CONTOURS OF INFORMATICS

Dines Bjørner

Prof., PhD, Dr.h.c., MAE, MRANS, ACM Fellow, IEEE Fellow, ...

Computer Science and Engineering
Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Kgs.Lyngby
Denmark

bjorner@gmail.com

March 30, 2007: 14:33

Presented at DTU: March 30, 2007

Agenda

- Aims & Objectives
- CVs
- Informatics – A Delineation — Syntax
- The Disciplines of Informatics — Semantics
  - ⋆ The Computer & Computing Sciences and Mathematics
  - ⋆ Linguistics/Semiotics and Philosophy
  - ⋆ Knowledge & Domain Engineering: A Domain Description Example
  - ⋆ IT Applications and Software Engineering
- On Methods and On Formal Techniques
- On Informatics Engineering:
  - ⋆ On Professionalism,
  - ⋆ CS, CS & SE
  - ⋆ On Obstacles to Professional SE
  - ⋆ A New Kind of Software Engineer
- Informatics — A New Universe
- Conclusion

Aims & Objectives

Aims

- To let you in on my way of thinking about my field of study —
- an approach that has obviously determined
  - ⋆ my teaching, my research and hence
  - ⋆ the structure and contents of my 3-volume book,
- and an approach and a view that pleads for a central rôle of the new field of domain science and engineering

Objectives

- To, perhaps, get you to think and act along a similar line —
- and maybe to put some more Informatics into DTU/IMM
- and some more interest in domain science and engineering.

CVs
Syntax

- MSc EE, Jan. 1962, PhD CS, Jan. 1969
- **IBM** Devt.: Stockholm, Hursley (UK) and San Jose (Calif.)
  - ⋆ Hardware Design, March 1962 — June 1969 Wonderful years
- **IBM** Research:
  - ⋆ Calif., July 1969 — April 1973 Wonderful years
  - ⋆ Wien: May 1973 — Aug. 1975 Wonderful and decisive years
- Vis.Prof.:
  - ⋆ **UC Berkeley** 1971–72
  - ⋆ **DIKU** Sept. 1975 — Aug. 1976
  - ⋆ **Kiel** Spring/Summer 1980
  - ⋆ **NUS**, Singapore 2004/2005
  - ⋆ **JAIST** (Japan) 2006
- Professor at **DTU**: Sept. 1976 — March 2007
- Co-founder (w/ **Chr. Gram**) and Sci. Advisor: **DDC**: Sept. 1979 — Dec. 1989
- Founding and First UN Director: **UNU-IIST**: Feb. 1992 — June 1997

CVs (Continued)
Semantics

- Computer Architectures and Machine Organisation 1962 – 1969
  IBM 1070, IBM 1800, ACS-1
- With (the late) John W. Backus: Red-1, Red-2 $\Rightarrow$ ffp 1969 – 1971
- With (the late) Ted Codd: DSL-α $\Rightarrow$ SQL 1971 – 1973
- Semantics of Programming Languages: IBM's PL/I, etc. $\Rightarrow$ VDM 1973 – 1975
- VDM: The Vienna Development Method 1973 – 1984
  formal, mathematics-based staged devt. of provably correct software
- Use of VDM for CHILL, Ada, etc. 1978 – 1984
- R&D of RAISE: Rigorous Approach to Industrial Softw. Eng. 1985 – ...
- Influence:
  - ⋆ *J. McCarthy, P. Landin, Dana Scott, (the late) C. Strachey, ... 1964/1973 – ...*
  - ⋆ *The IBM Vienna Group (Lucas, (the late) Bekič, Walk, Jones) 1973 – 1975 – ...*
  - ⋆ *IFIP WG2.3: (the late) E.W. Dijkstra, C.A.R. Hoare, M.A. Jackson, ... 1979 – ...*
  - ⋆ *(the late) Søren Prehn, Chris George, ... 1984 – ...*

Informatics — A Delineation (Syntax)

Software development requires many more proficiencies than "just programming":

- **Informatics**

  $=$ Computer Science $\oplus$ Computing Science *Datalogi*

  $\oplus$ Mathematics

  $\oplus$ Linguistics/Semiotics $\Rightarrow$ Philosophy

  $\oplus$ Software Engineering *Datamatik*

  $=$ **Domain Engineering**

  $\oplus$ Requirements Engineering

  $\oplus$ Software Design

  $\oplus$ IT Applications

Computer Science (I of II) — Rough Delineation

There are, pragmatically speaking two rather distinct sciences involved here:

- Comput**er** science
  - ⋆ is the study and knowledge about
  - ⋆ which "kind of things"
  - ⋆ can "exist inside" computers —
  - ⋆ including investigating the issues
    - ◇ "what are computers", "what does exist mean".

Computing Science (I of II) — Rough Delineation

- Comput**ing** science
  - ⋆ is the study and knowledge about
  - ⋆ how to construct "those things."

Computer Science (II of II) — Examples

- Examples of computer science topics:
  - ⋆ computability, computational models
  - ⋆ abstract complexity theory
  - ⋆ automata theory and formal languages
  - ⋆ *&c.*

Computing Science (II of II) — Foundational Examples

- Examples of computing science topics:
  - ⋆ Algorithmics, searching, sorting, ...
  - ⋆ Functional, logic, imperative and parallel programming
  - ⋆ Abstraction and modelling, refinement calculi
  - ⋆ Verification, model checking, (formal) testing *&c.*

Computing Science (II of II (Continued)) — Application Examples

- Semantics of programming languages
  and "their" interpreters and compilers
- Semantics of information (database) systems
  and their database management systems
- Semantics of process systems
  and their management (operating) systems
- Semantics of data movement
  and its management (data communication) systems
- Semantics of the Internet and the Web concept,
  Internet systems, Web programming, *&c.*
- *&c.*

Mathematics

Informatics is not based in the natural science.
Informatics has a main base in mathematics.

- **"Pure" Mathematics** **foundations and practice**
  - ⋆ Discrete Math.:
    - ◇ Logic **but not in the classical sense of math. logic**
    - ◇ Algebra **but not in the classical sense of algebra**
    - ◇ Graph Theory and Combinatorics, ...
  - ⋆ Meta-Mathematics (λ-Calculus, Recursive Function Theory)
  - ⋆ Calculus, Probability Theory, Statistics, ... *sometimes forgotten*
- **"Applied" Mathematics** **practice**
  - ⋆ Operations Research, ...

Linguistics/Semiotics

Informal narrative descriptions, prescriptions and specifications go hand-in-hand with formal such.

And: descriptions, prescriptions and specifications cover the below facets of semiotics:

- **Pragmatics**
  SEs tend to "hide" the pragmatics
- **Semantics** *Can be formalised*
- **Syntax** *Can be formalised*
  CS Depts. tend to focus mostly on syntax

Philosophy

- What can exist ?
- What can be described and what is a description ?

- **Mereology:** *theory of parts (and wholes)*
- **Espistemology:** *the study of knowledge and justified belief*
  - ⋆ epistemology is about issues
    - ◇ having to do with the creation and dissemination of knowledge
    - ◇ in particular areas of inquiry;
  - ⋆ translates into issues of scientific methodology:
    - ◇ how can one develop theories or models
    - ◇ that are better than competing theories ... ?
- **Ontology:** *specification of a conceptualization*

The Domain Engineering Dogma

- *Before* **software** *can be* **design***ed*
- *we must understand its* **requirements**.
- *Before* **requirements** *can be expressed*
- *we must understand the (application)* **domain**.

Software Engineering

- Ideally
  - ⋆ first: $\mathcal{D}$**omain Engineering**
  - ⋆ then: $\mathcal{R}$**equirements Engineering**
  - ⋆ finally: $\mathcal{S}$**oftware Design**
- such that $\mathbf{D}, \mathbf{S} \models \mathbf{R}$

Domain Science and Engineering

- **By a domain we understand a universe of discourse.**
- Examples of domains:
  - ⋆ *airports* (**Changi**)
  - ⋆ *air traffic* (**Pearl River Delta**)
  - ⋆ *container logistics* (**Maersk, SG**),
  - ⋆ *documents* (**Fuji Xerox, IBM**)
  - ⋆ *financial services*
  - ⋆ *health care (incl. EPJ)*
  - ⋆ *Internet (ubiquitous computing)*
  - ⋆ *IT security* (**IBM TRL**)
  - ⋆ *robotics (1991 paper,* **Kyushu** *[2006])*
  - ⋆ *transportation, as such, or*
    - ◇ *electronic road pricing* (**Singapore**)
    - ◇ *railways*
    - ◇ *road traffic*

  &c.
- **By domain engineering we understand the R&D of domain descriptions.**
- **By domain science we understand the study and knowledge about domain descriptions.**

Domain Engineering — Why Not ?

- All other, i.e., the classical, engineering disciplines builds on (the natural) sciences.
- No-one would hire a $\mathcal{Y}$ engineer unless that person was strong in science $\mathcal{X}$, where $\mathcal{X}$ provides the foundation for $\mathcal{Y}$.
- But as for domain $\mathcal{A}$ software engineers, no-one asks for competence in science $\mathcal{B}$, where science $\mathcal{B}$ could be the domain science of for example
  - ⋆ air traffic,
  - ⋆ container logistics,
  - ⋆ financial services,
  - ⋆ health care,
  - ⋆ transportation,
  - ⋆ or other.

Example of a Domain Description

- A multi-modal transport net consists of one or more segments and two or more junctions.
- With segments [junctions] we can associate the following attributes:
  - ⋆ segment [junction] identifiers,
  - ⋆ the identifiers of the two junctions to which segments are connected [the identifiers of the one or more segments connected to the junction],
  - ⋆ the mode (road, rail, air-lane, shipping lane) of a segment [the modes of the segments connected to the junction].

type

N, S, J, Si, Ji, M

value

obs_Ss: N $\longrightarrow$ S-**set**, obs_Js: N $\longrightarrow$ J-**set**

obs_Si: S $\longrightarrow$ Si, obs_Ji: J $\longrightarrow$ Ji

obs_Jis: S $\longrightarrow$ Ji-**set**, obs_Sis: J $\longrightarrow$ Si-**set**

obs_M: S $\longrightarrow$ M, obs_Ms: J $\longrightarrow$ M-**set**

axiom

$\forall$ n:N $\cdot$ **card** obs_Ss(n) $\geq 1 \wedge$ **card** obs_Js(n) ≥ 2

$\forall$ n:N $\cdot$ **card** obs_Ss(n) $\equiv$ **card** {obs_Si(s)|s:S $\cdot$ s $\in$ obs_Ss(n)}

$\forall$ n:N $\cdot$ **card** obs_Js(n) $\equiv$ **card** {obs_Ji(c)|j:J $\cdot$ j $\in$ obs_Js(n)}

...

type

Nm, Co, Ye

value

obs_Nm: N $\longrightarrow$ Nm, obs_Co: N $\longrightarrow$ Co, obs_Ye: N $\longrightarrow$ Ye

Software Engineering

- Software Engineering

  $=$ **Domain Engineering**

  $\oplus$ Requirements Engineering

  $\oplus$ Software Design

- We have yet to more fully understand the interplay between knowledge engineering and domain engineering (K&DE).
- Is knowledge engineering a proper part of domain engineering ?
- Maybe K&DE is one discipline ?

Requirements Engineering

- Requirements Engineering

  $=$ **domain requirements engineering**

  $\oplus$ **interface requirements engineering**

  $\oplus$ **machine requirements engineering**

- is about
  - ⋆ systematic to formal ways
  - ⋆ of "turning" oftentimes **non-computable** domain and knowledge descriptions
  - ⋆ into prescriptions for **computable** software.

Software Design

- Software design is about
  - ⋆ systematic to formal ways
  - ⋆ of "turning" abstract requirements prescriptions
  - ⋆ into
    - ◇ **correct** and
    - ◇ **efficiently executable**

  program code.

On Methods and Methodology

- Software artifacts are not manifest in the sense of being
  - ⋆ viewable,
  - ⋆ hearable,
  - ⋆ touchable,
  - ⋆ tastable,
  - ⋆ smellable
  - ⋆ or ... physically measurable.
- Their construction thus necessitates
  - ⋆ **a new approach** to **“methodiciy”**
  - ⋆ (being methodological).

On Methods and Methodology (Continued)

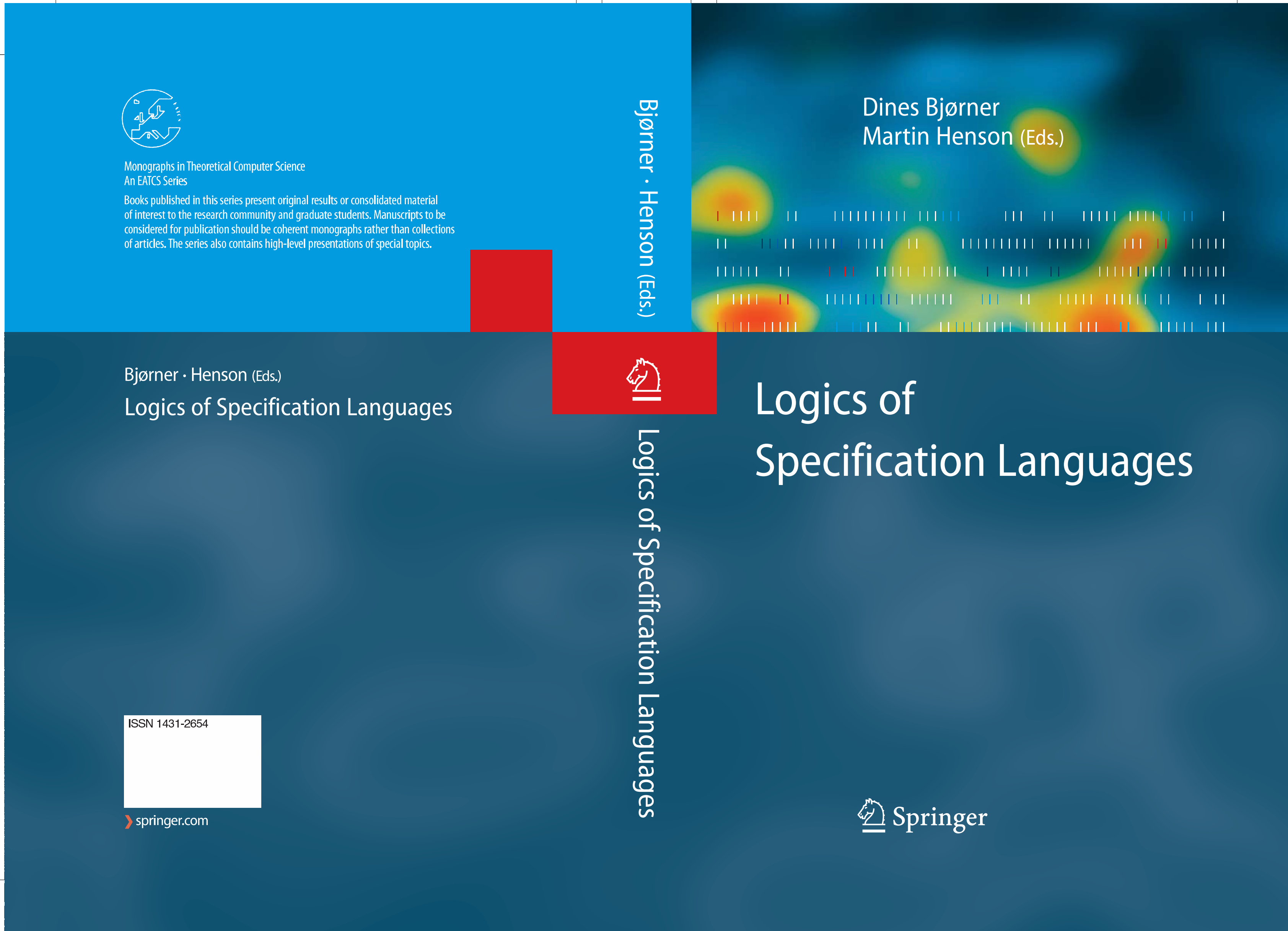
- By a **method** we shall understand a set of
  - ⋆ **principles** for
  - ⋆ **selecting** and **applying**
  - ⋆ **analysis** and **synthesis**
  - ⋆ **techniques** using a set of
  - ⋆ **tools.**
- By **methodology** we shall understand
  - ⋆ the study and knowledge of methods.
- The discipline of software engineering
  - ⋆ **is full of (many claims of) methods !**
- These **principles, techniques** and **tools** are of “a new kind” — different from classical engineering, we claim.

On Formal Techniques

- Correctness of software is of overriding concern.
- To understand a domain, a set of requirements, and software
  - ⋆ we resort to a combination of
  - ⋆ precise **natural language** narratives and
  - ⋆ **formal**, mathematical **specification**s.
- By a **formal technique** we understand a technique
  - ⋆ which applies to formal specifications
  - ⋆ where these specs. have
    - ◇ a **formal syntax**,
    - ◇ a **formal semantics**, and
    - ◇ a **proof system**.
- Propagation of formal techniques (colloquially: "formal methods") seems to invoke **controversy**.

On Informatics Engineering
On Professionalism

- The professional informatics engineers possess the following skills:
  - ⋆ MSc level in computer & computing science,
  - ⋆ IE: knowledge, domain and requirements engineering and software design: $\mathcal{K}, \mathcal{D}, \mathcal{S} \models \mathcal{R}$,
  - ⋆ well-founded in the mathematics disciplines outlined above,
  - ⋆ clear understanding of the rôle of semiotics in SE,
  - ⋆ well aware of the philosophical issues outlined above,
  - ⋆ certified for work in one well-delineated domain, and
  - ⋆ pursues this work **also** using formal techniques:
    - ◇ that is, the professional informatics engineer is versant in varieties of formal techniques, say
      - ∘ *B, RAISE/RSL, VDM-SL, or Z,*
      - ∘ *DC, TLA+, or some other Temporal Logic,*
      - ∘ *Modal, Kripke, Epistemic, Deontic, etc., Logics, and*
      - ∘ *formal foundations of LSC, MSC, Petri nets, Statecharts, etc.*


Monographs in Theoretical Computer Science
An EATCS Series
Books published in this series present original results or consolidated material of interest to the research community and graduate students. Manuscripts to be considered for publication should be coherent monographs rather than collections of articles. The series also contains high-level presentations of special topics.
Bjørner · Henson (Eds.)
Logics of Specification Languages
ISSN 1431-2654
springer.com
Bjørner · Henson (Eds.)
Logics of Specification Languages
Dines Bjørner
Martin Henson (Eds.)
Logics of
Specification Languages
Springer

The Professional Engineer

- The engineer **“walks the bridge”**
  - ⋆ between **science**
  - ⋆ and **technology** (in the sense of technological artifacts)
- The professional engineer
  - ⋆ designs technological artifacts based on scientific insight and
  - ⋆ studies technology in order to acquire new scientific knowledge.

CS, CS & SE

- But there are problems wrt. CS & CS !

  CS: Comput**er** Science, CS: Comput**ing** Science
- Most CS people confuse the two issues.
- Most CS people do not appreciate the work of the other CS people !
- Some CS people use mathematical-looking techniques
  where they could, and in MHO should, use formal specifications.
- Many CS papers suffer from unnecessary "mathematics" !
- The result is that most software engineers
  have a confused understanding of the rôle of foundational science.

On Obstacles to Professional SE

- **Academia:** *lack of scientific honesty | collegiality syndrome*
  - ⋆ We are not teaching professional software engineering
  - ⋆ We are not connecting computer science up to computing science
  - ⋆ Half the colleagues do not understand/appreciate the other half's work
  - ⋆ Academic staff are "barking up the wrong trees", "navel-fixated"
- **Industry:**
  - ⋆ Critical mass syndrome
  - ⋆ Unawareness *generation gap*
  - ⋆ Their own "watchdogs" do not prescribe professionalism — *as in & for other engineering disciplines: certification*
- **Customers** — do not demand professionalism
- **Public government**
  - ⋆ is not spearheading *why no academic 'Amanda' ?*
  - ⋆ no pathfinder projects *why no academic 'EPJ' ?*

A New Kind of Software Engineer
The Background Problem

- Major Software Development Projects Fail
  - ⋆ Exceed Estimated Development Costs
  - ⋆ Exceed Estimated Development Time
  - ⋆ Fail to Meet Customer Expectations
- Many Projects Fail
  - ⋆ The Danish EPJ project
    - ⋄ 17 different interpretations of the term ‘document’
    - ⋄ A number of existing EPJ systems cold not be “harmonised”
    - ⋄ Millions of $s are being wasted
  - ⋆ The Danish unemployment system was very problematic
- In the US of A alone an estimated US $ 20 bio is lost yearly

Software Development ⇒ Software R&D

- Many software development projects are actually research projects
  - ⋆ But many customers are “misled” by software houses
  - ⋆ And there is, in effect, no professional industry “watchdog”
- Software for “unfamiliar” domains need be R&D’ed
  - ⋆ Contracts need be flexible, “gliding”
  - ⋆ Research must account for “delays”, “failures”, “aborts”

A New Kind of Software Engineer

• Analogies — Somewhat "Stretched" •

- **Medical Profession**
  - ⋆ critical diseases treated at hospitals
  - ⋆ where some medical doctors
  - ⋆ are also researchers (profs. at univs.)
- **Architectural Profession**
  - ⋆ many leading designs done in architectural firms
  - ⋆ staffed by architects
  - ⋆ some of whom are also profs. at schools of architecture.
- *&c.*

• Software Scientists •

- Critical new software developments done at software house
- staffed by full-time industry and part-time academics
- where the latter thus have access to more foundational research

Informatics — A New Universe — Concluding Remarks

- I have "painted" an image of an engineering discipline.
  - ⋆ It is **not** based on the **natural sciences**.
  - ⋆ It is **more** based on **mathematics** — with supports from
    - ◇ linguistics cum semiotics and philosophy
- Informatics, I claim
  - ⋆ offers a universe of **intellectual quality**
  - ⋆ in contrast to classical engineerings' universes of **material quantity**
- I have introduced a new discipline, as part of informatics:
  - ⋆ **domain science and engineering:**
    - ◇ where the natural sciences study the universe as given to us,
    - ◇ domain science studies man-made universes:
      - ○ from infrastructure components *viz. transportation*
      - ○ to mechatronics/biological subsystems

Informatics — A Clarification

- **Classical Engineering Sciences**
  - ⋆ **Quantitative, Material**
    - ◇ smaller, faster,
    - ◇ higher capacity
    - ◇ lower energy,
    - ◇ less costly, etc.,
    - ◇ bound by laws of physics.
- **Informatics**
  - ⋆ **Qualitative, Intellectual**
    - ◇ correctness,
    - ◇ intellectual elegance,
    - ◇ fit to human psyche,
    - ◇ bound by laws of mathematics
    - ◇ and by "laws" of philosophy
- **Computing scientists are not gadget builders —**
  They build concepts and are not funded to implement these.

Closing Remarks

Closing Remarks — and Thanks

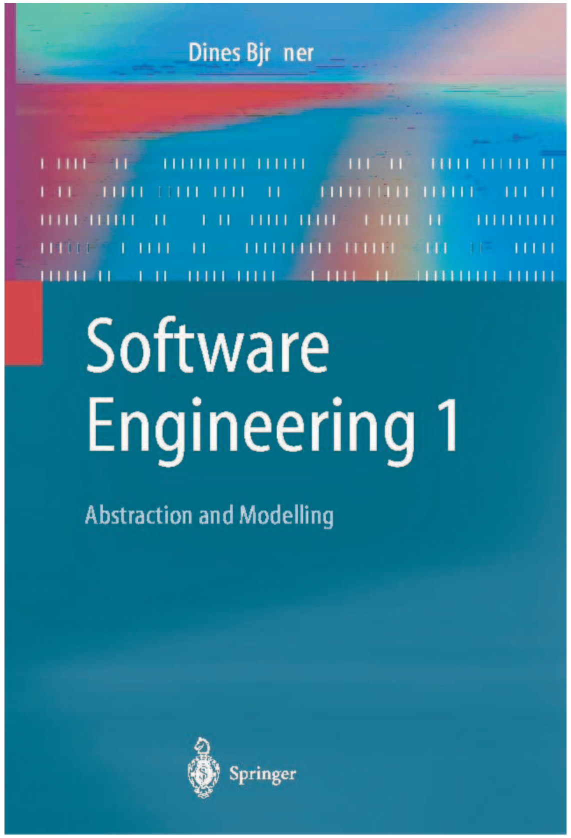
- An era, my era, at DTU; is "vorbei, zu ende". New eras are in sight.
- It was great fun — ... I am grateful to
  - ⋆ Prof. **Per Gert Jensen** for starting all this, "inviting me in"; to
  - ⋆ Prof. **Chr. Gram**

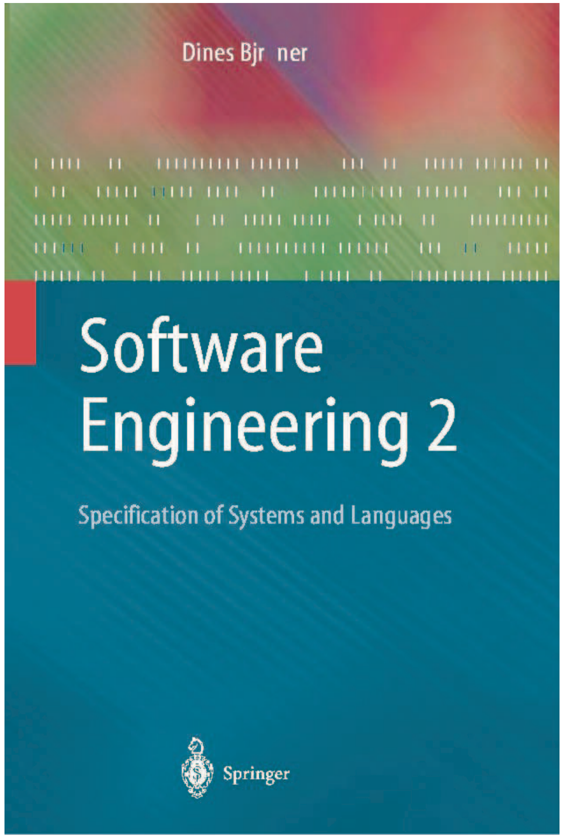
    for his gentle and fine leadership and for his co-starting DDC, providing leverage for much of "this"; to
  - ⋆ Prof. **Jørgen Fischer Nilsson**

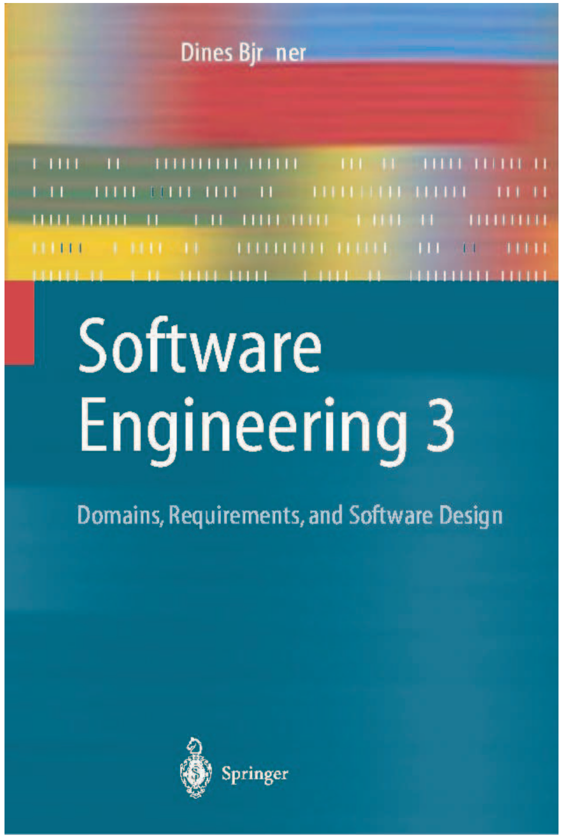
    for always stimulating talks, for steering a scientifically deep and relevant course — keep on, persevere; to
  - ⋆ Dr. **Ole N. Oest** DDCI, Phoenix, Arizona, USA

    for co-initiating the DDC Ada Project, the one project that made DDC a household name on at least three continents; to
  - ⋆ Prof. **Kaj Madsen** for allowing my two sabbaticals; and to
  - ⋆ **DTU** for having provided, for many years, wonderful students and a proper academic and stimulating frame for serious work.

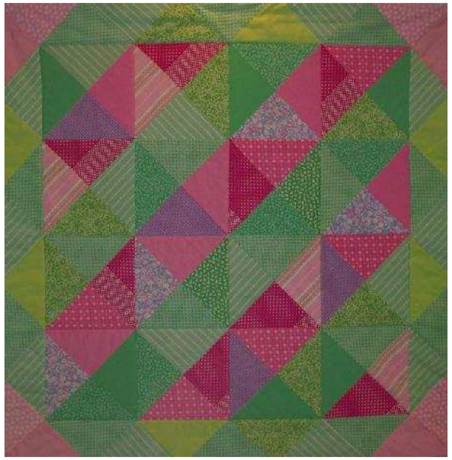
Buy — Read — Practice



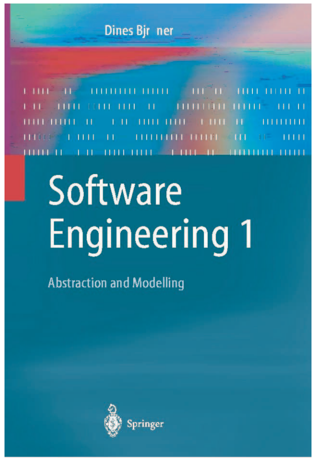


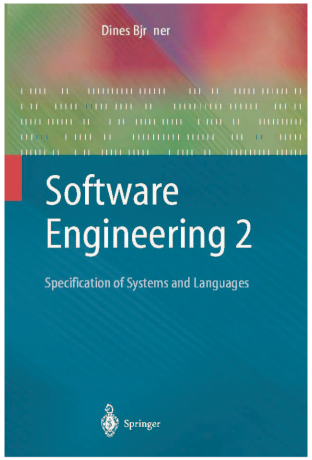


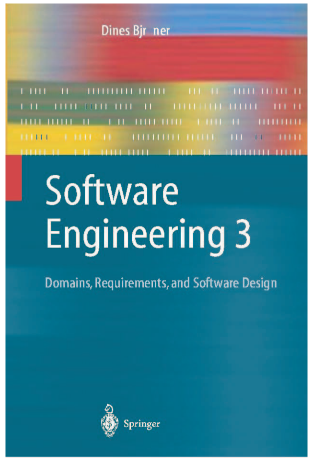
- Thanks to Christian Krog Madsen ↑ and Kirsten Mark Hansen ↑
- and also to Steffen Holmslykkke ↑



Thanks to Kari — the best thing that ever happened to me







Special Tak til Hans Bruun

Kære Hans,

⋆ **Uden Dig intet DDC.**

⋆ Uden Dit enorme arbejde og dybe indsigt intet DDC.

⋆ Mange er Dig taknemmelig

- for at Du på var fødselshjælper mmm. —
- for dem blev **DDC** en finest tænkelig karrierestart.