Computational Grids are persistent environments established by combining diverse high-performance computing systems and specialized peripheral hardware. The ultimate goal of grid initiatives is to provide ubiquitous access to networked resources comparable to the omnipresence of the electrical power grid. Access to the grid must be offered via a transparent, unified view across individual systems, organizations and even international borders.

In order to exploit the processing power provided by grids, software applications must be adapted and tailored to its characteristics and concepts. Since grid computing is in many aspects a generalization of technologies that were originally developed in the context of high-performance computing and networking, it is an essential fact to employ parallel and distributed computer systems and techniques.

Consequently, grid computing affects every aspect of program development, which in turn must be addressed by parallel and distributed programming tools and environments. This includes tools for monitoring, testing and debugging, performance analysis and tuning, load-balancing and scheduling, and software visualization. Some examples grid software development tools are presented below. The contribution “Managing Distributed Resources in the SVG Project” by J.R. Luaces et al describes resource management activities in the context of the Superordenador Virtual Galego (SVG) system, which represents a grid computing infrastructure established on top of the Galician communications network. The goal of the SVG project is to allow high-throughput computing within the given resource limitations of a standard network and without compromising the local usage of shared machines. This is achieved with a scheduling scheme, that allows job distribution and job migration according to the availability of resources.

In contrast, the paper “Scalable Monitoring and Configuration Tools for Grids and Clusters” by P. Augerat et al targets the area of collecting, visualizing, and analyzing grid information for resource monitoring and tuning in the Ka-admin project. The main concept of Ka-admin is a generic filter module, which can be adapted to the users need for removing, aggregating, and transporting data from grid monitoring tools to corresponding visualization tools. With this approach, scalable visualization of large data sets as processed in grid applications can be achieved.

The area addressed by the article “Modular MPI Components and the Composition of Grid Applications” by Y. Cotronis is design and implementation of message passing algorithms for grids. Based on the Ensemble methodology, the author outlines the composition of modular message passing components for demanding irregular or partially regular process topologies, and demonstrates programming with Ensemble with two simple examples.

The paper “DeWiz - Event-based Debugging on the Grid” by D. Kranzlmüller deals with error detection in grid applications. The main idea of the DeWiz debugging plugin is to describe program behavior with the event graph model, and to apply this model for all possible debugging activities.

Comparable to the paper by P. Augerat et al, this approach allows to cope with large-scale grid applications by defining filtering and abstraction rules for distinct analysis modules. In addition, DeWiz itself uses the grid to perform automatic analysis and error detection activities.

All contributions offer some insights into different aspects of program development for computational grids. This identifies several problems of grid program development and application, and indicates some of the open topics for future research projects.