Subject: Re: Should the pick semantics be changed?
Posted by mirek on Tue, 04 Mar 2014 11:12:28 GMT
View Forum Message <> Reply to Message

piotr5 wrote on Tue, 04 March 2014 05:00c++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...

I am well aware about '&&' and pondering this too. The problem is that it does not have (AFAIK) any composition rules:

```
struct Bar {
  int foo;
  Vector<int> bar;
};
```

Now in U++ Bar has pick semantics without doing anything. To define proper move "&&" constructor/copy, you would need to provide the code to copy all elements one by one. That might be quite tedious error-prone for classes with a large number of elements.

Also, the decision to make pick the implicit variant for containers goes a bit deeper. Consider

```
Array<Ctrl> ctrl;
```

now how would you define 'deep copy' of this?

Quote:
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...

Yes. There are some really nice features in C++11 and I will be happy to use them someday. Parser needs to be fixed for C++11 anyway (actually, I guess there is some need to fix "old" C++ too

That said, I do not think C++11 would fundamentally change the way I write software. For me, basically all needed is already there in C++98...

BTW, I just wonder, what do you mean by "suffix" to strings?

Mirek
