Subject: Should the pick semantics be changed?
Posted by piotr5 on Tue, 04 Mar 2014 10:00:59 GMT
View Forum Message <> Reply to Message

c++11 introduced the r-value notion with "&&" after the type. this means that in namespace std picking is default only for those r-values, deep-copy otherwise. to force picking you have to use the function std::move(). this is quite the opposite to u++ where picking always is default and deep copy must be enforced. what about a redesign to make use of the way std containers handle picking? of course such a rewrite wouldn't work with non-c++11 compilers. as far as I remember at the beginning of u++ old compiler versions were not fully supported either, so it might be a good idea to start coding up some Core lib 2.0 and rewrite all applications...

needless to say, before we do that, first the u++ code parser needs to be made ready for parsing c++11 code, so Assist and documentation will work too...

also interesting for u++ is the suffix to strings, thereby allowing for automatically generated String objects and for applying translations with less bracket-usage. I guess many more improvements could be made if u++ would break backwards-compatibility and focus on the new c++11 features...