Chapter 6, *Data Grows Up: The Architecture of the Facebook Platform*, by Dave Fetterman, advocates data-centric systems, explaining how a good architecture can create and support an application ecosystem.

Part III: Systems Architecture

Systems software is arguably the most demanding type of software to design, partly because efficient use of hardware is a black art mastered by a selected few, and partly because many consider systems software as infrastructure that is "simply there." Seldom are great systems architectures designed on a blank sheet; most systems that we use today are based on ideas first conceived in the 1960s. The chapters in Part III walk you through four innovative systems software architectures, discussing the complexities behind the architectural decisions that made them beautiful.

Chapter 7, *Xen and the Beauty of Virtualization*, by Derek Murray and Keir Fraser, gives an example of how a well-thought-out architecture can change the way operating systems evolve.

Chapter 8, *Guardian: A Fault-Tolerant Operating System Environment*, by Greg Lehey, presents a retrospective on the architectural choices and building blocks (both software and hardware) that made Tandem the platform of choice in high-availability environments for nearly two decades.

Chapter 9, *JPC: An x86 PC Emulator in Pure Java*, by Rhys Newman and Christopher Dennis, describes how carefully designed software and a good understanding of domain requirements can overcome the perceived deficiencies of a programming system.

Chapter 10, *The Strength of Metacircular Virtual Machines: Jikes RVM*, by Ian Rogers and Dave Grove, walks us through the architectural choices required for creating a self-optimizable, self-hosting runtime for a high-level language.

Part IV: End-User Application Architectures

End-user applications are those that we interact with in our everyday computing lives, and the software that our CPUs burn the most cycles to execute. This kind of software normally does not need to carefully manage resources or serve large transaction volumes. However, it does need to be usable, secure, customizable, and extensible. These properties can lead to popularity and widespread use and, in the case of free and open source software, to an army of volunteers willing to improve it. In Part IV, the authors dissect the architectures and the community processes required to evolve two very popular desktop software packages.

Chapter 11, *GNU Emacs: Creeping Featurism Is a Strength*, by Jim Blandy, explains how a set of very simple components and an extension language can turn the humble text editor into ~~an operating system~~[*] the Swiss army knife of a programmer's toolchest.

[*] As some die-hard users say, "Emacs is my operating system; Linux just provides the device drivers."