Required Features

In short, an RSS client downloads news *articles* and displays them. The articles are organized into *feeds*. A feed is identified with a URL, and, at any one time, accessing that URL returns a file containing the articles. Each article typically has a title, a summary, and a link to a web page for further information. The feed file is formatted in XML, using one of several RSS standard formats. (To see an example of a feed, look at http://www.slashdot.org/index.rsswhich is an RSS 1.0 feed for http://www.slashdot.org/.) You can also try Google's online RSS reader for free.

As new articles enter the feed, older articles are typically dropped out. Different sources update their feeds at different rates; a client usually polls the feed to determine whether it has been updated. RSS clients therefore have elements in common with browsers, newsgroup readers, and email and instant messaging clients.

This project is very open ended; you are free to develop your RSS client with your own choice of features. Nevertheless, a base set of features is required:

- **Display of articles**. The user can display the titles and summaries of articles in a feed, as well as the full articles (which will in general require rendering webpages). A visual indication shows which articles have been read, and users have the option of only displaying articles that have not yet been read. The client must be capable of reading feeds in the 3 most common formats (RSS 1.0, RSS 2.0 and Atom 0.3).
- **Subscription to feeds**. The user can enter feed addresses, associate informal names with them, and organize them into groups (usually called *channels*). The user can select the update period for individual feeds, which determines how often the client polls them. The collection of feeds, their names, organization into channels, and update periods can be saved as a file, or loaded from a file. In addition, the list of feeds alone (without organization into channels or user preferences) can be saved and loaded in a standard interchange format known as OPML (see below), so that users can share subscriptions.
- **Caching of articles**. When the machine on which the client is running is offline, the user can access articles that have been previously downloaded. It is not necessary to make all articles that have been downloaded in the past accessible; you can invent some reasonable scheme for retiring old articles, ideally under user control.
- **Basic keyword search**. The user can search amongst previously downloaded articles by keywords appearing in titles.
- **Support for the amendment**. When the amendment is released (see below), it will amend the requirements listed here, so your final project must comply with the requirements of the amendment, as well.