This is a chapter from the book

System Design, Modeling, and Simulation using Ptolemy II

First Edition, Version 1.0

Please cite this book as:

Claudius Ptolemaeus, Editor,
System Design, Modeling, and Simulation using Ptolemy II, Ptolemy.org, 2014.
http://ptolemy.org/books/Systems.

Preface

My last written work was published nearly 1,900 years ago. I am pleased to come out of retirement to give voice to a project that I'm proud to have named after me, the Ptolemy Project. Like much of my prior work in astronomy and geography, this project deals with complex systems. Like most of my prior writings, this text compiles the thinking and contributions of many people.

The motions of the planets, the sun, the earth, and the moon, which I studied in my work *The Almagest*, are concurrent interacting processes. They are *deterministic*, not subject to the whims of the gods. More accurately, the models that I developed, as well as those of many of my successors, deliberately ignore any effects that the gods might capriciously impose. These models focus instead on precisely matching observed behavior, and more importantly, on predicting behavior. The Ptolemy Project similarly studies concurrent processes and focuses on deterministic models.

Ideally, an intellectual quest moves human knowledge from superstition and unfounded beliefs to logic and measurement. What we now call "science," particularly in the study of natural systems, is deeply rooted in the scientific method, where we form hypotheses, design experiments, and draw conclusions about the hypotheses based on the experiments. To be able to make measurements, of course, the artifact or process being measured must exist in some form. In my earlier studies, this was not an issue, since the sun, earth, moon, and planets already existed. Engineering disciplines, which focus on human-constructed

artifacts and processes, however, study systems *that do not yet exist*. Nevertheless, the scientific method can and is applied in engineering design. Engineers construct simulations and prototypes of systems, formulate hypotheses, and perform experiments to test those hypotheses.

Because of the focus on artifacts and processes that do not yet exist, engineering design should not be based solely on the scientific method. The goal of experiments is to improve our understanding of the artifact or process being designed. But we have to create the artifact or process before we can perform experiments. Being forced to create something before we understand it dooms our design to roots in superstition and unfounded beliefs.

An important part of a science, quite complementary to the scientific method, is the construction of models. Models are abstractions of the physical reality, and the ability of a model to lend insight and predict behavior may form the centerpiece of a hypothesis that is to be validated (or invalidated) by experiment. The construction of models is itself more an engineering discipline than a science. It is not, fundamentally, the study of a system that preexists in nature; it is instead the human-driven construction of an artifact that did not previously exist. A model itself must be engineered.

Good models can even reduce the need for measurement, and therefore reduce the dependence on the scientific method. Once we have a model of the motions of the planets, for example, that we know accurately predicts their positions, there is less need to measure their positions. The role of measurements changes from determining the positions of the planets to improving the models of their motions and to detecting capricious actions of the gods (something that engineers call "fault detection").

In both science and engineering, models can be iteratively improved. My own geocentric model of the universe required much iterative refinement to bring it into close conformance with experimental observation of the motions of the planets. I am quite proud of the predictive power of the resulting models, and also of the way predictions based on these models could be mechanized by an *astrolabe*. I nevertheless acknowledge that my esteemed colleague Nicolaus Copernicus provides a better model for the motion of the planets. His model is conceptually simpler, once you make the leap that the center of our *observable* universe, the ground we stand on, need not be the center of a *model* of the universe. Indeed, we have much more freedom with models than with the physical world, as models need not be constrained by nature. Nevertheless, my models were the best ones available for nearly 1400 years.

The Ptolemy Project is indeed a study of models of systems. The systems, however, are quite different from the ones I focused on in the past. Those were given to me by nature, but the ones in this book are created by humans. In this book, the purpose of modeling is to improve systems, and there is nothing I could have done to improve the planetary system given to us by nature.

In short, in engineering, as opposed to science, models play a role in the *design* of the systems being modeled. As with science, models can be improved, but unlike science, so can the systems being modeled.

Even more interestingly, in engineering, unlike science, the choice of models affects the systems being modeled. Two engineers, given the same goals, may come up with radically different designs and realizations of a system simply because they started with radically different *models* of the system. Moreover, these two engineers may have come up with different models simply because they started with different tools for constructing models. An engineer who constructs models with pencil and paper will likely come up with very different models than those of an engineer who starts with a software modeling tool. As a consequence, they will likely come up with very different system designs.

This text collects an astonishingly rich variety of modeling tools and techniques for complex systems. Some of these will undoubtedly be improved upon in the future, like my own epicycles, a modeling complexity that was rendered unnecessary by Copernicus. Nevertheless, the goal of this book is to offer to engineers the best modeling techniques available today, with the hope that this will lead to the best system designs achievable today. Tomorrow, we can be sure we will be able to do still better.

How to Use This Book

The book is intended for use by engineers and scientists who need to model a wide variety of systems or who want to understand how to model complex, heterogeneous systems. Such systems include, for example, mechanical systems, electrical systems, control systems, biological systems, and – most interestingly – heterogeneous systems that combine elements from these (and other) domains. The book assumes a familiarity with simulation and modeling tools and techniques, but does not require in-depth background in the subject matter.

The book emphasizes modeling techniques that have been realized in Ptolemy II. Ptolemy II is an open-source simulation and modeling tool intended for experimenting with system design techniques, particularly those that involve combinations of different types of models. It was developed by researchers at UC Berkeley, and over the last two decades it has evolved into a complex and sophisticated tool used by researchers around the world. This book uses Ptolemy II as the basis for a broad discussion of system design, modeling, and simulation techniques for hierarchical, heterogeneous systems. But it also uses Ptolemy II to ensure that the discussions are not abstract and theoretical. All of the techniques are backed by a well-designed and well-tested software realization. More detailed descriptions of Ptolemy's underlying software architecture and the finer points of its operation and underlying theory are found in sidebars, references, and links.

The electronic version of the book provides a number of links to material both within the book and on external websites. Furthermore, most of the illustrations showing models provide a hyperlink that enables you to view, download, edit, and run the model shown in the figure. Such figures are indicated in caption with a link reading "[online]" that will first take you to a web page that enables browsing the model and downloading it to run on your own Ptolemy II installation (available from `http://ptolemy.org`). This web page also provides a link to a **Java Web Start** deployment of Ptolemy II, which circumvents the steps of downloading and installing the software. The Java Web Start deployment, however, will not work if your browser does not have Java enabled or if you are reading on a machine that does not have Java capability (such as an iPad). For the print version of the book, to get to the models marked with "online," go to `http://ptolemy.org/systems`.

The book is organized in three parts. First is an introduction. The first chapter outlines the principles behind the styles of modeling that are covered in this book, and it gives a tour of the multiple models of computation (MoCs) that are described. The second chapter is a

tutorial on using Ptolemy II via its graphical editor, Vergil. This chapter is a good starting point for the impatient reader who wants to just get directly into building models.

The second part, Chapters 3 through 11, covers models of computation. Each chapter covers one or a small family of related models of computation, explaining how they work, how to build models using them, and what sorts of models are well matched to the MoCs.

The third part covers capabilities of Ptolemy II that span models of computation. Chapter 12, perhaps the most important one for readers who are interested in extending Ptolemy II or in writing their own actors in Java, describes the software architecture. Ptolemy is open-source software, with well-documented code that is meant to be read, and this chapter serves as a good orientation for readers who would like to look at the code and build on it. Chapter 13 describes the expression language that is used for specifying parameter values for models and for giving custom functions to actors. Chapter 17 describes the capabilities of the signal plotters that are included in the Ptolemy II standard library. Chapter 14 describes the type system in Ptolemy II; it is a sophisticated type system designed to minimize the burden on the model builder (by emphasizing type inference rather than type declarations) while maximizing safety by providing a strong type system. Chapter 15 describes ontologies, which extend the type system with an ability to associate units, dimensions, and concepts with data values in models. Again, the emphasis is on inference and safety. Finally, Chapter 16 describes the web interfaces included with Ptolemy II. Specifically, it explains the capabilities for exporting web pages from models and for building web servers and services into models.

At the end of the book is an extensive bibliography (where most entries provide hyperlinks to the documents that are cited), a comprehensive actor index (listing all the actors in the Ptolemy II standard library), and an extensive index with more than 4,000 entries.

Acknowledgements

The modeling techniques described in this book have been developed in the Ptolemy Project at the University of California at Berkeley over many years. The roots of the project date back to the 1980s. The first germ came from Messerschmitt (1984), who built a software framework called **Blosim** (for Block Simulator) for simulating signal processing systems. Messerschmitt's PhD student, Edward A. Lee, inspired by Blosim, developed the synchronous dataflow (SDF) model of computation (Lee, 1986; Lee and Messerschmitt, 1987b) and a scheduling theory for it (Lee and Messerschmitt, 1987a). Lee and his students then developed the Lisp-based software tool called **Gabriel** (Lee et al., 1989), with which they developed and refined the SDF model of computation. In the early 1990s, Lee and Messerschmitt set out to develop an object-oriented block diagram framework called **Ptolemy** (Buck et al., 1994) (now called **Ptolemy Classic**). Later in the 1990s, Lee and his group started a complete redesign called **Ptolemy II** based on the brand-new programming language Java (Eker et al., 2003). Ptolemy II developed the key ideas of actor-oriented design (Lee et al., 2003) and hierarchical heterogeneity.

The trajectory of the software in many ways reflects the evolution of computing during this time period. Blosim was written in C. Gabriel was written in Lisp. The first Ptolemy system, now called Ptolemy Classic, was written in C++. The next version, Ptolemy II, was written in Java. Each of these transitions reflected an effort to leverage the best available technology to solve real design problems. Lisp made sense over C because of its robustness and suitability for specification of complex decision logic. C++ made sense over Lisp because of its much better developed (at the time) notions of object-oriented design (particularly inheritance and polymorphism). Java made sense over C++ because of its portable support for multithreading and user interfaces. And, perhaps most importantly, switching languages periodically made sense because it forced the team to redesign the system, leveraging lessons learned.

This text is based heavily on the Java-based Ptolemy II. Nevertheless, a great deal of credit goes to the designers of Ptolemy Classic (Buck et al., 1994), most particularly Joseph Buck, Soonhoi Ha, Edward A. Lee, and David Messerschmitt.

People

Many people have contributed to this book and to the Ptolemy II software used to illustrate the concepts in the book. In addition to the authors of individual chapters, people who had major influence on the book include Shuvra S. Bhattacharyya, David Broman, Adam

Cataldo, Chihhong Patrick Cheng, Daniel Lazaro Cuadrado, Johan Eker, Brian L. Evans, Teale Fristoe, Chamberlain Fong, Shanna-Shaye Forbes, Edwin E. Goei, Jeff Jensen, Bart Kienhuis, Rowland R. Johnson, Soonhoi Ha, Asawaree Kalavade, Phil Lapsley, Bilung Lee, Seungjun Lee, Isaac Liu, Eleftherios D. Matsikoudis, Praveen K. Murthy, Hiren Patel, José Luis Pino, Jan Reineke, Sonia Sachs, Farhana Sheikh, Sun-Inn Shih, Gilbert C. Sih, Sean Simmons, S. Sriram, Richard S. Stevens, Juergen Teich, Neil E. Turner, Jeffrey C. Tsay, Brian K. Vogel, Kennard D. White, Martin Wiggers, Michael C. Williamson, Michael Wirthlin, Zoltan Kemenczy, Ye (Rachel) Zhou, and Jia Zou. Christopher Brooks has had overall responsibility for the software, and more than anyone is responsible for its high quality. Other contributors are given at `http://ptolemy.org/people`.

We also extend particular thanks to Jennifer White, who provided extensive editorial help, improving the the writing and clarity throughout the book. Others providing significant editorial help include Yishai Feldman, William Lucas, Aviral Shrivastava, Ben Zhang, and Michael Zimmer.

Funding

The work described in this text has received funding from a variety of sources over many years. In particular, the National Science Foundation (NSF) supported much of the work described in this book under the following projects:

- CCR-0225610: ITR: *Foundations of Hybrid and Embedded Systems and Software*
- CNS-0931843: Large: CPS: *ActionWebs*
- CNS-1035672: CPS: Medium: *Timing-Centric Software*
- CNS-0647591: CSR-SGER: Cyber-Physical Systems — *Are Computing Foundations Adequate?*

In addition, key pieces were sponsored by:

- The Army Research Laboratory (ARL), Cooperative Agreement Number W911NF-11-2-0038: *Disciplined Design of Systems of Systems*, as part of the Office of Secretary of Defense (OSD) Software-Intensive Systems Producibility Initiative.
- The Army Research Laboratory (ARL), Cooperative Agreement Number W911NF-07-2-0019: *Scalable Composition of Subsystems*, as part of the Office of Secretary of Defense (OSD) Software-Intensive Systems Producibility Initiative.

- The Naval Research Labs (NRL), Cooperative Agreement Number N00173-12-1-G015: *Software Producibility for Systems of Systems*, as part of the Office of Secretary of Defense (OSD) Software-Intensive Systems Producibility Initiative.
- The Air Force Research Lab (AFRL), Contract FA8750-11-C-0023: *Extensible Modeling and Analysis Framework*.
- The Air Force Office of Scientific Research (AFOSR) under the Multi-University Research Initiative (MURI) grant #FA9550-06-0312: *High-Confidence Design of Adaptive, Distributed Embedded Control Systems (HCDDES)*.

Additional funding was provided by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which over several years has received support from the following companies:

- Agilent
- Bosch
- Lockheed-Martin
- National Instruments
- Thales
- Toyota

Support for work on modeling of heterogeneous systems was also provided by the iCyPhy (industrial cyber-physical systems) research consortium, sponsored by IBM and United Technologies.

Key support for work described in Chapters 10 and 16 was provided by the Focus Center Research Program (FCRP), a Semiconductor Research Corporation (SRC) program funded by MARCO and DARPA, under the following projects:

- The *Multiscale Systems Center* (MuSyC).
- The *TerraSwarm Research Center*, one of six centers supported by the STARnet phase of the FCRP.

The views and conclusions contained in this text are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any of the sponsors. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

Part I

Getting Starting

This part of this book introduces system design, modeling, and simulation. First, Chapter 1 outlines the guiding principles for disciplined modeling of heterogeneous systems. It gives a high-level overview of the models of computation that are described in detail in Part II. It also provides a highly simplified case study (an electric power system) that illustrates the roles that various models of computation play in complex system design.

Chapter 2 provides a tutorial on using Ptolemy II through its graphical user interface, Vergil. One of the goals of this book is to enable the reader to exploit the open-source Ptolemy II system to conduct experiments in system design. This chapter is intended to provide enough information to make the reader a competent user of Ptolemy II. To extend Ptolemy II, the reader is referred to Part III.