Migration of an APL-system from APL+Win to Dyalog

Dr. Markos Mitsos

In DKV/ERGO APL has been used for many years for reference systems and controls on databases, but also for specifications on the technical level as well as steering purposes. A highly interconnected sheaf of APL+Win workspaces has developed, that may be described at another place. This must be migrated to Dyalog.

For various reasons, among others lack or time and developers, this turns out to be a long-time project. Underlying the migration is a workspace management framework. This has been described in the APL-Journal of APL-Germany under the title "Management of Dyalog APL-workspaces".

It uses **Link** for managing code in Unicode text files, TortoiseSVN for versioning them and Array Notation as well as DB2 for debugging and documenting the results of a sort of Unit tests. Of course the latter lead to even more migration work — to create the tests!

Because of constraints imposed by the necessity to keep operation uninterrupted and very low developer capacities, it was decided to use "migration" in a broad sense. It is understood to encompass not only copying code and making absolutely necessary changes, but creating a "true" Dyalog workspace. It also means that some interfaces between workspaces must be streamlined in a way that APL+Win component files may be used as bridges between migrated and not-yet-migrated ones.

Part of the migration is for example the structure of workspaces and objects. At the highest level the "superstructure" is fitted to the requirements of the management framework (cooperation of workspaces, testing), whereas the code proper is compartmentalised in namespaces. All functional objects acquire a schematic structure using multiline headers, Migration Level 1 and clean-up at the end. Semi-globals are replaced by locals or true globals in defined paths.

Another part of the migration consist in the use of new/different methods offered by Dyalog. Multitudes of similar variables are organized into local namespaces. Some dfns are used either as stand-alone objects or locally within others. Some first trains appear sparingly in the code. Primitives not available in APL+Win are used and partially replace complicated algorithms. Newer system functions are used instead of self-written ones or enhance readability.

Parts of workspaces touching Object Oriented Programming have to be more or less rewritten, as APL+Win does support the concept only partially. OOP is only used in basic utilities. Some objects are now proper Classes, the infrastructure for schematic GUIs and COM interaction has been replaced by code using the namespace syntax.

In fact the migration is used, overloading its meaning, as an opportunity to make further changes, which would have been very difficult within the above constraints and with the additional burden of classic workspaces and no object versioning. Those may be described in another article. Some could in principle have been done in APL+Win and some not.

An example of the first sort is better error reaction/handling, of the second automated tests based on **Link** and Array Notation. Many other changes boil down to enhanced modularisation, also known as problems accumulated "historically" over the years...