Review on *Domain Science and Engineering - A Foundation for Software Development* *By Dines Bjørner*

Monographs in Theoretical Computer Science, Springer International Publishing, ISBN 3030734831, 1st ed., 400 pages, 2021.

J. PAUL GIBSON, Telecom Sud Paris

ACM Reference format:
J. Paul Gibson. 2024. Review on *Domain Science and Engineering - A Foundation for Software Development* *By Dines Bjørner* Monographs in Theoretical Computer Science, Springer International Publishing, ISBN 3030734831, 1st ed., 400 pages, 2021. . 1, 1, Article 1 (March 2024), 3 pages.
https://doi.org/XXXXXXX.XXXXXXX

As can be seen explicitly from its title, the book is about a methodological approach for describing and analysing domains. Such domains include both natural and artificial parts, and correspond to some cohesive segment of reality in which human-produced artefacts (including software systems) are expected to operate and interact with other elements of the domain. These include transport (with subdomains such as rail, road, air and sea), critical infrastrucure (with subdomains such as water, gas, electricty), health (with subdomains such as physical, emotional, mental); et cetera. System boundaries are usefully defined by the domain (or domains) to which they belong.

The motivation that drives the book can be found in the Preface's "**Tryptich Dogma**" -

> "In order to *specify* **software** we must understand its requirements
> In order to *prescribe* **requirements** we must understand the **domain**.
> So we must **study**, **analyse** and **describe** domains."

Consequently, the book is principally targetting software engineers who will be working with requirements specifications; either as producers or consumers. (The reviewer would argue that it is not just software engineers, but all engineers of all types of systems, who should be interested in domain engineering.)

Implicit in the book's subtitle is that it includes much foundational material based on discrete mathematics, and that the reader must be willing to interact with such mathematics in order to get the most from the contents. However, as the author makes clear, the book is about *using* mathematics rather than *doing* mathematics, in order to investigate domain science and engineering.

The book is structured into 4 main parts, with a total of 11 chapters, together with a substantial bibliography and supporting technical annexes. The vast majority of the 11 main chapters build on and reuse previous work published by the author in the last decade. It is to the credit of the author that the book reads as a single coherent text, rather than just a collection of separate contributions.

XXXX-XXXX/2024/3-ART1 $15.00
https://doi.org/XXXXXXX.XXXXXXX

Part I - *Setting the Scope* - "does exactly what it says on the tin". Chapter 1 - *Concepts* provides a general vocabulary with short definitions of the most important notions; these are often taken from secondary sources, but some of the most interesting are those proposed by the author himself. It also defines basic software engineering principles that will be followed throughout the book. Chapter 2 - *Domain Philosophy* may be a bit of a surprise to the reader who is new to the author's previous pubications. However, the need for a chapter to introduce the philosophical notions fundamental to understanding the focus of the book is clear from the author's discussion of the "clash" between the informal real world and the formal mathematical model of (parts of) that world. Chapter 3 " - *Space, Time and Matter* - introduces the foundational mathematics, including logic, sets, functions; et cetera. It acts more of a reminder to the reader of what they should already know, rather than as a detailed description from which a reader could learn about these topics. The chapter also nicely links mathematical notation with formal specification languages. It continues with philosophical discussions around the mathematical modelling of space, time and matter; and concludes with a short description of mereology[2] (theory of part-whole).

Part II - *Domains* - includes the core four chapters of the monograph. Chapter 4 - *Domains - A Taxonomy: External Qualities* introduces the concepts of entities, endurants and perdurants[1], with a method for describing and analysing them. It is the first chapter in which we see two different types of explanatory examples: *informal* using structured natural language, and *formal* using more mathematical specification languages. This chapter (like all those in Part II) will be a challenge for readers not familiar with the formalisms being employed (although reviews/summaries are given in the annexes). Chapter 5 - *Domains - An Ontology: Internal Qualities* covers the key notions of unique identification of endurants, mereology, and attributes of endurants. The examples, linking the informal with the formal, illustrate the importance of model validation, and provide a good test of the reader's understanding of the presented material. This chapter is very dense, covering a lot of material and the examples present model descriptions without discussing the challenging process of how such descriptions are produced and validated. Chapter 6 - *Transcendental Deduction* is a very short (a couple of pages) introduction to this sub-field of epistemology. The chapter includes an important example (of a bus and its three *senses*) and how transcendentality links endurants, perdurants and attributes. Chapter 7 - *Domains - A Dynamic Ontology: Perdurants* describes the methodology of transforming parts into behaviours (states that change over time). It reviews different approaches/languages to/for modelling discrete concurrent behaviours, including CSP, Petri Nets and RSL (Raise Specification Language). Again the reader may have to frequently refer to the annexes in order to understand the examples given. As with the previous chapters in Part II, the focus is on understanding the scientific concepts and models. There is a brief discussion on the importance of engineering method, with claims that their approach improves on object oriented methods. The reader may be frustrated that such claims are not examined in more detail. Overall, part II of the book provides a very good treatment of analysing and describing external and internal qualities of endurants. It acknowledges that an equally good treatment of perdurants is missing. Chapter 8 - *Domains Facets* provide a mechanism for structuring a domain in terms of different generic views not specific to that domain. (The *intrinsics* facet correponds to a non generic view which is specific to the domain in question.) The chapter provides examples of 8 different facets/views. This is the least challenging chapter of Part II, and provides insight into some of the software engineering aspects of domain modelling.

Part III - *Requirements* is a single chapter (9) which illustrates one approach to "deriving" requirements prescriptions from domain descriptions. Domain engineering is about modelling the world as it is, and requirements engineering as about modelling the world as we would like it to be. Clearly, the requirements model has to be in some way consistent with the domain model. Also, requirements should be (but are not always) internally consistent[3]. The question of consistency could have been better considered. Overall, the concrete example in the chapter aids the reader in understanding the approach. However, the reader may expect a more explicit discussion of why the domain-driven approach is considered better than other requirements engineering approaches.

Part IV - *Closing* includes 2 short chapters. Chapter 10 - *Demos, Simulators, Monitors and Controllers* is not as formal/rigorous as previous chapters. It speculates about families and varieties of software, their relation

to the "domain" of application, and sketches some ideas with respect to validation and verification. Chapter 11 - *Winding Up* summarises the contributions of the monologue, and emphasises that the book reports on a work-in-progress (albeit one that has been progressing for more than a decade). The challenges for the future of domain engineering are clear, and many are described in the exercises at the end of each chapter.

To conclude this book will (or should) be of great interest to the requirements engineering and formal specification communities. The author's insights into domain science and engineering are of great value. The book is didactic in nature, and would be well suited as a text for teaching undergraduate and postgraduate students.

REFERENCES

[1] Thomas Bittner, Maureen Donnelly, and Barry Smith. 2004. Endurants and perdurants in directly depicting ontologies. *AI Communications* 17, 4 (2004), 247–258.

[2] Aaron J Cotnoir and Achille C Varzi. 2021. *Mereology.* Oxford University Press.

[3] Pamela Zave and Michael Jackson. 1997. Four dark corners of requirements engineering. *ACM transactions on Software Engineering and Methodology (TOSEM)* 6, 1 (1997), 1–30.