in C or C++ may run Java applications. If the runtime were to have a bug relating to memory safety, it could crash the Java application, even though the Java application itself has memory safety. Removing bugs is an important reason to have a self-hosting runtime.

As computer systems are better understood and evolve, the requirements of a programming language change. For example, the programming languages of C and C++ have no standard library for utilizing multiple processors using threading (although popular extensions such as POSIX threads and OpenMP do exist). Modern languages will have such features designed into the language and standard library. Allowing the runtime to take advantage of better libraries and abstractions is another important reason to have a self-hosting runtime.

Finally, whenever a runtime and the application it is running must communicate with each other, there is a layer for the communication. One job of this communication layer can be to marshal objects, changing the format in one programming language to that of the other. In the case of objects, the communication layer also needs to remember not to garbage collect any objects that may be in use from outside of the managed runtime. Such communication layers are not necessary, or at least not necessary in as many situations, when the runtime is self-hosting.

We hope we've provided compelling reasons for making a self-hosting runtime. In this chapter, we present an overview of such a runtime, Jikes RVM, which is written in Java and runs Java applications. Self-hosting runtime environments are known as *metacircular* (Abelson et al. 1985). Jikes RVM is not unique in being metacircular, and indeed it draws inspiration from similar runtime systems, such as Lisp (McCarthy et al. 1962) and the Squeak virtual machine for Smalltalk (Ingalls et al. 1997). Being a metacircular virtual machine written in Java allows the use of excellent tools, development environments, and libraries. As Java lacks credibility in certain communities, we will first address some of the myths that may lead people to believe that a metacircular Java runtime has inherent flaws.

Myths Surrounding Runtime Environments

There is still much active debate on how best to create applications for different environments. Factors such as the resources available where the application will be run, the productivity of the developers challenged with creating the application, and the maturity of the development environment come into play. If the application is implemented in the same way, performance and memory requirements are a feature of the development environment. Next, we look to overturn the most common myths about managed environments.

As Runtime Compilers Must Be Fast, They Must Be Simple

A misconception about runtime environments is that they are interested purely in *just-in-time* (JIT) compilation. JIT compilation must create code quickly because it will be put to use as soon as it is ready. Although this simple execution model was used in many early JVMs and