the KDE developer community itself, they are also used by many third-party developers, both commercial and noncommercial, to produce thousands of additional applications and components.

Although the initial focus of the KDE project was to provide an integrated desktop environment for free Unix operating systems, notably GNU/Linux, the scope of KDE has broadened considerably, and much of its software is now available on not just various flavors of Unix, but also on Microsoft Windows and Mac OS X, as well as embedded platforms. This implies that the code written for the KDE libraries has to work with many different tool chains, cope with the various platform peculiarities, integrate with system services flexibly and in extensible ways, and make judicious and careful use of hardware resources. The broad target audience of the libraries also means that they have to provide an API that is understandable, usable, and adaptable by programmers with diverse backgrounds. Someone accustomed to dealing with Microsoft technologies on Windows will have different preconceptions, biases, habits, and tools from an embedded programmer with a Java background or an experienced Mac developer. The goal is to make all programmers able to work comfortably and productively, to allow them to solve the problems at hand, but also (and some say more importantly) to benefit from their contributions should they choose to give back their suggestions, improvements, and extensions.

This is a very agile, very diverse, and very competitive ecosystem, one in which most active contributors are interested in collaborating to improve their software and their skills by constantly reviewing each other's work. Opinions are freely given, and debates can become quite heated. There are always better ways to do one thing or another, and the code is under constant scrutiny by impressively smart people. Computer science students analyze implementations in college classes. Companies hunt down bugs and publish security advisories. New contributors try to demonstrate their skills by improving existing pieces of code. Hardware manufacturers take the desktop and squeeze it onto a mobile phone. People feel passionately about what they do and how they think things should be done, and one is, in many ways, reminded of the proverbial bazaar.† Yet this wild and unruly bunch is faced with many of the same challenges that those in more traditional organizations who are tasked with maintaining a large number of libraries and applications must overcome.

Some of these challenges are technical. Software has to deal with ever-increasing amounts of data, and that data becomes more complex, as do the workflows individuals and organizations require. The necessity to interoperate with other software (Free and proprietary) used in corporations or government administrations means that industry paradigm shifts, such as the move towards service-oriented architecture (SOA), have to be accommodated. Government and corporate mission-critical uses pose stringent security requirements, and large deployments need good automation. Novice users, children, or the elderly have different needs

† *http://en.wikipedia.org/wiki/The_Cathedral_and_the_Bazaar*