The priorities are virtual properties of the job objects and are set there. It is important to keep in mind that all these objects are set to not process until they are queued, and in this case, until the processing is explicitly resumed. So the whole operation to create all these sequences and jobs really takes only a very short time, and the program returns to the user immediately, for all practical matters. The view updates as soon as a preview image is available.

From Concurrency to Scheduling: How to Implement Expected Behavior Systematically

The previous examples have shown how analyzing the problem completely really helps to solve it (I hope this does not come as a surprise). To make sure concurrency is used to write better programs, it is not enough to provide a tool to move stuff to threads. The difference is scheduling: to be able to tell the program what operations have to be performed and in what order. The approach is remotely reminiscent of PROLOG programming lessons, and sometimes requires a similar way of thinking. Once the minds involved are sufficiently assimilated, the results can be very rewarding.

One design decision of the central `Weaver` class has not been discussed yet. There are two very disjunct groups of users of the `Weaver` classes API. The internal `Thread` objects access it to retrieve their jobs to process, whereas programmers use it to manage their parallel operations. To make sure the public API is minimal, a combination of decorator and facade has been applied that limits the publicly exposed API to the functions that are intended to be used by application programmers. Further decoupling of the internal implementation and the API has been achieved by using the PIMPL idiom, which is generally applied to all KDE APIs.

A Crazy Idea

It has been mentioned earlier that at the time ThreadWeaver started to be developed, it was not really possible to implement all its ideas. One major obstacle was, in fact, a prohibitive one: the use of advanced implicit sharing features, which included reference counting, in the Qt library. Since this implicit sharing was not thread-safe, the passing of every simple Plain Old Data object (POD) was a synchronization point. The author assumed this to be impractical for users and therefore recommended against using the prototype developed with Qt 3 for any production environments. The developers of the KDEPIM suite (the same people who now develop Akonadi) thought they really knew better and immediately imported a preliminary ThreadWeaver version into KMail, where it is used to this day. Having run into many of the problems ThreadWeaver promised to solve, the KMail developers eagerly embraced it, willing to live with the shortcomings pointed out by its author, even against his express wishes.