The growth of academic fields is an interesting process. It seems to me that most of them follow a similar pattern, at least up to a certain point. They start when a small group of people begin working on a problem, or a set of ideas, that are distinct from those in the mainstream. This group starts meeting to discuss their concern, and this meeting grows into a regular workshop. If there is enough interest in the ideas or problem, the group grows, the workshop turns into a conference, and eventually (all things being well) the ideas that started out as so distinct from the mainstream eventually become the mainstream, and the field can be considered to have reached maturity.

One important marker along this path is the appearance of the first textbook on the subject. That is an indicator that the field is sufficiently mature that there is some kind of consensus on what constitutes it. A further marker, and an indicator of considerable additional maturity, is when the second and third textbooks appear—that is when it is judged that there is enough of an audience to support them. Looked at from this standpoint, Bayesian Artificial Intelligence shows that the field of probabilistic reasoning in AI is now a long way along the path to maturity.

In fact, despite the title, Bayesian Artificial Intelligence is mainly a book about probabilistic causal models, one of the more widely accepted probabilistic approaches to be studied under the AI umbrella. Like Jensen’s Bayesian Networks and Decision Graphs (2001), reviewed here recently, Bayesian Artificial Intelligence does an excellent job of introducing the basic mechanisms of probabilistic networks in a way that will be accessible to most undergraduates (by which I mean that it is not too highly mathematical) but also gives enough detail to make the step up to the primary research literature (or more advanced texts) not too demanding.

Following on from this gentle introduction, the book covers material on more advanced inference mechanisms (including junction trees and logic sampling), before examining various kinds of decision network—probabilistic causal networks augmented with information about utilities. Again the basics are covered in careful detail, before turning to more advanced topics like dynamic decision networks. A chapter on applications of the techniques covered so far, which include some quite detailed descriptions, rounds out the first part of the book.

The remaining parts of the book complement this basic theory, together answering the question ‘How do you build these networks’. Part II describes techniques for learning causal models from data, including how to learn causal structure and how to learn probability values, and including material on all the classic approaches from the field. Part III describes how to go about the process of knowledge engineering when the target system is based on a probabilistic causal model—the case studies drawn from the authors’ prior work are especially useful here. Just as in Part I, the material is carefully aimed to make it accessible for the bulk of undergraduate students, while leading on to the appropriate level for graduate study.

Overall, then, Bayesian Artificial Intelligence is a very good textbook. It gives novices all the information they need to start using the techniques described in the book, assuming that they pick up one of the many tools that implement these techniques, and which are listed in the Appendix. If, hypothetically, novices want to build their own tools, or to get sufficiently deeply into Bayesian

1 Not that there is anything wrong in this, but there are now a wide range of other probabilistic techniques increasingly used in AI, including, for example, Markov decision processes and particle filters, which might also be thought of as Bayesian artificial intelligence.
networks in order to contribute to the field, well they will have to turn to the technical literature, and good points of departure are provided. To put it another way, *Bayesian Artificial Intelligence* does exactly what a textbook should do, and I hope to see it included on the reading list of any course on probabilistic reasoning in AI.

Reviewed by Simon Parsons
Brooklyn College, City University of New York, USA

Reference

doi:10.1017/S0269888905220209

After at least a decade of agent-based research, agents are no longer simply abstract artefacts or theoretical notions shared between AI practitioners, but rather have emerged as well understood, deployed entities appearing within a plethora of multi-agent systems (within industrial as well as academic contexts). Since the early 1990s, many facets of agent research have received attention: from communication metaphors, agent communities (and corresponding architectures), and negotiation and reasoning mechanisms, to agent architectures and agent-oriented programming languages. Research into the latter has resulted in the emergence of a variety of platforms and tools that facilitate the creation of agent systems. This includes the specification of individual plans, architectural designs (such as BDI), desired emergent behaviour, and the resultant roles that different classes of agents may assume. However, for there to be a growth in the successful transition of such technology from academia to industry, it is critical that well understood design methodologies emerge for agent-based systems, and that such methodologies are accessible to students and IT professionals in a clear and concise way.

The PROMETHEUS methodology, described in detail in Padgham and Winikoff’s book ‘*Developing Intelligent Agent Systems*,’ is one of the most mature methodologies currently proposed. Developed since the mid-1990s, it consists of a detailed process for specifying, designing, implementing and testing/debugging agent-oriented software systems. Using both graphical and structural elements, a top-down design framework is provided, by initially considering high-level concepts, such as goals, plans (that achieve the goals), the necessary percepts and actions, and different scenarios in which the goals can be achieved. Additional artefacts can be constructed (through successive iterations and refinements of the design) which result in a transition through three design phases: the *System Specification*, *Architectural Design* and *Detailed Design* phases.

• The *System Specification* phase focuses on identifying the goals and basic functionalities of the system, along with the inputs (percepts) and outputs (actions).
• The *Architectural Design* phase uses the outputs from the previous phase to determine which agent types the system will contain and how they will interact.
• The *Detailed Design* phase looks at the internals of each agent and how it will accomplish its tasks within the overall system.

This book, aimed primarily at industrial software developers and undergraduate students, takes a pragmatic, and easy to follow journey through this design process, grounded by the development of a case study (an ‘Electronic BookStore’, listed in detail in Appendix 1). Aimed primarily at industrial software developers and undergraduate students, it does not assume a background in
agents. Whilst some notions of agency are introduced within the first two chapters, the book avoids dwelling on the different facets of the agent research field, and rather focuses directly on the elements necessary for designing an agent system. Equally, the book remains essentially agnostic with regards to implementation framework, describing a design mechanism that could be applied to various platforms.

The book is divided into five main parts, and includes appendices that provide additional detail on some of the notation used, as well as a full listing of the descriptor forms used in the book. The first part (consisting of chapters 1–3) introduces the reader to the concept of agents, briefly justifying the use of agents for different classes of problems, and then expands on agent-oriented design methodologies, by introducing the different concepts necessary for building agent systems. PROMETHEUS and its principal design stages are then introduced and contrasted to other agent-oriented methodologies.

The second part (chapter 4) commences the design process with a high-level specification of Goals, Functionalities, Scenarios, and Interfaces that constitute the System Specification phase. Each of the system-level artefacts are presented, and the constituent fields within their descriptor forms are described. Approaches for refining the design are then described, followed by heuristics for checking completeness and consistency of the design.

The third part (chapters 5–7) focuses on the Architectural Design phase, consisting primarily of the decision on the types of agents that would be used in the system, understanding the necessary interactions between agents, and then developing the overall system structure. The fourth part describes the Detailed Design phase, where the agents themselves are decomposed into specifications of their capabilities and processes. The final part (chapter 10) preserves the pragmatism of the whole book, by illustrating how the resulting design could be implemented on an existing agent platform; the one used here is JACK (developed by the company Agent Oriented Software), though, as the authors point out, there are several other agent platforms that could also be used. Although little code is actually given (it would be unreasonable to include code that actually implemented the resulting capabilities within the book), this chapter conveniently summarizes the significant artifacts generated by the design, and could easily be used as a guide for mapping these artefacts to other agent platforms.

Throughout the book, emphasis is placed on the designer fully specifying each of the artefacts and their descriptor forms at each part of the design process, with checks for completeness and consistency. Useful tips and ‘rules-of-thumb’ are presented to simplify the process, and to identify possible pitfalls or problems. Examples are used in each of the phases to illustrate the design process, in addition to the bookstore case study which is developed throughout the book. Further, the PROMETHEUS design tool can be obtained from http://www.cs.rmit.edu.au/agents/prometheus.

Despite the fact that a detailed bibliography is provided, ‘Developing Intelligent Agent Systems’ is somewhat light on the theory behind agent-oriented programming and system design, and avoids details regarding architectural components of large-scale agent systems (such as issues to do with discovery, mobility, etc.). Likewise, little is said about issues pertaining to heterogeneous, or open multi-agent systems. Although heterogeneity is of interest to many researchers, most agent deployments are designed to achieve specific goals and are typically implemented using a single agent development platform. Thus, the limitations of this book are actually its strengths, as IT developers can pragmatically focus on rapidly designing and deploying a working multi-agent system.

To conclude, the authors have done an excellent job in describing the different stages involved in designing agent artefacts with PROMETHEUS, and clearly demonstrate how these can be used to design, analyse, and refine multi-agent systems by means of a detailed running case study. This book is a valuable contribution, not only in terms of teaching agent-oriented software engineering and considering design decisions associated with developing a multi-agent system, but in the pragmatic sense of ‘how are agents actually built’. For agent-based research to have an impact on
IT and software development, agent principles and their use should be clearly accessible to IT professionals charged with the task of implementing agent systems. This book goes along way towards achieving this goal.

Reviewed by Terry R. Payne
University of Southampton, UK

The Semantic Web Primer by Grigoris Antoniou and Frank van Harmelen, MIT Press, 238 pp., $40.00, ISBN
doi:10.1017/S0269888905230205

Like ancient Gaul under Caesar, the history of the Internet to date is divided into three parts. The first part, the premodern era, is the construction of the basic network and the provision of the first services across it. This enabled a generation of computer scientists, like me, to cut our research teeth while transferring files by ftp, sending email, and wasting our days reading newsgroups like alt.politics.uk. While at this point the impact of the Internet was limited, it was clearly very handy to be able to keep in touch with colleagues across the world (well across those parts of the world that were actually connected to the Internet) without the need for letters, to be able to share code with collaborators without having to send diskettes around, and to be able to argue about politics without having to go down the pub.

The second part, the modern era, began with the World Wide Web and the first Mozilla browser. While the underlying capabilities of the Internet did not change because of this—its underlying use remained the transmission of files—the perception of the technology fundamentally changed (the World Wide Web was a great PR makeover for the Internet). As a result usage grew massively, and a wide range of new services were developed. Amongst other things, these now allow me to keep in touch with overseas colleagues using instant messaging and VoIP calls, to share code through SourceForge, and to see what politicians are writing in their blogs. And I can buy just about anything I need without leaving my desk.

However, despite all this development, using the Internet remains essentially a hands-on experience. If I want to purchase some service, I have to locate it on Google, then browse the relevant site, extract the information from some piece of text (or, on a particularly badly designed site, from some graphic) and then fill out a form or send an email. All rather twentieth century (and usually not very far from using a mail order catalogue, which is not only twentieth century but early twentieth century)—most of the value that online shopping adds to a mail-order catalogue is the speed with which an order is delivered by the merchant. What I’d really like is to be able to delegate much of the sorting and searching, and preferably even a lot of the form-filling, to some program. The problem with doing this is that the modern era Internet is a free-form mess of text of the kind that programs have great difficulty in being able to handle intelligently.

The aim of the Semantic Web is to overcome this kind of difficulty and hence usher in the third, postmodern, era of the Internet in which far more of our use of it can be automated. This will be possible because the Semantic Web, so its proponents claim, will provide meaning for the content on Web pages, meaning that can be understood and acted on by programs. This is a grand aim, and the technology for achieving it is still very much an active research topic (or more correctly, a set of related topics). The Semantic Web Primer is an early attempt to summarize the tools and techniques that will form the basis of the Semantic Web.

Overall, I think that The Semantic Web Primer does a very competent job. The book provides a solid introduction to the use of XML for creating structured document, the use of RDF and RDF Schema to describe Web resources, and the use of OWL for building ontologies, covering each of these three aspects in a separate chapter. These topics are sensible choices, being standard technologies and those that readers of the book are likely to use in developing Semantic Web applications. The book then goes a little further than would strictly be necessary to get readers up
and developing, devoting a chapter to first-order logic and its use in providing a semantics for the techniques introduced in the previous chapters.

In my opinion, the four chapters just described form the heart of the book and are the most likely to prove useful to the students whom I imagine will be its primary audience. The last two chapters—one giving a high-level view of some Semantic Web applications and another sketching how to create an ontology—strike me as less substantial, though maybe will appeal to some readers.

Of course, the appeal of this book will largely depend upon one’s view of the Semantic Web. While everyone, I think, will find it well written, correct, and careful to pitch its explanations at the right level of detail for an introductory text, I don’t doubt that Semantic Web zealots will find The Semantic Web Primer a more satisfactory book than those who are more skeptical. I am a skeptic, largely because I struggle to see where the semantics come in—to my jaundiced eye all Semantic Web technologies provide is a structured way of arranging an arbitrary set of symbols. To me this falls short of a semantics which should provide a means of tying those symbols to objects in the real world. However, providing such a ‘concrete’ semantics is not only a hard problem, but also seems to be outside the scope of the Semantic Web as a whole, and so it is not surprising to find it omitted from a book like The Semantic Web Primer.

Reviewed by Simon Parsons
Brooklyn College, City University of New York, USA