CHAPMAN & HALL/CRC INNOVATIONS IN
SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT

Building Enterprise Systems with ODP

An Introduction to Open Distributed Processing

Peter F. Linington
Zoran Milosevic
Akira Tanaka
Antonio Vallecillo

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Chapman & Hall/CRC Innovations in Software Engineering and Software Development

Series Editor
Richard LeBlanc
Chair, Department of Computer Science and Software Engineering, Seattle University

AIMS AND SCOPE

This series covers all aspects of software engineering and software development. Books in the series will be innovative reference books, research monographs, and textbooks at the undergraduate and graduate level. Coverage will include traditional subject matter, cutting-edge research, and current industry practice, such as agile software development methods and service-oriented architectures. We also welcome proposals for books that capture the latest results on the domains and conditions in which practices are most effective.

PUBLISHED TITLES

Software Development: An Open Source Approach
Allen Tucker, Ralph Morelli, and Chamindra de Silva

Building Enterprise Systems with ODP: An Introduction to Open Distributed Processing
Peter F. Linington, Zoran Milosevic, Akira Tanaka, and Antonio Vallecillo

Building Enterprise Systems with ODP

An Introduction to Open Distributed Processing

CHAPMAN & HALL/CRC INNOVATIONS IN
SOFTWARE ENGINEERING AND SOFTWARE DEVELOPMENT

Building Enterprise Systems with ODP

An Introduction to Open Distributed Processing

Peter F. Linington
Zoran Milosevic
Akira Tanaka
Antonio Vallecillo

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group an **informa** business

A CHAPMAN & HALL BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20110727

International Standard Book Number-13: 978-1-4398-6626-9 (eBook - PDF)

Contents

List of Figures

This page intentionally left blank

About the Authors

Peter F. Linington is Emeritus Professor of Computer Communication at the University Kent in the United Kingdom. He has been involved in the standardization of the ODP Reference Model and its various supporting standards since the activity started. He has co-chaired WODPEC, the main workshop in this area, since its inception. His recent research interests cover architectural description, use of policies and model-based techniques. Further information can be found at `http://www.cs.kent.ac.uk/people/staff/pfl/`.

Zoran Milosevic is a Principal of Deontik Pty Ltd., a small consulting and software company specializing in the planning, development and deployment of enterprise systems. He was involved in the standardization of the ODP Enterprise Language, and with several OMG standards, while working for the Distributed Systems Technology Centre (DSTC), based in Brisbane. He was the founder of IEEE's Enterprise Distributed Object Computing (EDOC) conference. Further information can be found at `http://deontik.com/About/Zoran.html`.

Akira Tanaka is a founder of view5 LLC, a small consulting company in Japan, specializing in applying viewpoints and model-based approaches to software development. He has been involved in RM-ODP standardization from its early days. While with the Hitachi Ltd. Software Division, he led the ODP committee of INTAP in Japan, and participated frequently in EDOC's WODPEC. He was also active in OMG, including as a contributor to the UML Profile for EAI specification and SoaML RFP. Further information can be found at `http://www.view5.co.jp/`.

Antonio Vallecillo is Professor of Languages and Information Systems at the University of Málaga, Spain. His research interests include open distributed processing, model-based engineering, componentware and software quality. He was co-editor of *ISO/IEC 19793 (UML4ODP)* and of the revised versions of RM-ODP Parts 2 and 3 *(ISO/IEC 10746-2/3)*. Further information can be found at `http://www.lcc.uma.es/~av`.

This page intentionally left blank

Foreword

Working as the rapporteur for Part 3 of the ISO/ITU-T Basic Reference Model for Open Distributed Processing (RM-ODP) was one of the most stimulating periods in my professional career and I am grateful to the authors for asking me to write a foreword to their excellent book describing the model and its application to the design and specification of practical distributed systems. The role of a foreword is to set the scene for the reader and, to that end, I offer a personal perspective on the origins of RM-ODP.

RM-ODP was in many ways a breakthrough in its approach to standardization, arising from a decade of fast-paced innovation in telecommunications and computing technology. Prior to RM-ODP, international standards for networking had been focused on the needs of mainframe computer operating systems for file transfer, remote job control and remote terminal access over relatively slow network links provisioned by regulated telecommunications providers. The ISO Open Systems Interconnection family of standards and its associated seven-layer reference model were developed to meet these needs. For point-to-point services such as file transfer, modelling communication as a protocol state machine moving messages back and forth between two service state machines, one at each end of the communications path, was sufficient. However, as OSI evolved to include multi-party distributed applications, limitations of the OSI model became clear. There was the need for a new framework that could better represent system-oriented concepts such as network management, network security and directory services. The missing capabilities from the OSI reference model were the ability to describe system structure and to model interactions richer than just basic data interchange.

During the 1980s and 1990s, there were huge changes to the landscape of computing driven by the advent of the microprocessor. Minicomputers brought computing out of the data centre and were rapidly followed by single-user workstations and in due course the now ubiquitous personal computer. In parallel, there was an equally revolutionary change sweeping through computer networking, driven by the growth of Local Area Network (LAN) technologies. LANs permitted the low-cost interconnection of computers by high speed links, leading to the appearance of *client-server* architectures in which a powerful server computer provides print, file and database services to a group of smaller user workstations and personal computers. Alongside client-server architectures there was also a parallel evolution of network and distributed operating systems, which followed a decentralized architecture. The client-server

model came to dominate as the personal computer became more popular, but the peer-to-peer model remained an important technology for building powerful servers from clusters of machines, a strand of development which has continued through to modern parallel high-performance supercomputers for scientific processing and from there to the scalable highly parallel architectures of modern data centres for online services and *cloud computing*.

The co-evolution of operating systems and LANs spawned an explosion in new network protocols and services. The greater bandwidth and reliability of LANs compared to earlier wide area telecommunications overlaid on the telephone network removed many of the constraints that had driven the design of the OSI standards. The relevance of OSI was further diluted by the growing adoption of the Internet protocols IP and TCP as UNIX emerged as the dominant workstation operating system of the time. At the same time, the implementation of TCP/IP for the personal computer displaced proprietary client-server protocols, making TCP/IP the *de facto* standard.

With the growth of LANs and client-server computing, the emphasis for standards moved from the protocols as networking functions like file transfer to more generalized access to operating system services and libraries for building distributed applications. For example, *Remote Procedure Call (RPC)* protocols allowed functions on remote machines to be invoked through programming language procedure calls, albeit with more complex failure modes. It was a small step from RPC to *Network Objects*, which generalized the model to method invocation on remote objects and introduced the concept of a *Remote Object Reference* as a pointer to an object that could be given to other computers to enable them to invoke the methods of the object. With these facilities, it became common to talk of a network object as something that provided a *service* defined in terms of the set of method calls provided by the object.

A number of RPC and network object systems were constructed building on these ideas, and standards began to emerge both for the programming interface and the protocols for interworking. First to appear was the Open Software Foundation's Distributed Computing Environment (OSF/DCE) based on RPC and then subsequently the Object Management Group's Common Object Request Broker Architecture (CORBA) and associated *CORBA services*. Both were examples of what came to be commonly described as *middleware* — a layer of distributed computing infrastructure that sat between the basic machine operating system and the (distributed) application.

As these technologies were applied to practical distributed applications, it became clear that while RPC and network objects were useful basic system components, they didn't address the need to build dependable systems and so further developments in middleware provided more advanced facilities such as atomic transactions, support for service replication, service migration and so forth. Generally, these took one of two forms: *explicit*, in which the capabilities were exposed to the application programmer or *transparent*, in which the capability was hidden behind the RPC or network object model. The

explicit model was more complicated to program, but offered greater control and the ability to exploit application-specific knowledge to improve system performance, whereas the transparent model, as its name suggests, required no special intervention on the part of the programmer.

All these technology developments defined the context in which RM-ODP was developed, with the ambition to define a reference model that would provide a descriptive framework for creating standards for interoperability between systems, to allow for the construction of such systems from components glued together by *middleware* and for (distributed) applications to be portable between different vendors' technologies.

During this period, I had personally been involved in several of the technological developments, first as an academic researcher in the Computer Laboratory at the University of Cambridge, England helping build the Cambridge Distributed System [86] and then subsequently as the Chief Architect of ANSA, a project funded by a consortium of computer and telecommunications suppliers. ANSA started as part of the UK government Alvey Programme, and subsequently became part of a European Union funded project called *Integrated Systems Architecture*. Starting slightly earlier than OSF and OMG, the vision for ANSA had been to develop a practical architectural framework and supporting components that could encompass the new distributed computing concepts demonstrated in research prototypes at that time. Alongside the architecture, the ANSA project also developed a reference implementation, called ANSAware, which was used by many of the project partners to build, evaluate and deploy practical distributed systems.[1]

Concepts from ANSA found their way into OSF/DCE, OMG/CORBA and ISO/RM-ODP, and there was close cooperation between the three organizations and the teams involved in each activity, with some overlap in membership. While OSF and OMG focused on producing standards for specific distributed computing functions, the RM-ODP gravitated towards providing a higher-level reference model into which such technical standards could be placed and with which the means of interoperation between systems using these standards (or others) could be discussed.

Possibly the most important idea to come to RM-ODP from ANSA was the concept of *viewpoints* (which in ANSA were called *projections*). These arose initially from a desire to separate discussion of the engineering structure of a distributed system as a composition of computers, communication links, operating systems and middleware components from the abstract distributed computational model seen by the programmer in terms of network objects, interfaces, concurrent threads and so forth. We also recognized that the ANSA computational model could be instantiated over several different engineering models, depending on what kind of transparency was required. This led to the concept of having separate *languages* for describing systems from each viewpoint and the need to be able to show consistency between models of the

[1] An archive of ANSA project documents can be found at `http://www.ansa.co.uk`

same system expressed in each such language. To do this, ANSA borrowed from Sowa's work on conceptual structures [93] by treating the architecture as an existential graph in the form originally proposed by the logician C. S. Pierce. The graph showed relationships between concepts in the architecture and the term *projection* was taken for its mathematical interpretation as a slice through the graph that included every architectural concept in the graph, at some level of abstraction. This gave us the means to explain *correspondences* between viewpoints, and the adoption of a logic-based approach opened the door for other colleagues with an interest in formal methods to apply rigorous mathematical specification techniques to the development of RM-ODP.

In adopting ideas from object-oriented programming as the basis for the ANSA computational model, we struggled with concepts such as *inheritance*, *class* versus *type*, *object* versus *interface* and so forth, not helped by great debate about the relative importance and relationship between these ideas in the programming language community. In the end, we settled on objects as a unit of modularity and encapsulation as the key property in distributed systems and relegated *class* and *inheritance* to be software engineering concepts relating to how specifications are organized. We further took the view that an object could exhibit multiple interfaces, with different interfaces possibly encapsulating different partitions of the object state and giving different kinds of service to different types of client. We talked about *selective transparency* as the abstraction of engineering viewpoint capabilities, controlled by engineering (that is, system management) interfaces in the computational view. The computational viewpoint turned out to be a very powerful system modelling tool but gave problems to colleagues who wanted to implement ANSA directly in object-oriented programming languages, forcing them either to restrict the ANSA model or to use programming language objects to represent both ANSA interfaces and ANSA objects.

The two initial viewpoints were not sufficient for the needs of ANSA or RM-ODP. Taking ideas from colleagues working on OSI protocol conformance testing, we introduced a *technology viewpoint* to give us the means to state which standards applied to specific interfaces in a system and how to go about testing for conformance at those points. To talk about topics such as system management and security we found the need for a language to describe the purpose for which a system was intended, the system's boundaries and the roles undertaken by people using the system. This led us to introduce an *enterprise viewpoint*. Additionally, we wanted to be able to look at a system in high-level terms as an information processing system, without committing to a specific computational structure, leading us to add an *information viewpoint* that allowed for specifications in terms of a conceptual schema for the information handled by a system and an information flow model for the processes by which that information is processed.

The enterprise viewpoint gives terms for describing organizational structures, policies for security, dependability, quality of service and other so-called *non-functional* requirements and the roles of human actors in the system. The

organizational concepts centred on the notion of *federation* representing the notion that many systems arise from the interconnection of previously autonomous systems (for example, as in *enterprise application integration*). In a federation neither system is subordinate to the other; rather, there must be an agreement of what is interconnected, what interactions can occur at those interconnections, and the meanings that are given to the interactions. In this last respect, ANSA work based on Searle's [90] development of Austin's *speech act theory* [48] provided some of the foundations. It gave RM-ODP the concepts of *performative actions* (ones that change the state of affairs in the system, such as making an online purchase) versus purely *informative actions* to exchange data (for example, looking up prices in an online catalogue). By expressing system behaviour in terms of its effect on the external environment, the enterprise viewpoint provides tools needed to express system policies and the desired outcomes and impact of system behaviour. Expressing correspondences between the enterprise viewpoint of a system and the other viewpoints is then essentially an exercise in showing how the technical system meets the external requirement — in other words, showing its fitness for purpose.

An associated concept that came with federation was the engineering viewpoint concept of *interception* to capture the idea of system components that allow bridges to be built between systems based on different infrastructures. Through the related concept of selective transparency, there was a similar decoupling between the computational (or programmatic) interfaces of the system — and the selection of engineering mechanisms that provided the necessary infrastructure to deliver the non-functional requirements set out in enterprise viewpoint policies. This was a contentious approach, standing against the philosophy at the time of seeking a universal set of standards to which all vendors and users would adhere. Recognizing that a model that could talk about interoperability between diverse systems run by separate authorities would be more general, the RM-ODP community embraced the ANSA federation and interception concepts and by that means ensured the model would remain relevant beyond the lifetime of many of the technologies that inspired it.

There were areas where ANSA was deficient and others in the RM-ODP community developed new ideas (several of which were pulled back into ANSA and ANSAware). Perhaps the most significant were concepts added to allow the modelling of other than RPC-based interaction. This need came from representatives of the telecommunications industry who wanted RM-ODP to be able to accommodate the signalling systems used to control switched networks and the need to carry synchronous streams of voice or video traffic as well as asynchronous data. To fill the gap J-B Stefani, at the time with France Telecom, introduced the concept of *signals* as synchronous atomic communication events from the Esterel [50] programming language into the RM-ODP computational viewpoint. This opened a broad vein of further development of rich multimedia systems based on RM-ODP and spawned a further industry consortium, TINA-C, which explored the application of RM-ODP ideas to

the design of *intelligent telecommunications networks* to support new kinds of telecommunications services being offered by the increasingly deregulated and privatized telecommunications industry.

Looking back now, some 20 years later, the obvious question is what impact RM-ODP had on subsequent developments. Certainly it provided ISO with a necessary tool-box for further work on distributed computing standards, although, with some exceptions, the momentum moved away from ISO to the industry consortia, with new ones springing up alongside OSF and OMG as distributed computing moved to new technologies such as *web services*. RM-ODP had a strong influence on the Object Management Group's CORBA and CORBA service standards, which then went on to strongly influence the distributed computing provisions of the Java programming language and virtual machine and, in turn, from there through to web services. Certainly the ODP work helped educate many in the computer and telecommunications industries, whether as supplier or customer, on how to exploit distributed computing during rapid evolution of networks from early LANs to today's global Internet.

What of the future? RM-ODP remains of significant value — the concepts are general and powerful enough to describe current systems. Moreover, as we come to grow ever more dependent on computers to run the modern world and need to manage the complexity of these systems and the rapid evolution of the technologies they use, the ability to model and specify them accurately and completely remains a key challenge. Looking further ahead, in the Microsoft Research laboratory where I work, some colleagues are using concepts from distributed computing to construct computational models of DNA replication, splitting, and recombination along with the systems biology of human cells driven by a vision of being able to program *theranostic* molecules to diagnose and treat genetically based conditions. Perhaps one day we will see humans modelled in the computational and engineering viewpoints of RM-ODP as well as the enterprise viewpoint where they mostly live today!

In terms of RM-ODP itself, the enterprise viewpoint has remained an active area of development through to the current day linked to work in formal methods for system design and specification used in approaches such as *Model-Driven Architecture* where the collaboration between the RM-ODP community and the OMG remains vigorous and productive.

I commend the authors for continuing with the RM-ODP agenda from the early days when we worked together and I welcome the opportunity to acknowledge the contributions of all those who have participated in RM-ODP. In this book the authors have done an excellent job in relating the concepts to modern systems and needs, retaining the rigour of the model found in the ISO documents but with a refreshing and informative style to make them more approachable.

Andrew Herbert
Cambridge

Preface

The Reference Model for Open Distributed Processing (the RM-ODP) is an international standard created by the standardization bodies ISO and ITU-T. It gives a solid basis for describing and building widely distributed systems and applications in a systematic way. Emphasis is placed on the need to build such systems with evolution in mind by identifying the concerns of major stakeholders and then expressing the design as a series of linked viewpoints representing these concerns. Each stakeholder can then develop an appropriate view of the system with a minimum of interference from the others.

Although ODP has been available as a standard for more than 10 years, standards are not easy bedtime reading. The ideas presented have an enthusiastic following, but, outside of it, many practitioners are still unaware of them. This book aims to provide a gentler pathway to the essential ideas that make up ODP and to show how they can be applied when designing and building real systems. It offers an accessible introduction to the design principles for software engineers and enterprise architects. In addition, it explains the benefits of using viewpoints to produce simpler and more flexible designs. It is not limited to any single tool or design method, but concentrates on the key choices that make an architectural design robust, flexible and long lived. The book also shows the power of enterprise architecture for the design and organization of complex distributed IT systems.

The book has been prompted by the recent revision of the standard, during which ISO has incorporated experience from the application of ODP to many different domains and has taken account of new technologies and fashions. We cover the most up to the minute developments in the ODP standards, including the recent updates to the ODP reference model and the ODP enterprise language. The book provides fresh insights into the design of enterprise systems. Another reason for producing this book is to mark the publication of the ISO/IEC 19793 standard (known as UML4ODP), which uses the Unified Modelling Language (UML) notation to provide a familiar and accessible way of expressing ODP designs; it does this by defining a standardized UML profile. The book also provides guidelines for using the UML notation for structuring and writing system specifications and for fitting such specifications into the Model-Driven Engineering toolchain. This gives users of the ODP ideas a simpler way of expressing them with existing design and modelling tools.

Finally, there is an ongoing interest in using ODP when addressing new standardization approaches and interoperability frameworks such as those in

e-government, e-health, and the energy and transportation industries. The book shows how the RM-ODP ideas can be applied to modern movements such as service engineering, cloud computing and the creation of the open enterprise.

This is the first book to introduce the ODP material in language that is common to software engineers and software architects. It offers a short, concise and focused presentation of the essentials of RM-ODP and shows where it fits within today's software processes. The book describes all the major concepts and mechanisms of the ODP framework, explains how to use them in a practical way for the specification of large open distributed systems, and presents the basic notation used for creating the specifications. It follows the standards faithfully, but provides extra information on the thinking behind them, and on how they should be interpreted. The reader can get acquainted with the best design concepts and practices, which are essential to anyone who designs large software applications professionally.

The Roadmap

The book is targeted at a number of different audiences. It has been written to be attractive both to the technical experts working on system architecture and to a much broader audience working on realizing such systems.

The book is divided into four parts, each having a different focus and each exploring progressively more detail of the different concepts and their use. There are also two appendices. The four parts provide

- An extended executive summary introducing the basic structuring ideas of ODP, particularly the central idea of there being a set of viewpoints.
- A more detailed explanation of the five ODP viewpoints and of the correspondences between them. Reading this part gives an idea of the style and use of the main elements of the ODP architecture.
- An explanation of the way these concepts are used to solve a number of the common problems met in the development and evolution of distributed systems. This part will help the reader to understand how use of the structure results in more flexible and adaptable systems.
- A discussion of some of the subtler ideas underlying this kind of system modelling and the new requirements they place on the supporting tools. This part answers some of the immediate questions people often want to ask about why the framework was defined in the way it is.

The first appendix brings together all the example model fragments used in the book to provide a single overview of a simple ODP-based design. Space

limitations mean that even this needs to be selective, and a fuller version is available from the authors' website (see `http://theodpbook.lcc.uma.es/`).

The second appendix presents some questions and scenarios that can be used to support teaching and training using this book. In addition to providing information for practicing professionals, we expect the book to provide a resource for graduate students and researchers who want to understand the main problems and principles involved in the design of large software systems; the book can also be used in Masters or Doctorate courses for teaching the concepts and design principles of ODP and for preparing students to research new problems in this area.

The whole structure is unified by the use of a single running example describing the IT support needed by a medium-sized company as it grows and develops. One of the problems in understanding a system's architecture is in seeing just how it helps the day-to-day activities of the system builders. Abstract structures in their purest form can seem dry and remote, so we have tried to relate them to the problems developers face by including a series of short vignettes illustrating why the various aspects of the architecture are needed. These fragments support, but do not form an essential part of, the main exposition. The individual chapters generally start with one of these fragments to give an informal introduction to the problem to be solved.

The book is targeted at three groups of readers. It is primarily intended for enterprise architects and software engineers who want to understand the concepts, mechanisms and problems involved in the design of complex enterprise systems and to use this knowledge in establishing a tool-based approach to documenting, evolving and testing systems. Readers will become more aware of the issues and options available for designing within a strong architectural framework. They can apply the ideas in general terms, or study the full detail further in the standards or in one of the reference books based on them.

A second target audience includes IT project managers and CIOs, who will be able to understand the possibilities of the RM-ODP framework and a viewpoint-based design approach, and the potential benefits that the adoption of this approach can bring to their companies and organizations. There are a growing number of large, multi-organizational information systems projects, for example in the aerospace and healthcare areas, and the designers involved are seeking urgently for a systematic architectural approach. ODP provides such an approach.

A third group, CEOs and business architects, can get an overall idea of this design approach and be able to evaluate the advantages of the use of a mature set of ISO and the ITU-T standards within their organizations, and also for interoperating with the IT systems of their customers, providers, financial services, and so on. The use of international standards is now essential to achieve (and to guarantee) the level of interoperability required in these large and complex IT systems with hundreds of customers, providers and developers, which need to exchange data and services with other IT systems in a seamless way.

Trademarks and Copyright

Acknowledgements

The ODP standards are the result of the efforts of a great many experts working within the standards process over many years. There is not room here to give credit to all of these people, but we should like, in particular, to acknowledge the contribution of the ISO conveners and document editors in leading the effort, particularly Joost J. van Griethuysen, Eng Chew, Bryan Wood, Jean Bérubé, Fausto Caneschi, Jean-Bernard Stefani, Andrew Herbert, Richard Sinnott, Mirion Bearman, Pramila Daryani, Peter Furniss, Lea Kutvonen, Laurent LeBoucher, Joaquin Miller, Kerry Raymond, Gerd Schürmann, and Sandy Tyndale-Biscoe. Arve Meisingset played a pivotal role in coordinating our link with the ITU-T, and Tom Rutt provided vital liaisons links with the OMG.

A number of people have read drafts of this book and provided us with very useful suggestions; in particular, our thanks go to Andy Bond, Fred Cummins, Andrew Herbert, Jishnu Mukerji, Bruno Traverson and Bryan Wood.

The authors would also like to thank No Magic, Inc. for making their UML tools available to facilitate the production of this book, and José Raúl

Romero and Juan Ignacio Jaen for building and maintaining the ODP plug-in for MagicDraw, and for maintaining the RM-ODP website.

Finally, we would like to thank all those involved at CRC Press, particularly Randi Cohen, Amber Donley and Karen Simon, for their expert help and advice throughout the publication process.

Peter F. Linington
Zoran Milosevic
Akira Tanaka
Antonio Vallecillo

Historical Background

The requirements for ODP and its reference model can be traced back a long way, from the earlier work on Open Systems Interconnection. It became clear to experts working on application protocols that the prevalent focus on peer-to-peer communications was not enough and that distributed systems design needed to take a more holistic approach, starting from the structure of the organizations involved. It was necessary to begin with a thorough understanding of the enterprise before proposing any technical solutions.

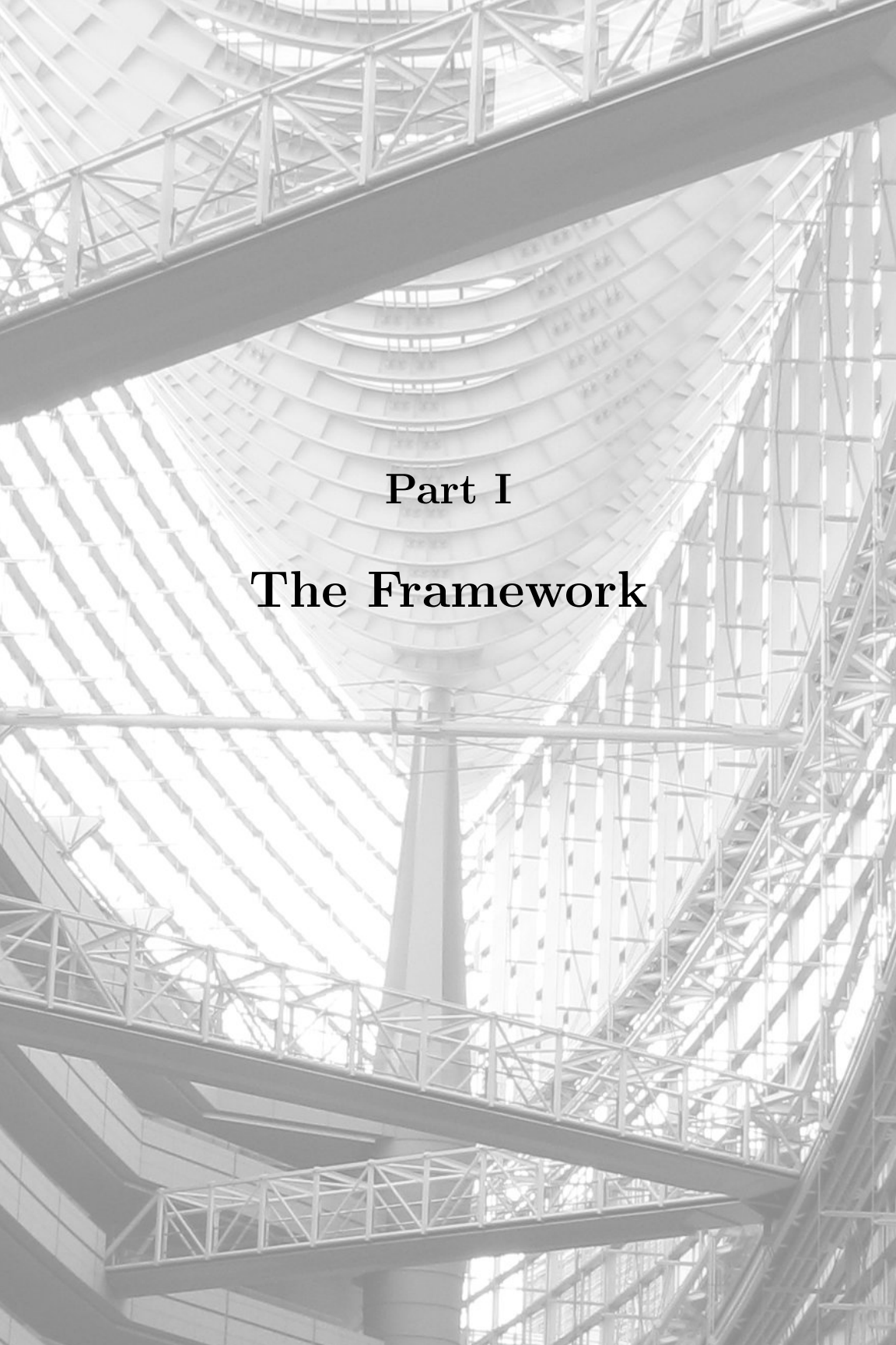
At the same time, there was considerable work in progress on the idea of middleware supporting a uniform distribution platform, leading, for example, to the ANSA architecture [72] and to early Object Management Group (OMG) specifications. The work on ODP started by harvesting the current research ideas available at that time and then used them to construct a vendor-neutral architecture. This principle of maintaining a broadly applicable framework by using the best current thinking has continued to be the basis for work on ODP.

Currently RM-ODP is being maintained and developed by an ISO/IEC Working Group (JTC1/SC7/WG19: *Techniques for the Specification of IT Systems*). This group works in close cooperation with ITU-T on the joint publication of a series of standards, and also maintains strong collaborative links with other international bodies, such as the OMG. These links help to provide the intelligence for continuously updating and improving the framework as new technologies and paradigms emerge, and for maintaining the consistency of a broad range of specifications and standards.

Conventions Used

Much of the explanation of architectural elements is about the way different concepts are represented and what artefacts might result from their application. It is easy to lose track of which use falls into which category, so we introduce some graphical conventions to help sort things out. Throughout the book, we will use the following typographical conventions:

- A ***bold italic*** font highlights a word that is an ODP concept.
- A sans serif font indicates that an element is from a UML model.
- A `typewriter` font flags an item as being a concrete instance.



Part I

The Framework