the Lab., its people, to settle the employment, and to get a place to live, I bought, on our stopover, in NYC, from San Francisco, a Saturday evening, at around 9pm, at the Doubleday Bookstore on 5th ave., the book *Wittgenstein's Vienna* by Allan Janik and Stephen Edelston Toulmin, in 5 copies. It was fresh off the press, March 1973. When Heinz Zemanek, Kurt Walk, Peter Lucas and Hans Bekič got their copies, at this first visit, they were clearly 'taken'.

Yes, this was Vienna. A city of culture.

We either "accidentally', but often, met our colleagues at either the Vienna State Opera, or at the Burg-theater, or in the Musikverein.

That culture "spilled" into the Laboratory work, in no small measure, to its Director, Heinz Zemanek – who had hand-picked his engineering cum scientific staff – from his previous work at the

Techn. Univ. of Vienna. Philosophy, the study of language, in all its forms, were a Laboratory trademark. No wonder the Laboratory was able to attract, as day- or week-visitors, such as Dana Scott, John Reynolds, Lotfi Zadeh, and many others.

IBM Research in San Jose was, except for John Backus, Ted Codd, and TienChi Chen, in the 1960s and early 1970s, in comparison, an intellectual "waste land".

• • •

With VDM SL I was given a tool with which to analyze and understand computing concepts. So over the years I expressed my understanding of many such concepts. They were first published in [52]. That is, after the engineering/scientific [cum research] work on co-designing VDM (and the formal definition of PL/I). followed, for me, years of the engineering use of VDM SL. It was not until my yearly years at DTU that I started, more-or-less subconsciously, studying "how to construct large-scale, formal specifications".

5 Early Years at The Technical University of Denmark, 1976-1980

I had the gratifying experience of a great many of my department colleagues also following my courses of abstract software specification, the `VDM` style. Some of "this" spilled over into their courses !

Tom Østerby, the manager ("head") of the department, followed-up by master-minding a complete redesign of our course programme: Courses in *imperative programming*, in the great, fine style inherited by those colleagues who had formerly been at the pioneering *Regnecentralen*, noted for its *Datalogy/Datamatics* views on programming, as then led by Peter Naur, now were followed up by separate courses in *functional*, *logic* and *parallel programming*. And I was able, in the 1980s, to build up my own set of three courses – later, much later, embodied in the three, corresponding volumes of [26, 27, 28].

In the first years of my chair at DTU I produced a number of reports. Most were published. Notably [17, 16, *Programming Languages: Linguistics and Semantics* and *Formal Development of Interpreters and Compilers*]. They contain the essence of my early courses on formal software development. And they laid the foundation for our subsequent work on the `CHILL` and `Ada` compiler projects.

With Cliff Jones [53, *The Vienna Development Method: The Meta-Language*, LNCS 61] was published.

It, possibly, is a first such book on formal methods.

My contributions to this book included [18, *Programming in the Meta-Language: A Tutorial*], [19, *Software Abstraction Principles: Tutorial Examples of an Operating System Command Language Specification and a PL/I-like On-Condition Language Definition*], and [20, *The Vienna Development Method: Software Abstraction and Program Synthesis*].

Publication of [53] led to our organizing the *Lyngby Winter School* and the subsequent publication of [21, *Abstract Software Specifications* LNCS 86]. It, possibly, was a first such event on formal methods. Lecturers, besides the organizers, included: Rod M. Burstall, Ole-Johan Dahl, Balint Dömölki, Barbara Liskov, Peter Lucas, David Park, Gordon Plotkin, Joe Stoy, and Stephen N. Zilles.

• • •

At noon the professors of a number of the electrical engineering departments lunched together. Prof. Asger Kierbye Nielsen discussed with me issues of language semiotics: syntax, semantics and pragmatics. Kierbye was a true scholar and scientist. He had an agenda – which he proceeded to reveal, in due time. The ITU (UN's Intl. Telecommunications Union, then called CCITT [Consultative Committee for International Telegraphy and Telephony]) was about to design a Communications HIgh-Level Language, now known as CHILL. Would I, with my colleagues and students, be interested in taking part in that effort ? So for a couple of years, 1978–1980, I traveled the world: attending CHILL meetings all over Europe, in Japan, Australia, etc. My colleague Hans Bruun followed up on the emerging CHILL description – modeling its static semantics in VDM. In the course of that he pointed out that some context conditions would be cumbersome to implement – and to little effect. So formal definition work helped design CHILL – by, e.g., removing those contect conditions. Eventually our group, with Dr. Hans Bruun, Søren Prehn and Peter L. Haff – our M.Sc. students – issued what became ITU's formal description of CHILL. Ivar Jacobsen, of UML fame, at L.M.Ericsson, the Swedish telephone system company, hired me to help them make sure that their language views on concurrency became part of CHILL – as did two other views !

As a result of our ongoing CHILL work, made realistic by the substantial contributions of Dr. Hans Bruun, we turned our application of the Vienna compiler methodology to the emerging US Department of Defense's (DoD's) Ada language design. After a number of pre-thesis term projects five M.Sc. students produced [56, *Towards a Formal Description of Ada*, LNCS 98].

My colleague, Prof. Christian Gram, then took the initiative, with me suporttng him, to form, with 10 Danish IT companies, the Dansk Datamatik Center, DDC.

6 Dansk Datamatik Center (DDC) and DTU, 1980–1991

6.1 DDC

DDC was to help propagate formal methods in the Danish software industry. It tried to do so – for 10 years. I was the nonsalaried scientific director of DDC during its 10 year existence, 1979–1989.

Our main formal methods projects were

- the CHILL compiler project [62],
- the [EU co-sponsored] Ada compiler project [57, 73],
- the [EU sponsored] Formal Definition of Ada (with, notably, Profs. Egidio Astesiano and Gianna Reggio of the Univ. of Genoa, Italy) [58], and
- the [EU co-sponsored] RAISE R&D project – in collaboration, notably, with the BT [British Telecom]. The latter included such people as Robert Milne, Tim Denvir and Chris W. George [60, 61].

Quite a substantial national formal methods effort.

The Ada compiler project resulted in the creation of DDC International, a commercial software house specializing in Ada compilers. DDC International moved to Phoenix, Arizona, the US, in the mid/late 1980s, and became DDCI Inc. DDC derived some income from DDCI Inc.

The DDC Chill and Ada compilers were developed according to the the *Software Development Graph* [23, 24, 25] shown in Fig. 1. First note that the development has three major phases: describing the domain, prescribing the requirements and designing the software. Then note that each phase has many separate stages: In the domain description phase one first develops an abstract syntax for programs. Then, in three "independent" steps a static, a dynamic sequential and a dynamic parallel semantics are developed. In the requirements prescription phase In the software design phase Note that we have annotated some of the steps mentioned above: $\boxed{\text{A}}$, $\boxed{\text{B}}$, $\boxed{\text{C}}$, $\boxed{\text{D}}$, and $\boxed{\text{E}}$. These refer to papers, i.e., theoretical foundations, supporting the development at hand. We could have listed many more such supporting theories ! The **R**→s indicate [some of the] places where explicit requirements to the compiler is expressed.

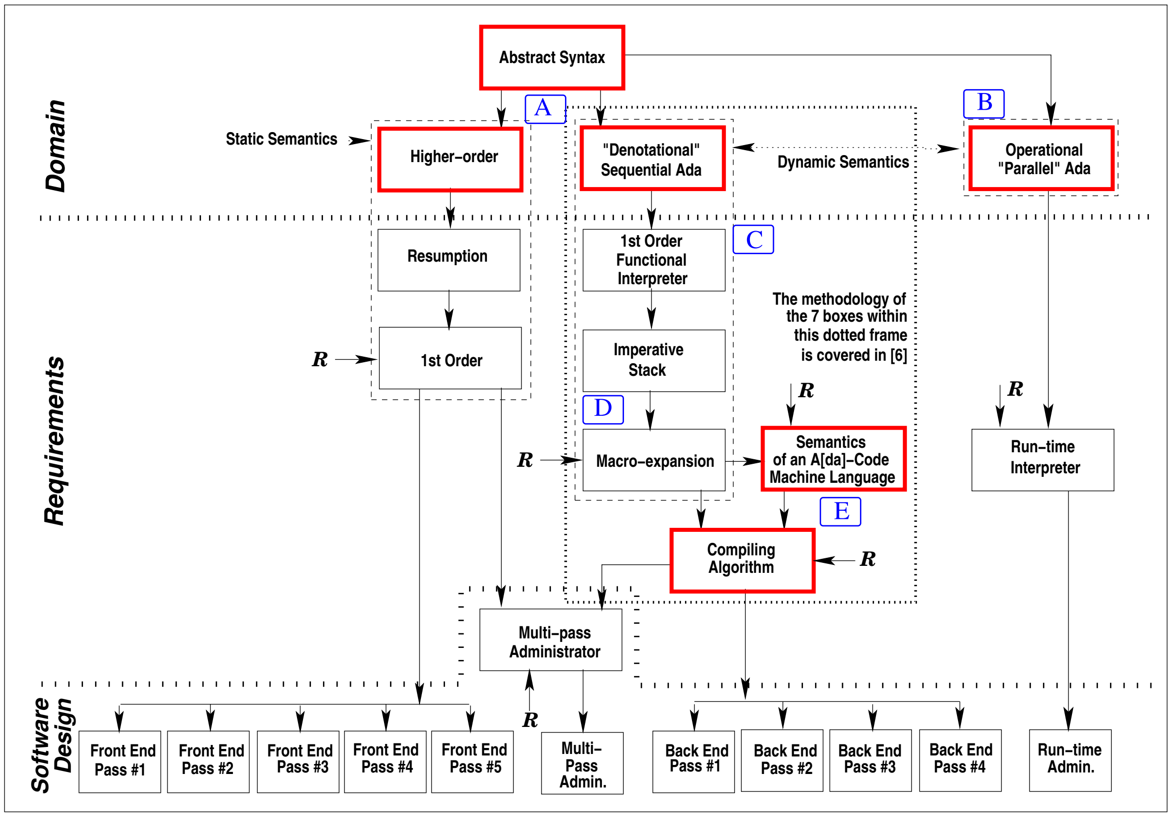


Figure 1: A Compiler Software Development Graph for Programming Languages with Concurrency

- [A] Denotational Semantics, *McCarthy, Scott and Strachey*[70, 71, 79].
- [B] CSP, *Hoare*[63].
- [C] First Order Semantics, *Landin* and *Reynolds* [66, 75].
- [D] Imperative Stack and Macro-expansion Semantics, *Bekič* [2].
- [E] X Code to Compiling Algorithm, *McCarthy & Painter* and Hans Bekič [72, 2].

The DDC CHILL and Ada compiler developments were, to my knowledge, the first time reasonably formal methods were used in, in these cases, the development for "large-scale" programming languages. The detected compiler errors, recorded in the first 10 years after their release, cost less that 1%, per year, of the original development costs, to repair. A "rate" many factors below [even annual] usual industry costs. Yet, no formal verification took place during these developments ! Versions of the DDC Ada compiler are still maintained by DDCI Inc., Phoenix, Arizona, the US.

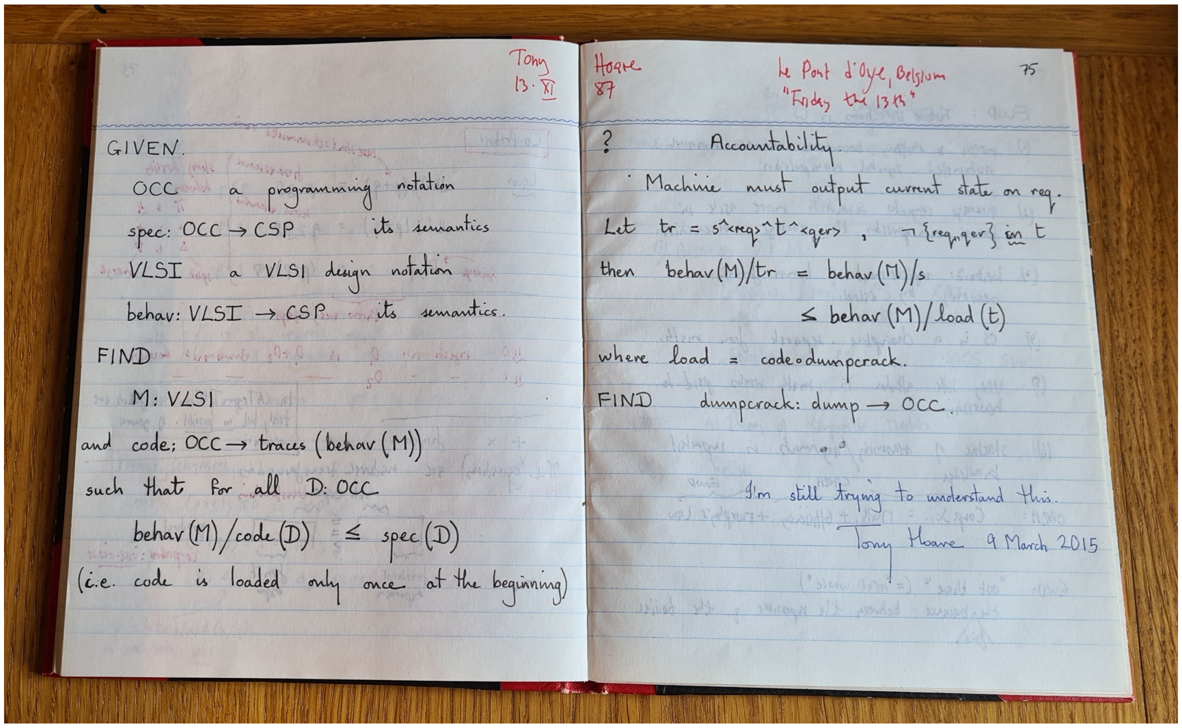
DDC had to close down its activity mid-summer 1989. The DDC Board, i.e., the financial backers, appears[7] to have spent too much money on "new, fashionably" SE "fads": HCI (human computer interaction) and "societal concern" efforts, etc. These financial backers were not prepared, otherwise, for formal methods. The DDC RAISE-related projects continued with the *CR Computer* firm.

[7]I am here expressing my own, personal opinion – in contrast to the Conclusion of [50].

6.2 DTU

Encouraged by the *Lyngby Winter School* event and the publications of [21, 52] Cliff Jones and I then published [54, *Formal Specification and Software Development*].

The EU-cosponsored `ProCoS` project had its root in the late 1980s. The `ProCoS` acronym stands for *Pro*vably *Co*rrect *S*ystems. The idea of having that project, in the context of *EU*'s `ESRIT Programme` was Tony Hoare's. At an IFIP WG 2.3 meeting at Chateau Du Pont D'Oye, Habay-la-Neuve, in Belgium, after a presentation of mine on a category theory-like structure of our 1980–1984 Ada compiler development, Tony, in the following break, asked for my note book. And this, except the right page, 9 March addendum, is what he wrote – see Fig. 2. He then went on to encourage me to put forward a proposal for a project that could investigate his notes. That became the `ProCoS` project. I think that it was Jonathan Bowen who coined the acronym. The most important outcome of the `ProCoS` project was the "discovery" of the `Duration Calculus` and its initial research [92, 91]. The 9 March Note by Tony Hoare was added at a 25'th anniversary gathering in London[8].



Tony
13 · XI

GIVEN.

OCC a programming notation

spec: OCC → CSP its semantics

VLSI a VLSI design notation

behav: VLSI → CSP its semantics.

FIND

M: VLSI

and code; OCC → traces (behav (M))

such that for all D: OCC

behav (M) / code (D) ≤ spec (D)

(i.e. code is loaded only once at the beginning)

Hoare
87

Le Pont d'Oye, Belgium
"Friday the 13th"

75

? Accountability.

· Machine must output current state on req.

Let tr = s^<req>^t^<qer> , ¬ {req, qer} in t

then behav (M) / tr = behav (M) / s

≤ behav (M) / load (t)

where load = code ∘ dumpcrack.

FIND dumpcrack: dump → OCC.

I'm still trying to understand this.

Tony Hoare 9 March 2015

Figure 2: Tony Hoare's Notes in my notebook: 13 Nov. 1987 & 9 March 2015

• • •

I was involved in three IFIP Working Group conferences in the 1980s. (i) TC 2's [22, *Formal Description of Programming Concepts (II), 1982*,, see photos at end of paper] in Garmisch-Partenkirschen, Germany, (ii) TC 2's *Formal Description of Programming Concepts*, Gl. Avernæs, 1984, and (iii) TC 2's [49, *Partial Evaluation and Mixed Computation*, 1987]. The latter focused on the partial evaluation concepts of Yushihiko Futamura, Andrei Petrovich Ershov, Anders Haraldson and Neil D. Jones.

[8] `http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=43198` and `https://www.bcs.org/-media/3086/facs-mar15.pdf` and `https://www.springerprofessional.de/en/provably-correct-systems/-12103398`

7 UNU-IIST: UN's Intl. Inst.for Software Technology, 1992–1997

I was member, 1984–1991, of *UNESCO*'s *Interim Intergovernmental Informatics Programme, UNESCO/IIIP* – a group of some 80 *UN* countries which met every other year, in November/December, at the *UNESCO* HQ in Paris. I represented the governments of the Nordic Countries: Denmark, Finland, Iceland, Norway and Sweden. On behalf of these I had to make so-called interventions concerning *UNESCO*'s multi-lateral support of informatics projects in developing countries. I was also, in 1986, the general chair of the IFIP World Computer Congress, in Sept., in Dublin Ireland. Professor *Ines Wesley-Tanaskovic*, former chairwoman of the *UN University Council*, had been charged by *UNU*, to "head-hunt" potential candidates for a planned *UNU [International] Institute for Software Technology, UNU-IIST*. She had heard my interventions in *UNESCO/IIIP* and my opening speech at the *IFIP World Computer Congress*. So, as a result of her recommendation, I was head-hunted as Director of *UNU–IIST* which began its work in Macau, July 1st, 1992.

UNU-IIST's task was to help emerging (i.e., developing) countries attain highest skills in forming their own software capabilities. *UNU-IIST*, that is I, decided to fulfill this task along two closely intertwined strands: Along one strand we experimentally developed, using formal methods, with *Fellows* from developing countries, software for real applications such a `Train Running Map` monitoring and control for the Chinese Railways, `Financial Management` for the Vietnam Ministry of Finance, `Distributed Telephone Services` for the Philippines Ministry of Telecommunications, `Multiscript` software for Mongolian, etc., etc. Along another strand to research, also with *Fellows* from developing countries, the method principles, procedures, techniques and tools. A focus was put on the `Duration Calculus` [91]. One *UNU-IIST* professional staff researcher worked with 1-3 groups of a few Fellows – daily. The two strands intertwined. There were weekly seminars for all so that the members of the two strands in this way got involved in each others' work.

The publication rate of the six or so research staff and the up to 24 Fellows, at notable conferences and in likewise journals, appears to have exceeded that of most larger computer science departments.

8 Last Years at The Technical University of Denmark, 1997–2007

Sadly, upon my return to DTU after 5 years in The Far East, I could not further develop my previous courses around formal software development. A colleague had taken over the introduction course and the department chairman did not allow me to give the sequel courses ! Instead, with colleagues, we gave courses in Software Engineering, "American-style", using a block-buster book ! Instead I wrote and published, in 2005–2006, the three volume [26, 27, 28, *Software Engineering*] – collected from years of lecture notes !

In 2004/2005 and again in 2006 I spent 1+1 year sabbaticals at *NUS: National University of Singapore* and *JAIST: Japan Advanced Institute of Science & Technology*, graciously hosted by respectively Profs. JinSong Dong and Kokichi Futatsugi. During the former guest professorship I finished [26] and during the latter I wrote [34, *Domain Engineering: Technology Management, Research and Engineering*] – a collection of 10 reports.

9 IFIP Working Groups 2.2 and 2.3: 1980–2006

Due to the efforts by Cliff B. Jones, IFIP WG 2.3 [Programming Methodology], and Erich Neuhold, IFIP WG 2.2 [Formal Description of Programming Concepts], I became a member of these two working groups around 1980.

One aspect of their meetings was that they were held more-or-less world-wide !

Another aspect of the two working groups was that WG 2.2 was computer science-oriented, while WG 2.3 was [definitely] computing science-oriented. WG 2.2 workshop presentations were, more-or-less, in the "classical" style of 'those of 'theoretical" computer science conferences, while WG 2.3 workshop presentations were radically different.

On opening morning potential WG 2.3 presenters each got 5 minutes to outline their *offer*. Before lunch members would then vote their preferences. The WG 2.3 chairman would then prepare, over lunch, a schedule for the week. Presentations were not limited in presentation time. And presentations were interactive. The members would frequently interrupt the presenter. Critique could often be rather explicit. Edsger Wybe Dijkstra was a member !

• • •

So 45 years of work was over. In most of these I had worked in science. At IBM, at DTU and at UNU-IIST. Lectured, tutored students, done research, written papers and books.

It was all "run-of-the-mill" ! Traditional. Nothing really "earth-shaking". Well, yes, the VDM work, its conception, use and propagation – through RAISE/RSL – was, perhaps, a little "out-of-the-ordinary" !

But, as it turned out, now came my best years as a scientist. It would be too immodest to not claim so. I really, sorry, do think that my work on *Domain Science & Engineering* – to be covered next – is a solid scientific contribution. Perhaps not yet so appreciated. We shall see !

10 Retirement, 2007–...

I finally got time to think ! After 30 years of "projects" [many in an EU, i.e., European Union, context], there was time to reflect. What had I really been striving at – especially as from my time at UNU-IIST ?

At UNU-IIST, when formulating software designs, we first specified, conventionally, and in `RSL`, the domain-at-hand. Now, a decade later, I started to question that specification approach.

I had tried, in the 1980s-2000s to make head-or-tail of work in requirements engineering. But somehow I was not enlightened. Something was missing.

Then I came upon:

The Triptych Dogma:

Before software can be designed we must understand "its" requirements.
Before requirements can be prescribed we must understand "the" domain.

So we must study, analyze and describe the domain.

10.1 The Search

So a long search began.

In [33, *Domain Engineering*] — the paper title should, in retrospect have been: *Domain Facets* — I focus on a "side issue" of domains: their, as I call them, *Domain Facets*. An edited extension of that 2008 paper appears as Chapter 8 in [43, *Domain Science & Engineering – A Foundation for Software Development*].

In [29, *From Domains to Requirements*], based on a conventional way of describing domains, I unfolded a method, its principle, procedures, techniques and tools, for rigorously "deriving" requirements prescriptions from domain descriptions. The method of [29, *2008*] has been further "refined" in [44, *Chapter 9, 2021*].

With [38, 39, *Domain Science & Engineering – From Computer Science to The Sciences of Informatics: The Engineering Part* and *...: The Science Part*] I made a first attempt at a calculus for domain description.

In [40, *Domains: Their Simulation, Monitoring and Control – A Divertimento of Ideas and Suggestions*] I examined the role of domain models in the understanding of computer simulation, monitoring and monitoring & control of domains. Later work on so-called *Digital Twins* appears to be foreseen here.

Finally, with [41, 42, *Manifest Domains: Analysis & Description* and *Domain Analysis & Description – Principles, Techniques and Modeling Languages*, 2016–2019] I seem to have "gotten it right" !

What made me get it right, as I claim, is that I had come across the works of the Danish philosopher Kai Sørlander [84, 85, 86, 87]. The main thrust of these books is *"What must necessarily be true, and hence be present in any description of any universe".*

[42] led to the book [44, *Domain Science & Engineering. A Foundation for Software Development*, 2021].

Yet, even [44] got it too complicated. So, like in *Anatole France*'s story of the Persian Emperor's desire to know the history of the Persian People[9], I iterated, in the years since the publication of [44, Nov.2021], over the calculus and its presentation. At present [47, *Domain Modeling*, 2024][10] with the shorter paper [48, *Domain Modeling*, June 2024] is a most recent iteration. I, rather immodestly, consider [48] my most important paper !

10.2 My *Domain Modeling* Approach

By a domain we shall understand *a rationally describable*[11] *segment of a discrete dynamics fragment of a human assisted reality: the world that we daily observe – in which we work and act, a reality made significant by human-created entities.*

How do we tackle the study, analysis and description of domains ?

Figure 3 on the next page suggests an approach. You, the domain analyzer cum describer, confront the chosen domain. You are, so-to-speak, positioned at the root of the up-side-down tree of Fig. 3 on the facing page. The root node, given its to branched, represents the question: *is the domain rationally describable ?* If so, then it is an *entity*. Now, is that entity an endurant[12] or a perdurant [13] And so forth: is it a *solid* or a *fluid* ?; if solid, is it a *part* or a *living species* ?; if part, is it *atomic* or *compound* ?; if compound, is it *Cartesian* or a *Part Set* ?. If it is either 'Cartesian' or 'Part Set', then we can describe its composition from other endurants – and, eventually, its internal qualities in terms of *unique identification, mereology, attributes*, and, when relevant, its contribution to so-called *intentional pull*s. And if it is 'Atomic' then we can, likewise, describe its internal qualities, etc.

[9] `http://profzeki.blogspot.com/2012/05/anatole-france-and-reductionism.html` and `file:///home/db/admin,+Ariel%20Vol.%201%20no.%201_84-99.pdf` Page 97

[10] [47] is currently being translated into Chinese by *Yang ShaoFa of the Institute of Software, Chinese Academy of Science, Beijing, China* and by *Mikhail Chupilko* and his colleagues of the *Institute of Systems Programming, Russian Academy of Science, Moscow, Russia.* A Spanish translation may appear !

[11] Or: describable by means of the tools provided by the method !

[12] Endurants are those quantities of domains that we can observe (see and touch), in *space*, as "complete" entities at no matter which point in `time` – "material" entities that persists, endures – capable of enduring adversity, severity, or hardship `[Merriam Webster]`

[13] Perdurants are those quantities of domains for which only a fragment exists, in *space*, if we look at or touch them at any given snapshot in *time* `[Merriam Webster]` ?

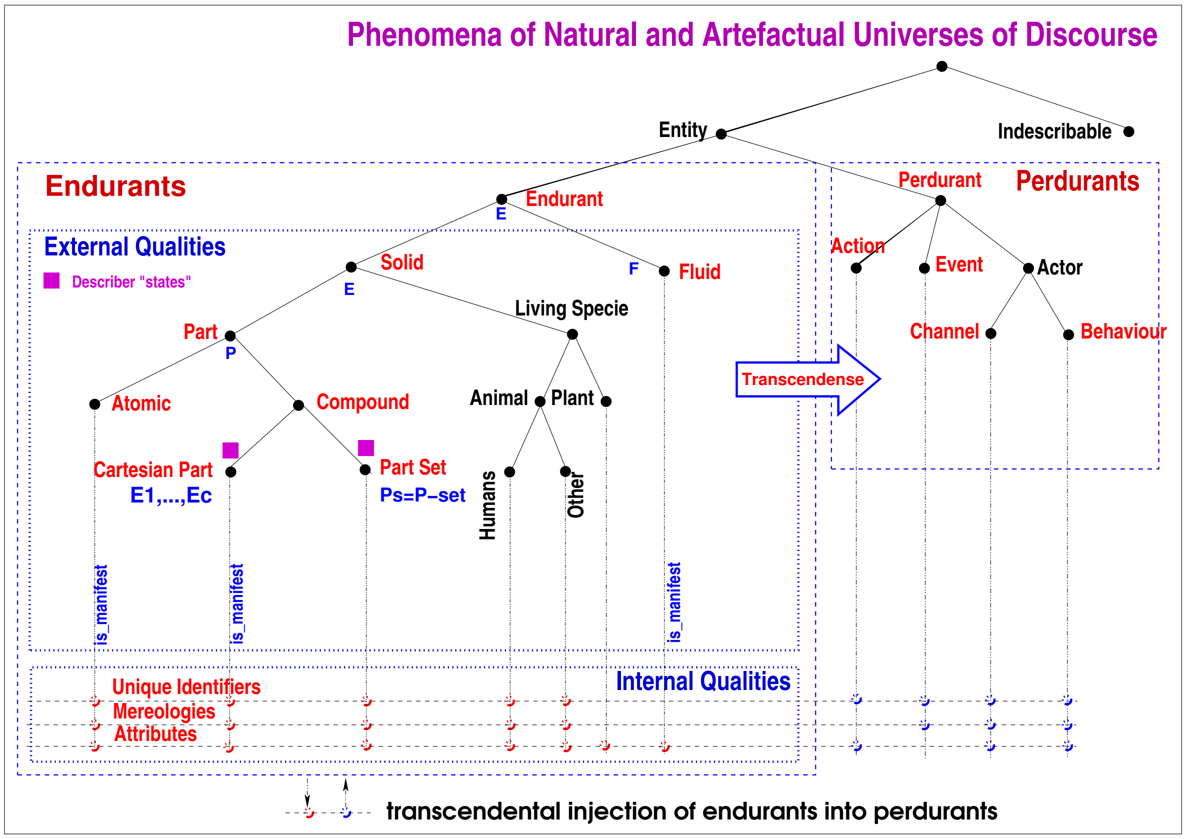


Figure 3: A Domain Analysis Ontology

10.3 My Immodest Own Assessment

With my work on domain science & engineering I immodestly think that I have made a significant, yet, yet to be recognized as such, contribution !

11 Conclusion

11.1 From Research to Science: Researchers & Scientists

A researcher, as I shall here understand it, is a person, who, with the existing tools of the field of research, investigates open problems of that field. A scientist, as I shall here understand it, is a person, usually with a background in research, who opens up for surprising new areas of science.

This paper is a scientific paper, not a research paper, I claim !

A PhD-study prepares the student for research.

(In Denmark a Dr.techn. degree testifies that the holder is a scientist. I got my PhD in 1969 and my Dr.techn. degree in 2002.)

At IBM and at DTU I was a researcher. For forty years. With my work on Domain Science & Engineering I have become a scientist.

11.2 Acknowledgments

In this review of my life in science I wish to acknowledge Jean-Raymond Abrial, Michéal Mac an Airchinnigh †, Gene M. Amdahl †, Stefan Andrei, Krzysztof Apt, Giorgio Ausiello, Arutyun I. Avetisyan, Ivan Bach †, John Warner Backus †, Hans Bekič †, Andrzej Blikle, Hans Bruun, Krzysztof Brzezinski, Egon Börger, Tien Chi Chen †, Mikhail Chupilko, Edward ("Ted") F. Codd †, Olivier Danvy, Edsger W. Dijkstra †, Jin-Song Dong, Balint Dömölki, Preben Folkjær, Kokichi Futatsugi, Chris W. George, Efim Grinkrug, Christian Gram †, Stefan Gruner, Jozef Gruska, Klaus Havelund, Magne Haveraaen, Otthein Herzog, Sir Tony Hoare, Henning Isaksson †, Viktor P. Ivannikov †, Ivar Jacobsen, Michael A. Jackson, Jean-Paul Jacob †, Tomasz W. Janowski, Per Gert Jensen †, Peter Johansen, Cliff B. Jones, Neil D. Jones †, Cai Kinberg †, Kouichi Kishida, Laura Kovacs, Leslie B. Lamport, Hans Langmaack, Peter Lucas †, Hans Henrik Løvengreen, Jørgen Steensgaard Madsen, Zohar Manna †, Christian Krog Madsen, Dominique Méry, Stefan Merz, Peter G. Neumann, Ronan Nugent, Asger Kierbye Nielsen †, Ole N. Oest, Martin Pečnička, Alexander K. Petrenko, Igor V. Pottosin †, Søren Prehn †, Douglas Taylor Ross †, Carlo Sanacroce, †, Marko Schütz–Schmuck, Natarajan Shankar. Kai Sørlander, Tetsuo Tamai, Andrzej Tarlecki, Enn Tõugu †, Kurt Walk, Gunnar Wedell †, Gerald M. Weinberg †, Jiří Wiedermann, Niklaus Wirth †, Aad van Wijngaarden †, Yang ShaoFa, Lotfi Zadeh †, Jiří Zlatuska, NaiJun Zhan, and Tom Østerby †: for their inspiration, trust in and support of me.

12 Bibliography

12.1 Bibliographical Notes

I have taken the liberty of "generously" citing my own work !

12.2 References

[1] H. Bekič, D. Bjørner, W. Henhapl, C.B. Jones & P. Lucas (1974): *A Formal Definition of a PL/I Subset.* Technical Report 25.139, IBM Laboratory, Vienna.

[2] Hans Bekič (1984): *Mathematical semantics and compiler correctness.* In Cliff B. Jones, editor: *Programming Languages and Their Definition: H. Bekič (1936–1982)*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 156–167, doi:10.1007/BFb0048943. Available at `https://doi.org/10.1007/BFb0048943`.

[3] Hans Bekič, Peter Lucas, Kurt Walk & Many Others (1966): *Formal Definition of PL/I, ULD Version I.* Technical Report, IBM Laboratory, Vienna.

[4] Hans Bekič, Peter Lucas, Kurt Walk & Many Others (1968): *Formal Definition of PL/I, ULD Version II.* Technical Report, IBM Laboratory, Vienna.

[5] Hans Bekič, Peter Lucas, Kurt Walk & Many Others (1969): *Formal Definition of PL/I, ULD Version III.* IBM Laboratory, Vienna.

[6] D. Bjørner, E. F. Codd, K. Deckert & I. L. Traiger (1973): *The GAMMA-0 Relational Data Base Interface Specifications of Objects and Operations.* Techn. Report RJ-1200, IBM Research, San José, Calif.

[7] Dines Bjørner (1969): *The Theory of Finite State Syntax Directed Transductions.* Ph.D. thesis, Dept. of Comp. Sci., Techn. Univ. of Denmark.

[8] Dines Bjørner (1970): *A Flow Mode, Self-Steering, Cellular Multiplier-Summation Processor. BIT* 10(2), pp. 125–14.

[9] Dines Bjørner, editor (1970): *Data Description & Access*. 1st ACM SICFIDET Workshop, Assoc.f.Comp.Mach., N.Y., USA.

[10] Dines Bjørner (1970): *Finite State Automaton Definition of Data Communication Line Control Procedures*. In: *FJCC (Fall Joint Comp. Conf.)*, 37, AFIPS, pp. 477–491.

[11] Dines Bjørner (1970): *Folded Syntax- and Recursive Flowchart-Machines*. In: *HICSS (Hawaii Int'l. Conf. Sys. Sci.)*, pp. 415–453.

[12] Dines Bjørner (1970): *Register Transfer and Transformation Machines*. In: *HICSS (Hawaii Int'l. Conf. Sys. Sci.)*, pp. 61–63.

[13] Dines Bjørner (1970): *The Synthesis of Finite State Syntax Directed Top-Down and Bottom-Up Transducers*. In: *SWAT (Symp. Switch. & Autom. Theory)*, IEEE, pp. 122–132.

[14] Dines Bjørner (1971): *On the Definition of Higher Level Language Machines*. In: *Computers and Automata*, *Microwave Research Inst. Symposia* 21, Polytechnic Inst. of Brooklyn, N.Y., USA, p. ...

[15] Dines Bjørner (1972): *Finite State Tree Computations (Part I)*. Research Rept. RJ-1053, IBM Research, San José, Calf.

[16] Dines Bjørner (1977): *Programming Languages: Formal Development of Interpreters and Compilers*. In: *International Computing Symposium 77 (eds. E. Morlet and D. Ribbens)*, European ACM, North-Holland Publ.Co., Amsterdam, pp. 1–21.

[17] Dines Bjørner (1977): *Programming Languages: Linguistics and Semantics*. In: *International Computing Symposium 77 (eds. E. Morlet and D. Ribbens)*, European ACM, North-Holland Publ.Co., Amsterdam, pp. 511–536.

[18] Dines Bjørner (1978): *Programming in the Meta-Language: A Tutorial*. In Dines Bjørner & Cliff B. Jones, editors: *The Vienna Development Method: The Meta-Language, [53]*, LNCS , Springer-Verlag, pp. 24–217.

[19] Dines Bjørner (1978): *Software Abstraction Principles: Tutorial Examples of an Operating System Command Language Specification and a PL/I-like On-Condition Language Definition*. In Dines Bjørner & Cliff B. Jones, editors: *The Vienna Development Method: The Meta-Language, [53]*, LNCS , Springer-Verlag, pp. 337–374.

[20] Dines Bjørner (1979): *The Vienna Development Method: Software Abstraction and Program Synthesis*. In: *Mathematical Studies of Information Processing*, *LNCS* 75, Springer-Verlag, p. ... Proceedings of Conference at Research Institute for Mathematical Sciences (RIMS), University of Kyoto, August 1978.

[21] Dines Bjørner, editor (1980): *Abstract Software Specifications*. *LNCS* 86, Springer.

[22] Dines Bjørner, editor (1982): *Formal Description of Programming Concepts (II)*. IFIP TC-2 Work.Conf., Garmisch-Partkirschen, North-Holland Publ.Co., Amsterdam.

[23] Dines Bjørner (1986): *Project Graphs and Meta-Programs: Towards a Theory of Software Development*. In N. Habermann & U. Montanari, editors: *Proc. Capri '86 Conf. on Innovative Software Factories and Ada, Lecture Notes on Computer Science*, Springer-Verlag, p. ...

[24] Dines Bjørner (1986): *Software Development Graphs — A Unifying Concept for Software Development?* In K.V. Nori, editor: *Vol. 241 of Lecture Notes in Computer Science: Foundations of Software Technology and Theoretical Computer Science*, Springer-Verlag, pp. 1–9.

[25] Dines Bjørner (1987): *The Stepwise Development of Software Development Graphs: Meta-Programming VDM Developments*. In: *See [55]*, *LNCS* 252, Springer-Verlag, Heidelberg, Germany, pp. 77–96.

[26] Dines Bjørner (2006): *Software Engineering, Vol. 1: Abstraction and Modelling*. Texts in Theoretical Computer Science, the EATCS Series, Springer. See [30, 35].

[27] Dines Bjørner (2006): *Software Engineering, Vol. 2: Specification of Systems and Languages*. Texts in Theoretical Computer Science, the EATCS Series, Springer. Chapters 12–14 are primarily authored by Christian Krog Madsen. See [31, 36].

[28] Dines Bjørner (2006): *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Texts in Theoretical Computer Science, the EATCS Series, Springer. See [32, 37].

[29] Dines Bjørner (2008): *From Domains to Requirements* *www.imm.dtu.dk/~dibj/2008/ugo/ugo65.pdf*. In: *Montanari Festschrift*, *Lecture Notes in Computer Science (eds. Pierpaolo Degano, Rocco De Nicola and José Meseguer)* 5065, Springer, Heidelberg, pp. 1–30.

[30] Dines Bjørner (2008): *Software Engineering, Vol. 1: Abstraction and Modelling*. Qinghua University Press.

[31] Dines Bjørner (2008): *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press.

[32] Dines Bjørner (2008): *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Qinghua University Press.

[33] Dines Bjørner (2009): *Domain Engineering*. In Paul Boca, Jonathan Bowen & Jawed Siddiqi, editors: *Formal Methods: State of the Art and New Directions*, Eds. Paul Boca and Jonathan Bowen, Springer, London, UK, pp. 1–42, doi:10.1007/978-1-84882-736-3_1.

[34] Dines Bjørner (2009): *Domain Engineering: Technology Management, Research and Engineering*. Research Monograph (#4); JAIST Press, 1-1, Asahidai, Nomi, Ishikawa 923-1292 Japan. This Research Monograph contains the following main chapters:

1. *On Domains and On Domain Engineering – Prerequisites for Trustworthy Software – A Necessity for Believable Management*, pages 3–38.
2. *Possible Collaborative Domain Projects – A Management Brief*, pages 39–56.
3. *The Rôle of Domain Engineering in Software Development*, pages 57–72.
4. *Verified Software for Ubiquitous Computing – A VSTTE Ubiquitous Computing Project Proposal*, pages 73–106.
5. *The Triptych Process Model – Process Assessment and Improvement*, pages 107–138.
6. *Domains and Problem Frames – The Triptych Dogma and M.A.Jackson's PF Paradigm*, pages 139–175.
7. *Documents – A Rough Sketch Domain Analysis*, pages 179–200.
8. *Public Government – A Rough Sketch Domain Analysis*, pages 201–222.
9. *Towards a Model of IT Security — – The ISO Information Security Code of Practice – An Incomplete Rough Sketch Analysis*, pages 223–282.
10. *Towards a Family of Script Languages – – Licenses and Contracts – An Incomplete Sketch*, pages 283–328.

.

[35] Dines Bjørner (2010): **Chinese:** *Software Engineering, Vol. 1: Abstraction and Modelling*. Qinghua University Press. Translated by Dr Liu Bo Chao et al.

[36] Dines Bjørner (2010): **Chinese:** *Software Engineering, Vol. 2: Specification of Systems and Languages*. Qinghua University Press. Translated by Dr Liu Bo Chao et al.

[37] Dines Bjørner (2010): **Chinese:** *Software Engineering, Vol. 3: Domains, Requirements and Software Design*. Qinghua University Press. Translated by Dr Liu Bo Chao et al.

[38] Dines Bjørner (2010): *Domain Science & Engineering – From Computer Science to The Sciences of Informatics, Part I of II: The Engineering Part*. *Kibernetika i sistemny analiz* 2(4), pp. 100–116.

[39] Dines Bjørner (2011): *Domain Science & Engineering – From Computer Science to The Sciences of Informatics Part II of II: The Science Part*. *Kibernetika i sistemny analiz* 2(3), pp. 100–120.

[40] Dines Bjørner (2011): *Domains: Their Simulation, Monitoring and Control – A Divertimento of Ideas and Suggestions*. In: *Rainbow of Computer Science, Festschrift for Hermann Maurer on the Occasion of His 70th Anniversary.*, Festschrift (eds. C. Calude, G. Rozenberg and A. Saloma), Springer, Heidelberg, Germany, pp. 167–183. `www.imm.dtu.dk/~dibj/maurer-bjorner.pdf`.

[41] Dines Bjørner (2017): *Manifest Domains: Analysis & Description. Formal Aspects of Computing www.imm.dtu.dk/~dibj/2015/faoc/faoc-bjorner.pdf* 29(2), pp. 175–225. First Online: 26 July 2016. DOI 10.1007/s00165-016-0385-z.

[42] Dines Bjørner (2019): *Domain Analysis & Description – Principles, Techniques and Modeling Languages. www.imm.dtu.dk/~dibj/2018/tosem/Bjorner-TOSEM.pdf. ACM Trans. on Software Engineering and Methodology* 28(2), p. 66 pages.

[43] Dines Bjørner (2021): *Domain Science & Engineering – A Foundation for Software Development*. Springer.

[44] Dines Bjørner (2021): *Domain Science & Engineering – A Foundation for Software Development*. EATCS Monographs in Theoretical Computer Science, Springer, Heidelberg, Germany. A revised version of this book is [46].

[45] Dines Bjørner (2023): *Domain Modelling – A Primer*. A short version of [46]. xii+202 pages[14].

[46] Dines Bjørner (2023): *Domain Science & Engineering – A Foundation for Software Development*. Revised edition of [44]. xii+346 pages[15].

[47] Dines Bjørner (2024): *Domain Modelling*. Technical University of Denmark. Revised edition of [44]. xii+208 pages. `http://www.imm.dtu.dk/~dibj/2024/primer/primer.pdf`.

[48] Dines Bjørner (2024): *Domain Modelling*. *To be submitted*, p. 26. Institute of Mathematics and Computer Science. Technical University of Denmark.

[49] Dines Bjørner, Andrei Petrovich Ershov & Neil Deaton Jones, editors (1988): *Partial Evaluation and Mixed Computation. Proceedings of the IFIP TC2 Workshop, Gammel Avernæs, Denmark, October 1987*. North-Holland. 625 pages.

[50] Dines Bjørner, Chr. Gram, Ole N. Oest & Leif Rystrøm (2010): *Dansk Datamatik Center*. In Benkt Wangler & Per Lundin, editors: *History of Nordic Computing*, Springer, Stockholm, Sweden, p. ...

[51] Dines Bjørner & Martin C. Henson, editors (2008): *Logics of Specification Languages*. EATCS Series, Monograph in Theoretical Computer Science, Springer, Heidelberg, Germany.

[52] Dines Bjørner & Cliff B. Jones, editors (1978): *The Vienna Development Method: The Meta-Language*. *LNCS* 61, Springer, Heidelberg, Germany.

[53] Dines Bjørner & Cliff B. Jones, editors (1978): *The Vienna Development Method: The Meta-Language*. *LNCS* 61, Springer. This was the first monograph on *Meta-IV*.

[54] Dines Bjørner & Cliff B. Jones, editors (1982): *Formal Specification and Software Development*. Prentice-Hall, London, England.

[55] Dines Bjørner, Cliff B. Jones, Micheal Mac an Airchinnigh & Erich J. Neuhold, editors (1987): *VDM – A Formal Method at Work*. Proc. VDM-Europe Symposium 1987, Brussels, Belgium, Springer, Lecture Notes in Computer Science, Vol. 252.

[56] Dines Bjørner & Ole N. Oest, editors (1980): *Towards a Formal Description of Ada*. *LNCS* 98, Springer, Heidelberg, Germany.

[57] Geert Bagge Clemmensen & Ole N. Oest (1984): *Formal Specification and Development of an Ada Compiler – A VDM Case Study*. In: *Proc. 7th International Conf. on Software Engineering, 26.-29. March 1984, Orlando, Florida*, IEEE, New York, USA, pp. 430–440.

[58] Denmark, DDC and Italy, Pisa+Genoa, editor (1984-1987): *Selected papers and reports on the CEC MAP Ada Formal Definition Project*. Dansk Datamatik Center.

AdaFD: "The Formal Definition of Ada Dynamic Semantics", Deliverables 15 and 16 of the CEC MAP project: The Draft Formal Definition of ANSI/MIL-STD 1815A Ada.

[14]This book is currently being translated into Chinese by Dr. Yang ShaoFa, IoS/CAS, Beijing and into Russian by Dr. Mikhail Chupilko, ISP/RAS, Moscow

[15]Due to copyright reasons no URL is given to this document's possible Internet location. A primer version, omitting certain chapters, is [45]

Astesiano et al.: "Final Specification Language", Deliverable 9 of the CEC MAP project: The Draft Formal Definition of ANSI/MIL-STD 1815A Ada.

Fantechi et al.: "Feasibility of ACVC validation with respect to the Ada Formal Definition", Deliverable 30 of the CEC MAP project: The Draft Formal Definition of ANSI/MIL-STD 1815A Ada.

Gallo et al.: "Ada FD Tool set: Architecture and Preliminary Design", Deliverable 20 of the CEC MAP project: The Draft Formal Definition of ANSI/MIL-STD 1815A Ada.

Karlsen et al.: "The Draft Formal Definition of Ada, The Dynamic Semantics Definition", Dansk Datamatik Center/CRAI/IEI/University of Genoa, January 1987.

Storbank et al.: "The Draft Formal Definition of Ada, The Static Semantics Definition", Dansk Datamatik Center, January 1987.

Reggio et al.: "The Draft Formal Definition of Ada, The User Manual of the Meta-Language", CRAI/IEI/University of Genoa, September 1986.

This is a temporary reference. An appropriate reference will be "constructed" before end of February 2010..

[59] E. Engeler (1971): *Symposium on Semantics of Algorithmic Languages*. *Lecture Notes in Mathematics* 188, Springer.

[60] Chris W. George, Peter Haff, Klaus Havelund, Anne Elisabeth Haxthausen, Robert Milne, Claus Bendix Nielsen, Søren Prehn & Kim Ritter Wagner (1992): *The RAISE Specification Language*. The BCS Practitioner Series, Prentice-Hall, Hemel Hampstead, England.

[61] Chris W. George, Anne Elisabeth Haxthausen, Steven Hughes, Robert Milne, Søren Prehn & Jan Storbank Pedersen (1995): *The RAISE Development Method*. The BCS Practitioner Series, Prentice-Hall, Hemel Hampstead, England.

[62] P.L. Haff, editor (1981): *The Formal Definition of CHILL*. ITU (Intl. Telecmm. Union), Geneva, Switzerland.

[63] Charles Anthony Richard Hoare (1985): *Communicating Sequential Processes*. C.A.R. Hoare Series in Computer Science, Prentice-Hall International, London, England. Published electronically: `using-csp.com/cspbook.pdf` (2004).

[64] IEEE CS (1990): *IEEE Standard Glossay of Software Engineering Terminology*. IEEE Std.610.12.

[65] Michael A. Jackson (1995): *Software Requirements & Specifications: a lexicon of practice, principles and prejudices*. ACM Press, Addison-Wesley, Reading, England.

[66] Peter J Landin (1964): *The mechanical evaluation of expressions*. *The Computer Journal* 6(4), pp. 308–320.

[67] Peter J. Landin (1964): *The Mechanical Evaluation of Expressions*. *Computer Journal* 6(4), pp. 308–320.

[68] Peter J. Landin (1966): *A Formal Description of ALGOL 60*. In: *[88]*, pp. 266–294.

[69] Peter J. Landin (1966): *A Lambda Calculus Approach*. In L. Fox, editor: *Advances in Programming and Non-Numeric Computations*, Pergamon Press, pp. 97–141.

[70] John McCarthy (1960): *Recursive Functions of Symbolic Expressions and Their Computation by Machines, Part I*. *Communications of the ACM* 3(4), pp. 184–195.

[71] John McCarthy (1962): *Towards a Mathematical Science of Computation*. In C.M. Popplewell, editor: *IFIP World Congress Proceedings*, pp. 21–28.

[72] John McCarthy & James Painter (1966): *Correctness of a Compiler for Arithmetic Expressions*. In: *[76]*, pp. 33–41. Dept. of Computer Science, Stanford University, California, USA.

[73] Ole N. Oest (1986): *VDM From Research to Practice (Invited Paper)*. In: *IFIP Congress*, pp. 527–534.

[74] P. G. Larsen and B. S. Hansen and H. Brunn N. Plat and H. Toetenel and D. J. Andrews and J. Dawes and G. Parkin and others (1996): *Information technology — Programming languages, their environments and system software interfaces — Vienna Development Method — Specification Language — Part 1: Base language*.

[75] John C Reynolds (1972): *Definitional interpreters for higher-order programming languages*. In: *Proceedings of the ACM annual conference-Volume 2*, ACM, pp. 717–740.

[76] Jack T. Schwartz (1967): *Mathematical Aspects of Computer Science, Proc. of Symp. in Appl. Math.* American Mathematical Society, Rhode Island, USA.

[77] Dana S. Scott (1970): *The Lattice of Flow Diagrams*. In: *[59]*, pp. 311–366.

[78] Dana S. Scott (1970): *Outline of a Mathematical Theory of Computation*. In: *Proc. 4th Ann. Princeton Conf. on Inf. Sci. and Sys.*, p. 169.

[79] Dana S. Scott & Christopher Strachey (1971): *Towards a Mathematical Semantics for Computer Languages*. In: *Computers and Automata*, *Microwave Research Inst. Symposia* 21, pp. 19–46.

[80] D.S. Scott (1972): *Continuous Lattices*. In F.W. Lawvere, editor: *Toposes, Algebraic Geometry and Logic*, Springer, Lecture Notes in Mathematics, Vol. 274, pp. 97–136.

[81] D.S. Scott (1972): *Data Types as Lattices*. Unpublished Lecture Notes, Amsterdam.

[82] D.S. Scott (1972): *Lattice Theory, Data Types and Semantics*. In R. Rustin, editor: *Symposium on Formal Semantics*, Prentice-Hall, pp. 67–106.

[83] D.S. Scott (1972): *Mathematical Concepts in Programming Language Semantics*. In: *Proc. AFIPS, Spring Joint Computer Conference, 40*, pp. 225–234.

[84] Kai Sørlander (1994): *Det Uomgængelige – Filosofiske Deduktioner [The Inevitable – Philosophical Deductions, with a foreword by Georg Henrik von Wright]*. Munksgaard · Rosinante, Copenhagen, Denmark. 168 pages.

[85] Kai Sørlander (2016): *Indføring i Filosofien [Introduction to The Philosophy]*. Informations Forlag, Copenhagen, Denmark. 233 pages.

[86] Kai Sørlander (2022): *Den rene fornufts struktur [The Structure of Pure Reason]*. Ellekær, Slagelse, Denmark. See [87].

[87] Kai Sørlander (2023): *The Structure of Pure Reason*. Publisher to be decided. This is an English translation of [86] – done by Dines Bjørner in collaboration with the author.

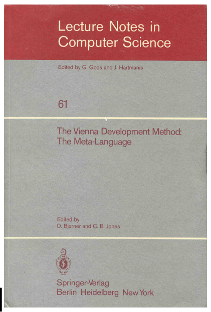
[88] T. B. Steel, editor (1966): *Formal Language Description Languages for Computer Programming, IFIP TC-2 Working Conference, 2964, Baden*. North-Holland Publ.Co., Amsterdam.

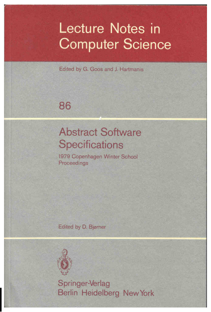
[89] C. Strachey (1966): *Towards a Formal Semantics*. In: *[88]*, pp. 198–220.

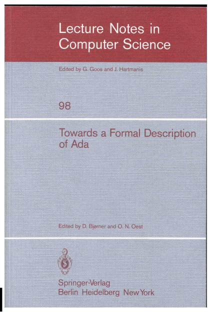
[90] C. Strachey (1968): *Fundamental Concepts in Programming Languages*. Unpubl. Lecture Notes, NATO Summer School, Copenhagen, 1967, and Programming Research Group, Oxford Univ.

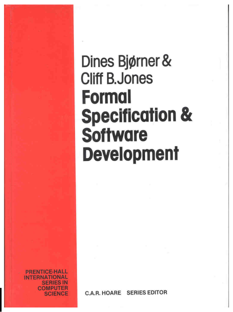
[91] Chao Chen Zhou & Michael R. Hansen (2004): *Duration Calculus: A Formal Approach to Real–time Systems*. Monographs in Theoretical Computer Science. An EATCS Series, Springer–Verlag.

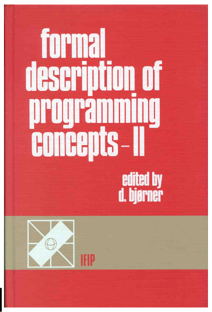
[92] ChaoChen Zhou, C.A.R. Hoare & Anders P. Ravn (1991): *A calculus of durations*. *Information Processing Letters* 40(5), pp. 269–276, doi:https://doi.org/10.1016/0020-0190(91)90122-X. Available at `https://www.sciencedirect.com/science/article/pii/002001909190122X`.

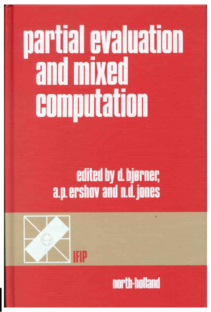
[52] 
Lecture Notes in Computer Science
Edited by G. Goos and J. Hartmanis
61
The Vienna Development Method: The Meta-Language
Edited by D. Bjørner and C. B. Jones
Springer-Verlag Berlin Heidelberg New York

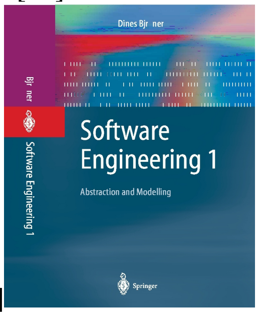
[21] 
Lecture Notes in Computer Science
Edited by G. Goos and J. Hartmanis
86
Abstract Software Specifications
1979 Copenhagen Winter School Proceedings
Edited by D. Bjørner
Springer-Verlag Berlin Heidelberg New York

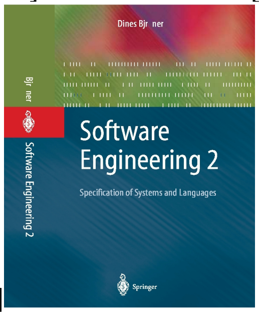
[56] 
Lecture Notes in Computer Science
Edited by G. Goos and J. Hartmanis
98
Towards a Formal Description of Ada
Edited by D. Bjørner and O. N. Oest
Springer-Verlag Berlin Heidelberg New York

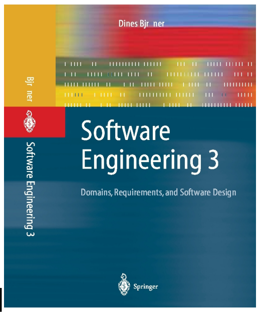
[54]

Dines Bjørner & Cliff B.Jones
Formal Specification & Software Development
PRENTICE-HALL INTERNATIONAL SERIES IN COMPUTER SCIENCE
C.A.R. HOARE SERIES EDITOR

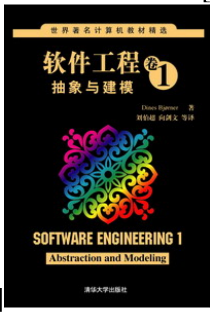
[22]

formal description of programming concepts – II
edited by d. bjørner
IFIP

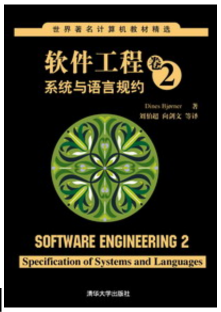
[49]

partial evaluation and mixed computation
edited by d. bjørner, a.p. ershov and n.d. jones
IFIP
north-holland

[26]

Dines Bjørner
Software Engineering 1
Abstraction and Modelling
Springer

[27]

Dines Bjørner
Software Engineering 2
Specification of Systems and Languages
Springer

[28]

Dines Bjørner
Software Engineering 3
Domains, Requirements, and Software Design
Springer

[35]

世界著名计算机教材精选
软件工程 卷1
抽象与建模
SOFTWARE ENGINEERING 1
Abstraction and Modeling
清华大学出版社

[36]

世界著名计算机教材精选
软件工程 卷2
系统与语言规约
SOFTWARE ENGINEERING 2
Specification of Systems and Languages
清华大学出版社

[37]

世界著名计算机教材精选
软件工程 卷3
领域、需求与软件设计
SOFTWARE ENGINEERING 3
Domains, Requirements, and Software Design
清华大学出版社

[30]

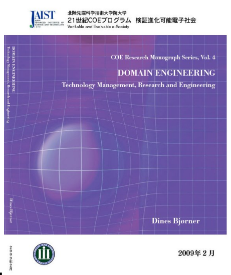
SOFTWARE ENGINEERING 1
ABSTRACTION AND MODELLING
软件工程 卷1
抽象与建模
Dines Bjørner 著
清华大学出版社

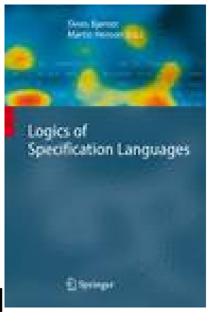
[31]

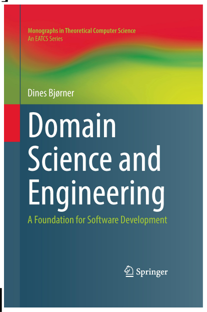
SOFTWARE ENGINEERING 2
SPECIFICATION OF SYSTEMS AND LANGUAGES
软件工程 卷2
系统与语言规约
清华大学出版社

[32]

SOFTWARE ENGINEERING 3
DOMAINS, REQUIREMENTS AND SOFTWARE DESIGN
软件工程 卷3
领域、需求与软件设计
清华大学出版社

[34]

JAIST
COE Research Monograph Series, Vol. 4
DOMAIN ENGINEERING
Technology Management, Research and Engineering
Dines Bjorner
2009年2月

[51]

Logics of Specification Languages
Springer

[44]

Monographs in Theoretical Computer Science
An EATCS Series
Dines Bjørner
Domain Science and Engineering
A Foundation for Software Development
Springer