Recent Advances in
Grammar-based Rapid Application Development

United Space Alliance

Charlie Fly
Jonathan Riehl

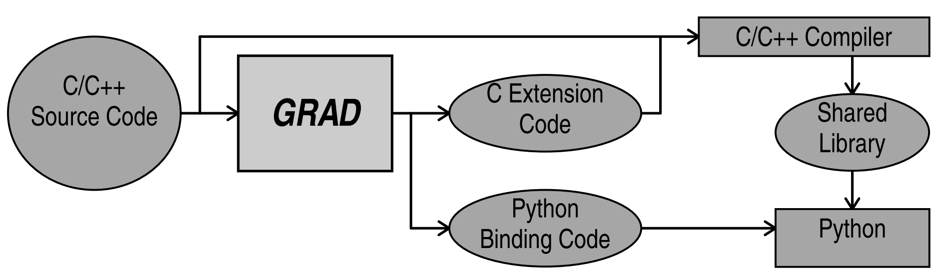
Grammar-based Rapid Application Development (GRAD) is a system for building Python interfaces for legacy systems. Additionally, GRAD allows large systems to employ Python as an interface language, replacing custom and often highly cross- coupled interface languages. GRAD attacks these problems using the grammars of other languages to build type interfaces and generate extension code for Python. GRAD allows interfaces to be built automatically without the use of scripts, and gives Python access to C/C++ pointers. GRAD infrastructure has driven other projects and yielded new capabilities, such as the PyFront Python to C interface front end.

The key to GRAD is the employment of a target language's grammar. Support tools used in internal GRAD development illustrate how a tree structure is built from a linear string of text (these trees are called abstract syntax trees or abstract parse trees.) One such tool is the C/Python parse tree browser, which will be demonstrated. Once a language's grammar is understood, it may be used to build other data structures that are manipulated or output. Specifically, GRAD employs a C/C++ grammar to generate object information that is processed and emitted as Python objects and extension code.

The prime benefit of using a native grammar is that an interface definition script is not needed. All that is required for GRAD to generate Python and C extensions for Python is the original C/C++ source code. Our demonstration will highlight the use of GRAD to build Python bindings to C++ orbit propagation software. Tkinter will then be used to visualize the data generated.

To provide the highest degree of compatibility with production C and C++ systems, GRAD furnishes an intrinsic type library that supports pointers and handles, (pointers to pointers). This feature will be illustrated as well.

Finally, GRAD has inspired other products. The most significant of these include Paths and PyFront. PyFront may be seen as an inverse of the GRAD system, in that interface code is generated in C from Python source. PyFront is related to GRAD because it uses the Python grammar to build an internal model of the Python C API calls needed to mimic the Python input script. It will be shown how PyFront may be used to generate C extension modules from pure Python.



Charlie Fly has been developing GRAD since IPC 2 (May 1995.) Mr. Fly may be reached at cdfly@ghgcorp.com, or Charlie.B.Fly@usahq.unitedspacealliance.com.
Jonathan Riehl is the developer of the Paths and PyFront offshoots. Mr. Riehl may be reached at jriehl@hypercon.com, or Jonathan.D.Riehl@usahq.unitedspacealliance.com.