Concept for a PIM Daemon

Tobias Koenig

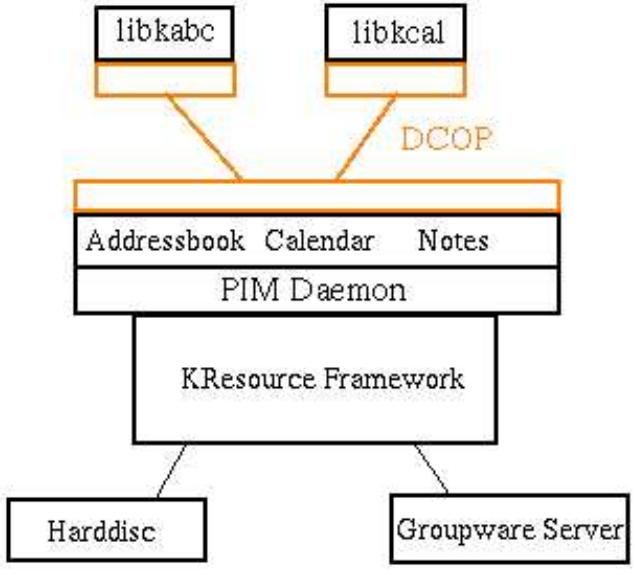
Why a PIM daemon

- every application loads own instance of pim data (addressbook, calendar)
- change notification only works when pim data are saved on disk
- locking is not available for all resources
- ...

Basics

- implemented as DCOP service (KDED module?)
- provides interfaces for accessing
  - addressbook
  - calendar
  - notes
  - emails (?)
- kabc/kcal become wrapper libs around DCOP service

The daemon


libkabc
libkcal
DCOP
Addressbook Calendar Notes
PIM Daemon
KResource Framework
Harddisc
Groupware Server

Tasks of the daemon

- keep all data in memory (loaded/saved by kresources)
- handling resources (add/edit/remove)
- locking of resources/contacts/events/todos etc.
- search methods (e.g. all contacts starting with ’A’; all events between today and tomorrow)
- data transfer
- change notification

Tasks of the wrapper libs

- hide the DCOP communication
- provide easy to use API
- offer components for common tasks (e.g. search dialogs)

TODO: KResource Framework

- cleanup API
- make everything asynchronous
- let it handle large amount of data
- support of subresources (?)

TODO: PIM daemon

- writing prototype (in osnabrueck?)
- define DCOP interfaces
- implement interfaces
- implement locking/notification code

TODO: wrapper libs

- cleanup APIs
- make libkcal value based (necessary for DCOP)
- move all functionality code to the server and replace it by DCOP calls
- add components (different kinds of search/selection widgets)

Let’s start a discussion...