Programming in ParaVision

Introduction

4.1

ParaVision has several possibilities to adapt and extend its functionality:

- setup and control of experiments (methods, pulse programs, Automations)
- setup of Subject / Study / Scan (macros, e.g. queued acquisition tool)
- control of the acquisition (macros, e.g. Localized_Spectroscopy_Guide)
- postprocessing of datasets (ISA Tool, Image Algebra, Filter Toolbox, macros)

ParaVision Infrastructure

4.2

ParaVision consists of different programs that have a common infrastructure shown in Figure 4.1:

Graphical User Interface (GUI)

4.2.1

The GUI is the interface to the user. The user performs his task using the GUI with a keyboard and a mouse. Nearly all tasks the user wants to perform are translated into commands that are sent to the command dispatcher.
**Programming in ParaVision**

**Command Dispatcher**

A command consists of an command identifier and optional / required arguments. Commands must be translated (dispatched) into operations that the user wants to perform. This is the task of the command dispatcher calling the functional components of ParaVision.

**Functional Components**

Functional components contain the implementation of all ParaVision operations. They communicate with the command dispatcher and the GUI. In this model the user can extend the functionality of ParaVision (Figure 4.2) using

- Macros - macros are programs/scripts sending ParaVision commands to the command dispatcher.
- Plugins - plugins are shared libraries, programs written by the advanced user (e.g. methods, pulse programs, etc.).

**Figure 4.2:** Plugin infrastructure of ParaVision
**Programming in ParaVision**

**Experiment setup and control**

**4.3**

**Pulse programs** determine the timing and sequence of NMR experiments. They are written in a special pulse programming language containing several simple commands. Minimal programming skills are required to learn its syntax.

Writing pulse programs and setting the acquisition parameters directly is referred to as "base level programming". Pulse programming is described in "Pulse Programming" in chapter D-5.

**NMR-SIM** is a tool for viewing and programming pulse programs which is able to show interactively the programmed pulse program in a graphical view.

**Automations** are a possibility to program experiments on a lower level. They allow easy and rapid parameter access and changing (e.g. large parameter array redimensioning). They are written in C-language, and are, therefore, convenient for applications involving complicated maths. They can also be used for postprocessing purposes. Programming of Automations is described in chapter D-11.

A unique feature of ParaVision is the possibility of inserting a user-programmed filter to the acquisition pipeline. **Pipeline Filters** can process the raw data on their way from the spectrometer to the computer. Pipeline Filters are special types of Automations and are described also in chapter D-11.

Programming of experiments at a "high level" is possible using **Methods**. **Methods**, which are also programmed in a derivation of the C-language, can calculate base level parameters from the input of the graphical user interfaces of ParaVision, e.g. from the **Geometry Editor**, as well as from the Protocols. They allow definition of new parameters. An experiment should be programmed as a Method rather than at the base level or using Automations when it is planned to run in routine mode, with the support of user interfaces, Protocols, and adjustments. A method programming tutorial is provided in chapter D-8.

**4.4**

**Macros**

Repetitive operations and batch execution of several ParaVision commands can be implemented using macros. Normally, they are shell scripts in an arbitrary scripting language (e.g. Bourne Shell, TCL/TK, PERL, etc.).

The shell scripting language must be able to start arbitrary programs and to retrieve their exit code as well as their standard and error output. Commands can be sent to a ParaVision program using the utility program pvcmd.
**Programming in ParaVision**

Examples for typical operations performed in macros are:

- setting and getting of ParaVision parameters,
- start / stop and control of acquisition,
- loop through all frames of a dataset, perform postprocessing operations,
- adapt the display of frames.

Macro programming rules and the interface of `pvcmd` are described in chapter D-10.

**Postprocessing plugins**

The following postprocessing plugins exist:

- **Image Algebra** *(Image Display & Processing)*
  The Image Algebra performs algebraic operations on one or more images. For example, with image algebra it is possible to subtract images. The algebraic operations can be specified by the user and are programmed using a derivative of the C programming language (see section O-7.9.4).

- **Image Filters** *(Image Display & Processing)*
  Special image filters can be developed and integrated by the user. Standard filters (like Smoothing) are delivered by Bruker. Filters can be extended by defining new filter kernels (see section O-7.9.2) or by programming own filters using a derivative of the C programming language.

- **ISA functions** *(Image Display & Processing)*
  Functions for image sequence analysis (e.g. FIT functions) can be developed and integrated into ParaVision. They are programmed in a derivative of the C programming language (see chapter D-9).

- **DataManager Conversion** *(DataManager)*
  The DataManager allows the conversion of complete dataset directories as they are stored on disk. Therefore, scripts are used which need not to be macros as defined above (see section O-12.9.6.2).