USENIX Association

Proceedings of the XFree86 Technical Conference

Oakland, California, USA
November 8–9, 2001

KDE: Interface, Standards, and Inter-Process Communication

Ellis Whitehead
KDE Project Team
kde@ellisw.net

Abstract

KDE is one of the leading *nix desktop environments, and new developments are integrated into the system at a steady and rapid rate. Many recent developments are apparent in the user interface, but much work has also been done which is transparent to the end-user. It is this behind-the-scenes functionality which this talk investigates, specifically looking at Interoperability with other desktop environments, Component reuse, and Interprocess Communication (IPC).

1 Interoperability

The KDE and Gnome projects, among others, have adopted the Desktop Entry Standard and the NET Window Manager Specification, allowing an application written for one environment to be properly launched and managed in another. The Desktop Entry Standard defines a configuration file format for describing such details as how a particular program should be launched and how it appears in menus. The Window Manager Specification defines interactions between window managers, applications, and the utilities that form a part of a modern desktop environment. The convention allows developers to write custom extensions (such as floating toolbars) that have a consistent look-and-feel in different desktop environments, but it is also flexible enough to avoid hampering creativity.

2 Components and Interprocess Communication

Components are supported in KDE via KParts, and they are used extensively through the entire system. Indeed, much of the power, flexibility, and extensibility of KDE can be attributed to its component-based design. Any KParts component can be seamlessly and easily plugged into any KDE application, giving the application charting and browsing capabilities, for example, with minimal coding. The plug-ability of KParts components into individual applications is complemented by the simplicity and efficiency of communication among distinct KDE applications. This is accomplished via KDE's IPC mechanism called the Desktop Communications Protocol (DCOP), which also serves as a scripting mechanism for controlling GUI programs from the command line or another program. Furthermore, communication with remote applications is made possible by XML-RPC and SOAP bridges.

We will look at these standards and protocols in action, and additionally at the lessons learned in the process of developing and implementing them. In a project as large as KDE, design decisions receive thorough testing over time for their strengths, weaknesses, and potentials. The results, both positive and negative, should also be of interest for other projects concerned with interoperability, code reuse, and data exchange.

3 Further Information

Further information for this talk regarding Components, KParts, IPC/RPC, DCOP, ICE, XML-RPC, and SOAP, is available at `http://www.kde.org/kdeslides/XFree2001`.