Hybrids: the evolution of ConTEXT

BachoTEX, May 3, 2010
Coding in \TeX\ is quite natural and given a proper macro set the overhead is not that large.

Coding in xml makes sense when you have to manipulate or reuse your data and \TeX\ is just the renderer.

For non-artistic graphics MetaPost provides a convenient input language. It also plays well with \TeX\.

Some problems can more conveniently be solved in a procedural programming language and Lua perfectly fits in there.
Of course we started with only TEX code. Functionality has been nicely split in modules. Front- and backend code has always been separated. The user interface is quite consistent which provides backward compatibility as well extensibility. For quite some time MetaPost support has been tightly integrated, including a two way communication between these subsystems. When we decided on Lua as language it didn't take long before large chunks of Con-TEXt were rewritten using it.
Most font handling takes place in Lua and as usual with TeX we can do more than fonts provide.

Other subsystems, like languages, input encoding, file io and xml also were among the first to be supported by Lua.

Lots of information is now carried around, especially related to structure. This will permit users more freedom.

Notes, descriptions and enumerations also rely on Lua.

Graphics (including MetaPost) is all dealt with in Lua. Float management is currently on the agenda.

The backend code is completely rewritten in Lua. We've disabled the low level primitives so that third party modules can not spoil the game (this was already the case in MkII).

Eventually most management tasks will move from TeX to Lua, but we keep in pace with LuaTEX development and don't push things to the limit.
Eventually we will have a more layered macro package so that one can make specialized versions.

In addition to the regular TeX interface there will be a Lua interface. We already have one such interface but there will be more.

It will be possible to avoid TeX code completely which makes sense in predictable workflows where no artistic intervening is needed.

Core TeX functionality will also be available as (often less efficient) Lua variant so that we can extend it. We already provide hooks into the callback subsystem.

We can already all of this intermixed so the user has complete freedom of choice.