The Triptych Model of Software Development

— and its relations to SEMAT

Dines Bjørner

DTU Informatics

GTSE/PFSE 2012, Stockholm, 8–9 November

KTH

1. Structure of Talk

2. Definitions

- **Formal Model:** A concise, abstract specification that may be subject to rigorous reasoning — implies use of formal languages: formal syntax, formal semantics, proof systems.
- **Method:** A set of principles for selecting and applying techniques and tools in order to analyse and synthesize an artifact.
- **Software Development:** Domain engineering, requirements engineering and software design + all the tests, model checks and theorem proofs related to the engineering and design documents and their relations, etc.
- **Software:** The full set of documents arising from a completed software project: domain descriptions, requirements prescriptions and software design + all the test data and test results, models and model checks, theorems and proofs relsted to the former documents; system, installation, user, etc. manuals, etc.

3. The Triptych Method — one amongst many

3.1. Development Phases

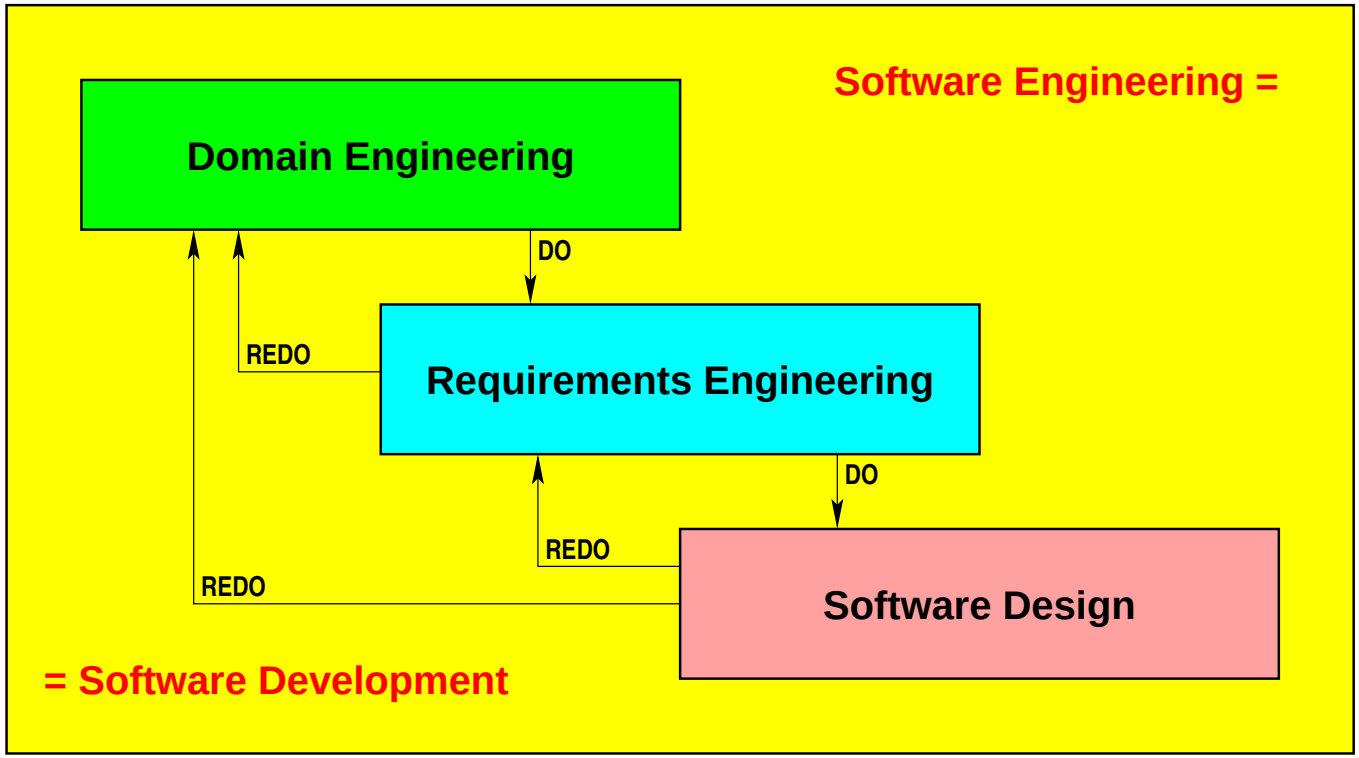

Figure 1: The triptych iterative phase development: Fig. **??** on Slide ??

3.2. Domain Engineering Phase

3.2.1. Generic DE Diagram

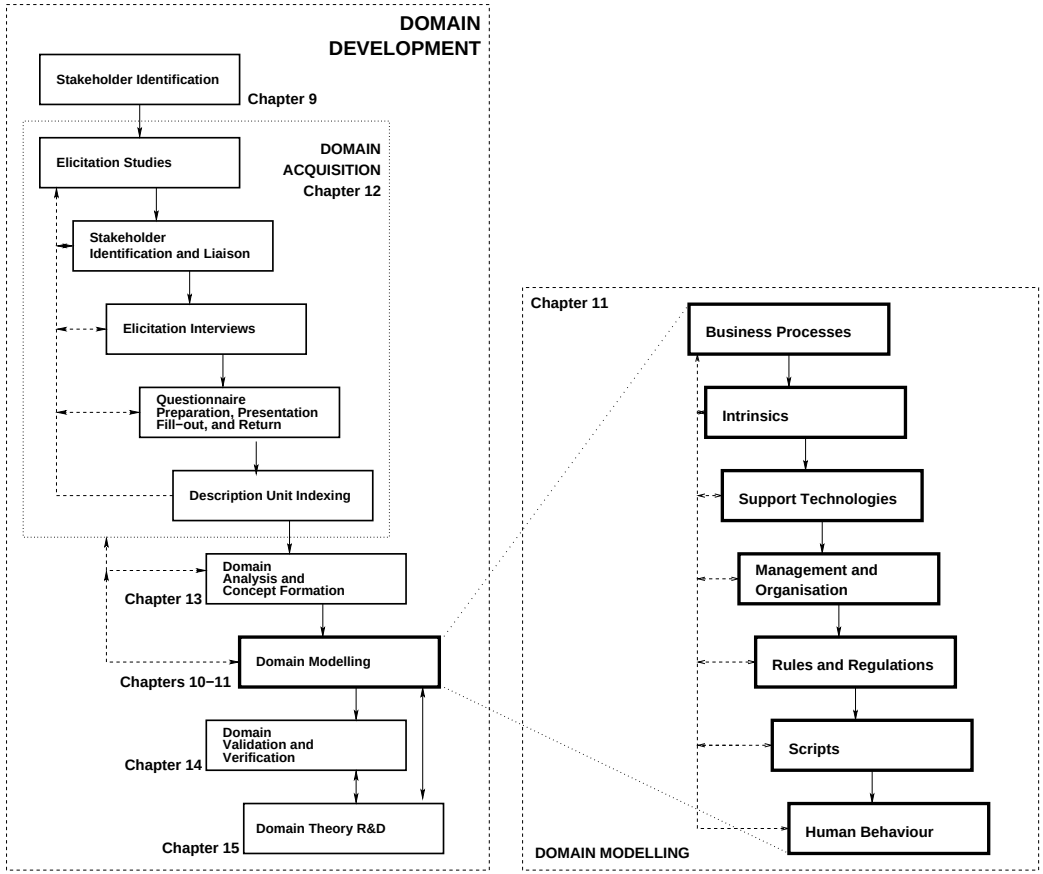

Figure 2: The domain development processes

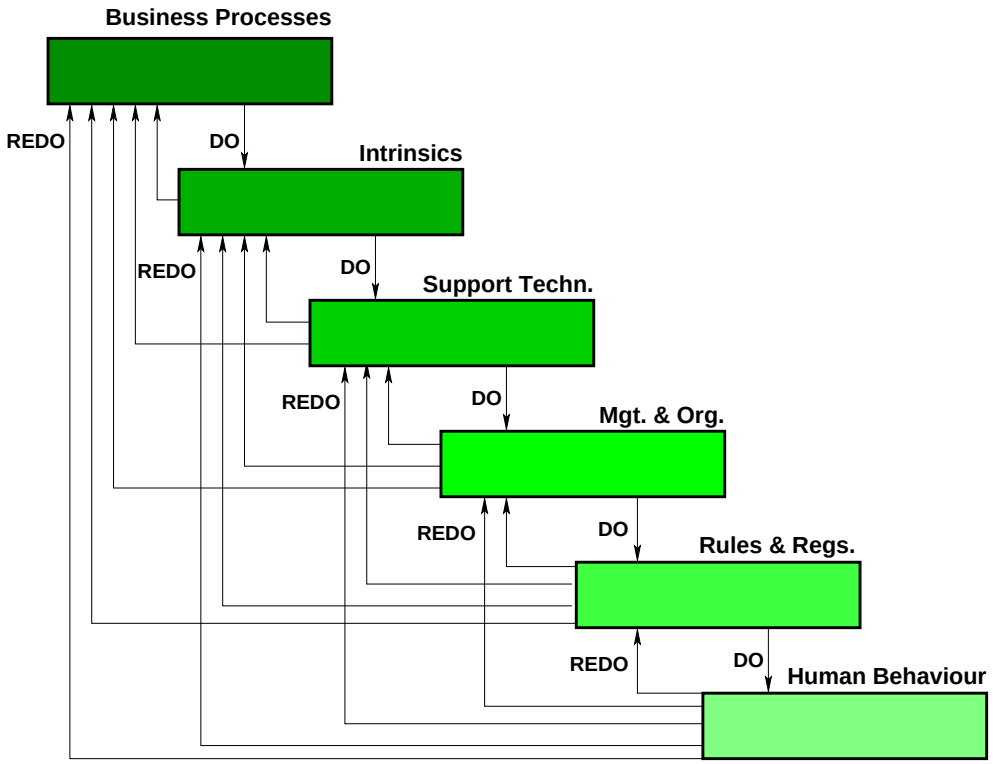

Figure 3: The domain stage iterations: Fig. **??** on Slide ??

3.2.2. Domain Description Documents

1. Information
   - a Name, Place and Date
   - b Partners
   - c Current Situation
   - d Needs and Ideas
   - e Concepts and Facilities
   - f Scope and Span
   - g Assumptions and Dependencies
   - h Implicit/Derivative Goals
   - i Synopsis
   - j Standards Compliance
   - k Contracts
   - l The Teams
     - i. Management
     - ii. Developers
     - iii. Client Staff
     - iv. Consultants
2. Descriptions
   - a Stakeholders
   - b The Acquisition Process
     - i. Studies
     - ii. Interviews
     - iii. Questionnaires
     - iv. Indexed Description Units
   - c Terminology
   - d Business Processes
   - e Facets:
     - i. Intrinsics
     - ii. Support Technologies
     - iii. Management and Organisation
     - iv. Rules and Regulations
     - v. Scripts
     - vi. Human Behaviour
   - f Consolidated Description
3. Analyses
   - a Domain Analysis and Concept Formation
     - i. Inconsistencies
     - ii. Conflicts
     - iii. Incompletenesses
     - iv. Resolutions
   - b Domain Validation
     - i. Stakeholder Walk-Throughs
     - ii. Resolutions
   - c Domain Verification
     - i. Model Checkings
     - ii. Theorems and Proofs
     - iii. Test Cases and Tests
   - d (Towards a) Domain Theory

3.3. Requirements Engineering Phase

3.3.1. Generic RE Diagram

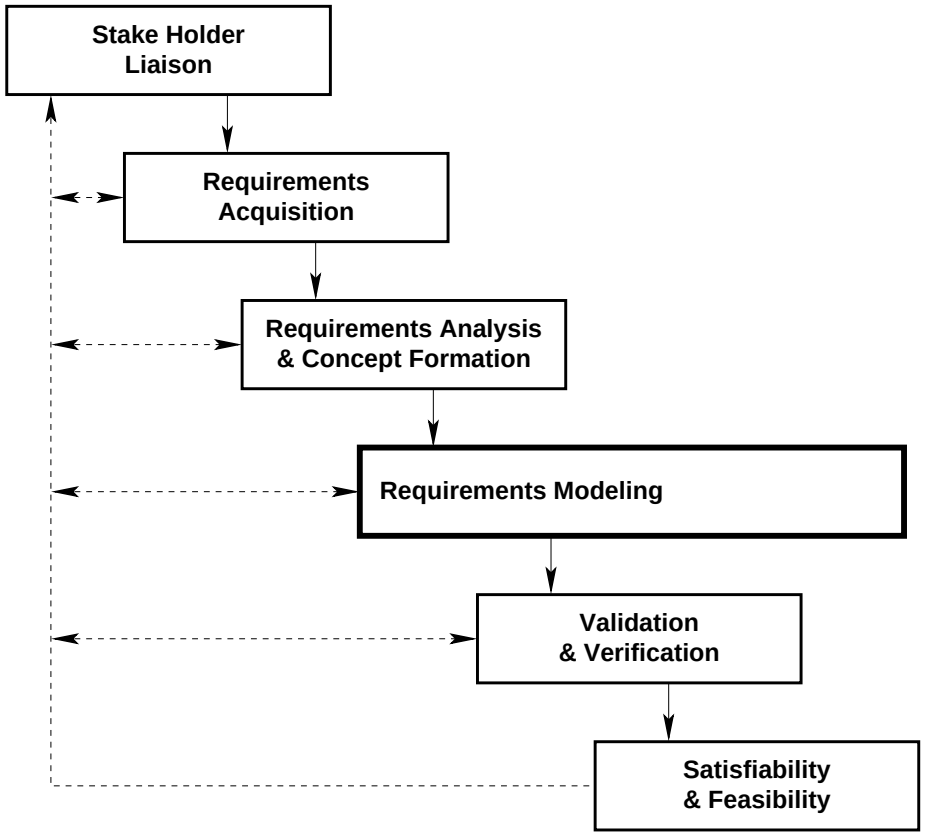

Figure 4: Diagramming a requirements process model

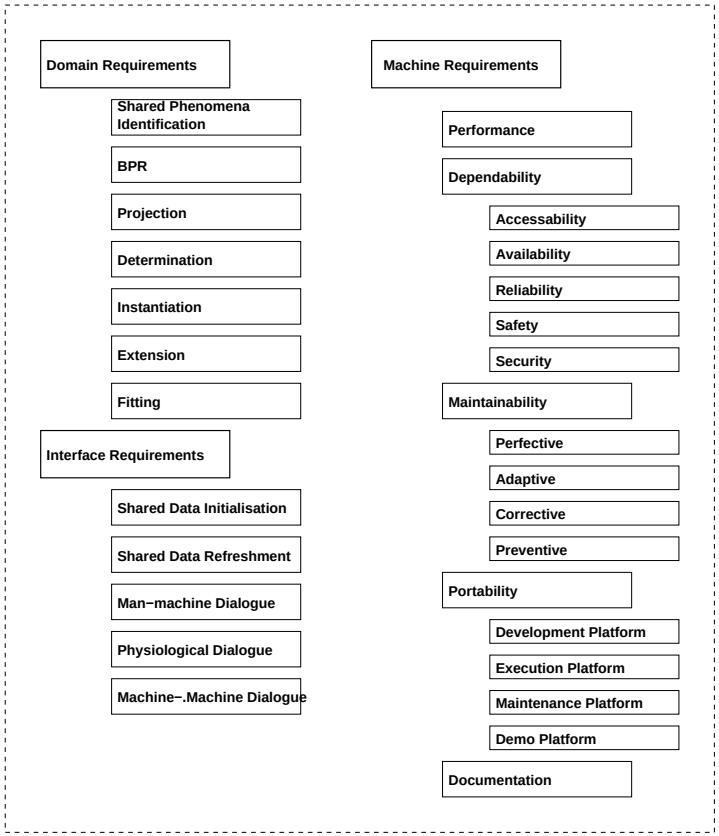

Figure 5: The requirements modelling stage

3.3.2. Requirements: Informative Documentation

1. Information
   - a Name, Place and Date
   - b Partners
   - c Current Situation
   - d Needs and Ideas (Eurekas, I)
   - e Concepts & Facilities (Eurekas, II)
   - f Scope & Span
   - g Assumptions & Dependencies
   - h Implicit/Derivative Goals
   - i Synopsis (Eurekas, III)
   - j Standards Compliance
   - k Contracts, with Design Brief
   - l The Teams
     - i. Management
     - ii. Developers
     - iii. Client Staff
     - iv. Consultants

3.3.3. Requirements: Prescriptive Documentation: Two Slides

2. Prescriptions (1 of 2)
   a Stakeholders
   b The Acquisition Process
      i. Studies
      ii. Interviews
      iii. Questionnaires
      iv. Indexed Description Units
   c Rough Sketches (Eurekas, IV)
   d Terminology
   e Facets:
      i. Business Process Re-enginering
      - Sanctity of the Intrinsics
      - Support Technology
      - Management and Organisation
      - Rules and Regulation
      - Human Behaviour
      - Scripting
      ii. Domain Requirements
      - Projection
      - Determination
      - Instantiation
      - Extension
      - Fitting
      iii. Interface Requirements
      - Shared Phenomena and Concept Identification
      - Shared Data Initialisation
      - Shared Data Refreshment
      - Man-Machine Dialogue
      - Physiological Interface
      - Machine-Machine Dialogue

(2. Prescriptions (2 of 2))

iv. Machine Requirements
- Performance
  - Storage
  - Time
  - Software Size
- Dependability
  - Accessability
  - Availability
  - Reliability
  - Robustness
  - Safety
  - Security
- Maintenance
  - Adaptive
  - Corrective
  - Perfective
  - Preventive
- Platform
  - Development Platform
  - Demonstration Platform
  - Execution Platform
  - Maintenance Platform
- Documentation Requirements
- Other Requirements

v. Full Reqs. Facets Doc.

3.3.4. Requirements: Analytic Documentation

3. Analyses
   a Requirements Analysis and Concept Formation
      i. Inconsistencies
      ii. Conflicts
      iii. Incompletenesses
      iv. Resolutions
   b Requirements Validation
      i. Stakeholder Walk-through and Reports
      ii. Resolutions
   c Requirements Verification
      i. Model Checkings
      ii. Theorem Proofs
      iii. Test Cases and Tests
   d Requirements Theory
   e Satisfaction and Feasibility Studies
      i. Satisfaction: Correctness, unambiguity, completeness, consistency, stability, verifiability, modifiability, traceability
      ii. Feasibility: Technical, economic, BPR

3.4. Software Design Phase

3.4.1. Generic SD Diagram

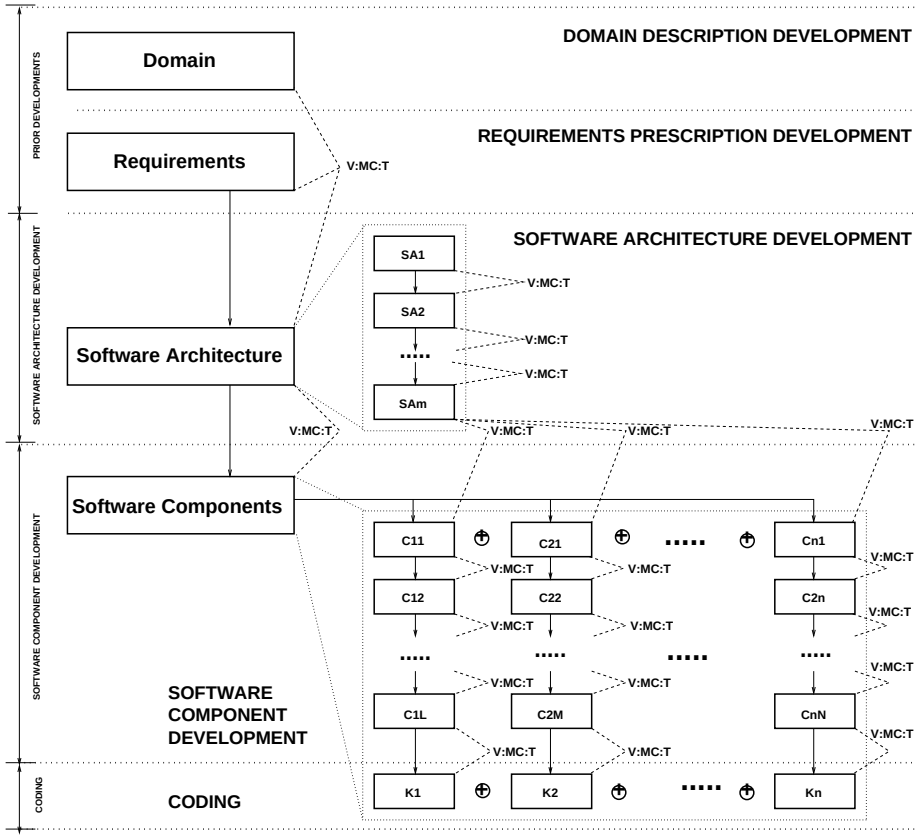

Figure 6: The software design development processes

3.4.2. Software Design Documentation

3.4.2.1 Informative Documentation

1. Information
   - a Needs and Ideas
   - b Concepts & Facilities
   - c Scope & Span
   - d Assumptions & Dependencies
   - e Implicit/Derivative Goals
   - f Synopsis
   - g The Teams
     - i. Management
     - ii. Developers
     - iii. Consultants
   - h Contracts

3.4.2.2 Specification Documentation

2. Software Specifications
   - a Architecture Design ($S_{a_1} \ldots S_{a_n}$)
   - b Component Design ($S_{c_{1_i}} \ldots S_{c_{n_j}}$)
   - c Module Design ($S_{m_1} \ldots S_{m_m}$)
   - d Program Coding ($S_{k_1}, \ldots, S_{k_n}$)

3.4.2.3 Analytic Documentation

3. Analyses
   a Analysis Objectives and Strategies
   b Verification (S_{i_p}, $S_i \sqsupseteq_{L_i} S_{i+1}$)
      i. Theorems and Lemmas L_i
      ii. Proof Scripts $\wp_i$
      iii. Proofs Π_i
   c Model Checking ($S_i \sqsupseteq P_{i-1}$)
      i. Model Checkers
      ii. Propositions P_i
      iii. Model Checks $\mathcal{M}_i$
   d Testing ($S_i \sqsupseteq T_i$)
      i. Manual Testing
         • Manual Tests
         $M_{S_1} \ldots M_{S_\mu}$
      ii. Computerised Testing
         A. Unit (or Module) Tests
         C_u
         B. Component Tests
         C_c
         C. Integration Tests
         C_1
         D. System Tests
         $C_s \ldots C_{s_{i_{t_s}}}$
   e Evaluation of Adequacy of Analysis

3.4.2.4 Legend

Legend:

S Specification
L Theorem or Lemma
$\wp_i$ Proof Scripts
Π_i Proof Listings
P Proposition
$\mathcal{M}$ Model Check (run, report)
T Test Formulation
M Manual Check Report
C Computerised Check (run, report)
$\sqsupseteq$ "is correct with respect to (wrt.)"
$\sqsupseteq_\ell$ "is correct, modulo ℓ, wrt."

4. Software Development Graphs

4.1. Example: CHILL, Ada, Java, ... Developement

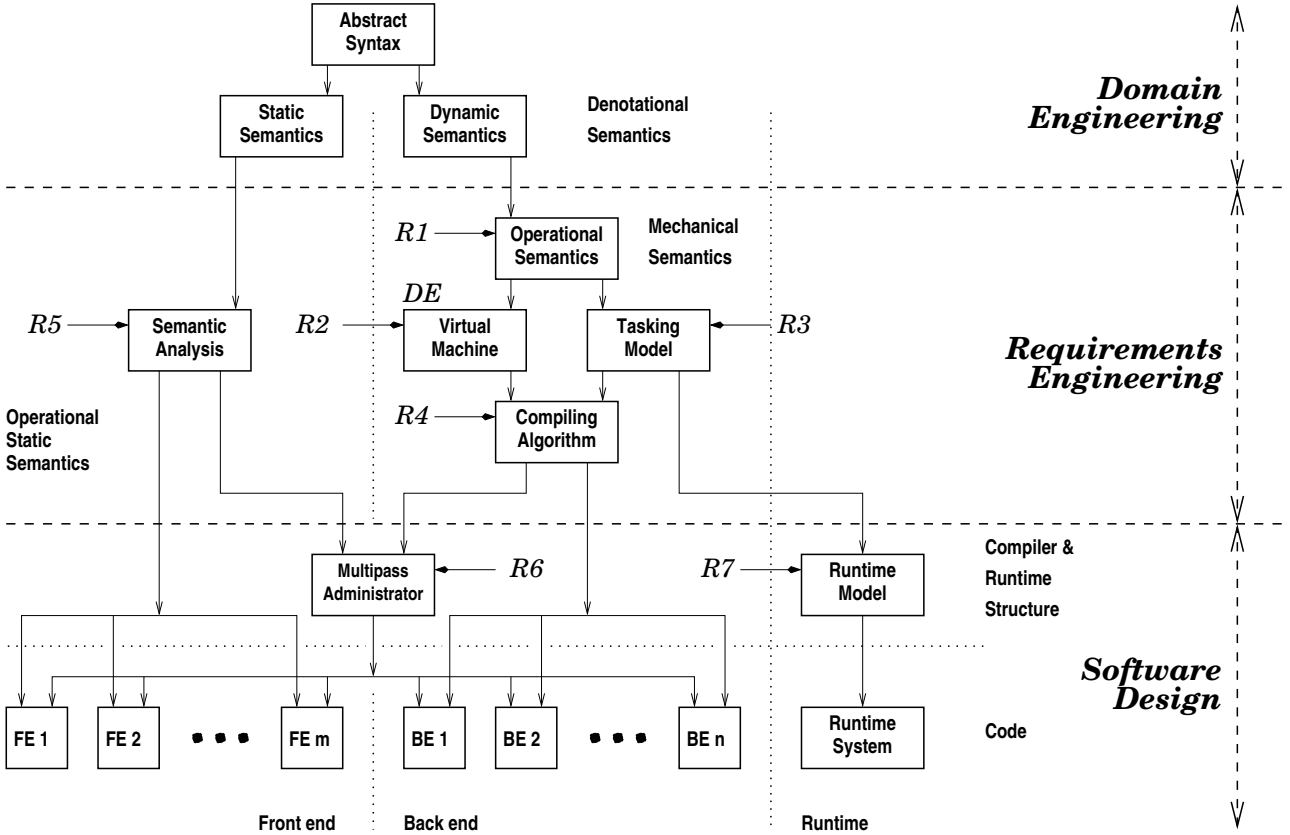

Figure 7: A compiler software

5. Process Graphs

- Replace every SDG node/arc by relevant GDD nodes/arcs.
- The result is a Process Graph
  - "zillions" of nodes and arcs —
  - best kept represented "inside" the computer
  - with "zoomable" and "expandable" SDG and GDD items
  - displaying only "as much as is comprehensible".

5.1. Process Graph Traversals

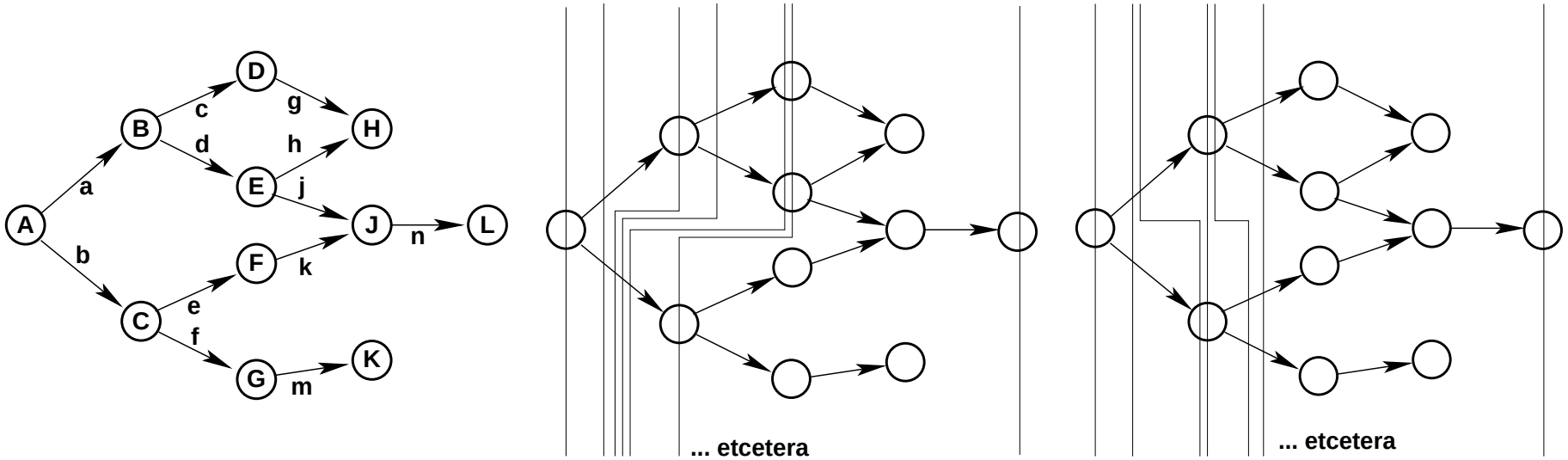

Figure 8: A graph (left) and two (incomplete) traversal traces (center and right)

- ⟨{A},{a,b},{B,b},{c,d,b},{D,E,b},{D,E,C},...,{L} ⟩
- ⟨{A},{a,b},{c,d,b},{D,E,b},{D,E,C},... ⟩
- ⟨{A},{a,z},{X},{D,Y,b},{D,E,C},... ⟩
- ⟨{A},{a,b},{B,b},{a,b},{B,b},{c,d,b},{B,b},{c,d,b},{D,E,b},{D,E,C},... ⟩

- Complete traversal
- Skipped B
- Extraneous x, X, Y
- Redo a, ..., redo B, ...

6. Relations to SEMAT

6.1. Review of the Definition of 'Method'

- **Method:**
  - A set of **principles**
  - for **selecting** and **applying**
  - **techniques** and **tools**
  - in order to
    - **analyse** and
    - **synthesize**

    an artifact.

6.2. Triptych + SEMAT

- It's all very simple, and, ..., yet:
  - with every transition (arc) in a process graph
    - perform the "SEMAT ..." practices
    - as an integral part of the Triptych practices,

    and
  - with every node in a process graph
    - perform the "SEMAT DO" practices
    - as an integral part of the Triptych practices
- The "SEMAT", based on its α**lpha** cards
  - contributes to and makes even more explicit
  - the analysis and selection principles of the Triptych approach.

7. Conclusion
7.1. What Next
7.1.1. αlpha

- Review αlphas;
  - in addition to RE and SD αlphas
    discover a DE αlpha;
  - discover – or refine existing – αlphas
    for formal testing, model checking and proofs;
  - discover αlpha types:
    for the αlpha predicates and
    related state-changing functions;
  - etcetera, etc.

7.1.2. Open Issues

- The αlpha review, above, will lead to revisions of the αlpha structures.
- Etcetera.

8. Many Thanks


Springer
SOFTWARE
ENGINEERING 1
ABSTRACTION AND MODELLING
软件工程
抽象与建模
卷1
Dines Bjørner 著
Software
Engineering 1
CHINA-PUB.COM
清华大学出版社


SOFTWARE
ENGINEERING 2
软件工程
系统与语言规约


SOFTWARE
ENGINEERING 3
软件工程


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


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


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
Dines Bjørner 著
SOFTWARE ENGINEERING 3
Domains, Requirements, and Software Design
清华大学出版社