OO problem with NXmonochromator

Eugen Wintersberger

March 18, 2015

1 The current problem

The current version of the reference manual states that NXmonochromatorcan store instances of NXvelocity_selector and NXcrystal. At least for NXcrystal this sounds reasonable. The manual thus suggests a *composed of* relationship between NXmonochromator and the two types NXcrystaland NXvelocity_selector as depicted in Fig. 1. Unfortunately, as a discussion with Mark on the NIAC mailing list revealed, this is entirely wrong. As Mark wrote on March 10th, 2015

> ... NXmonochromator came about when we were discussing OO NeXus. And it became our first NeXus OO base class: with NXmonochromator being the base class to NXcrystal and NXvelocity_selector. ...

This changes everything. Instead of the *composed of* relationship NXmonochromator now represents a *generalization* of NXcrystal and NXvelocity_selector as shown in Fig. 2. However, this causes a severe problem: if NXcrystal and NXvelocity_selector are indeed child classes of NXmonochromator, they must not be members of NXmonochromator. For obvious reasons, a child class cannot be a member of its base class.

2 A moderate solution

There are two things we have to do in any case

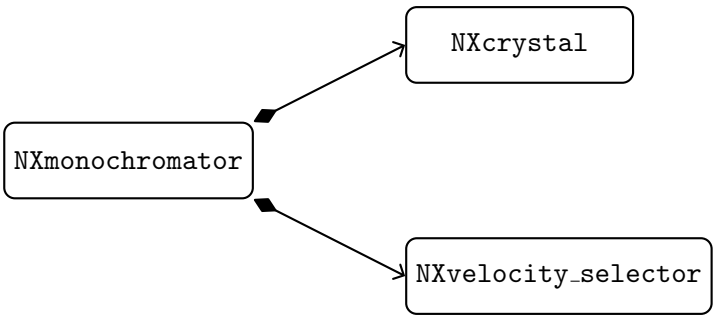

Figure 1: The current manual suggests an *composed of* relation ship between NXmonochromator and NXcrystal and NXvelocity_selector.

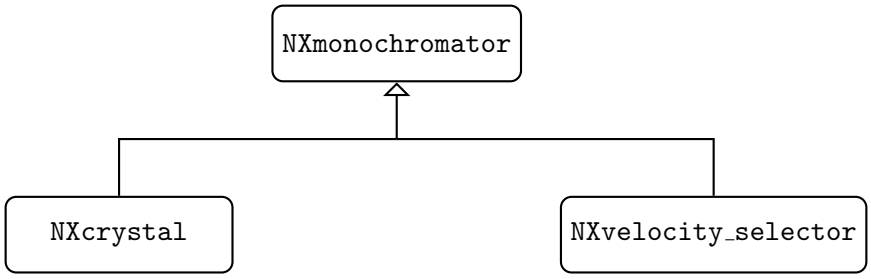

Figure 2: According to Mark, the intended relationship between `NXmonochromator`, `NXcrystal`, and `NXvelocity_selector` is generalization (inheritance).

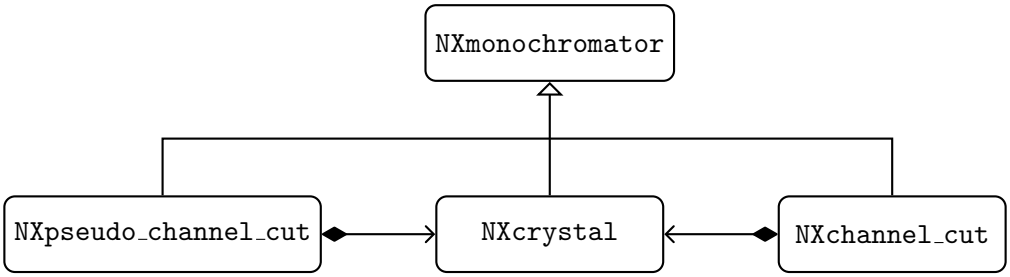

Figure 3: In its current state, `NXcrystal` can be reused only for new monochromator classes as a child class can have a composition relationship with one ore more of its siblings.

- we have to remove `NXcrystal` and `NXvelocity_selector` from `NXmonochromator` – this is plain wrong if we want to stick to the inheritance relation.
- we need a way to denote inheritance in the manual

As `NXcrystal` represents a single crystal monochromator we loose the type as a general purpose class for describing crystals appearing at all kinds of places at a beamline. Like for instance

- a hypothetical analyzer class (`NXanalyser`). Analyzers typically contain crystals.
- a beamline may wants to use `NXcrystal` as a description of a crystalline sample.

There are most probably much more situations where a class describing a crystal would be useful. Its strong coupling to `NXmonochromator` thus limits the reusability of `NXcrystal`. However, at least for new children of `NXmonochromator` `NXcrystal` can be reused as shown in Fig. 3.

This is possible because child classes can have a *composed of* relationship with one ore more of their siblings.

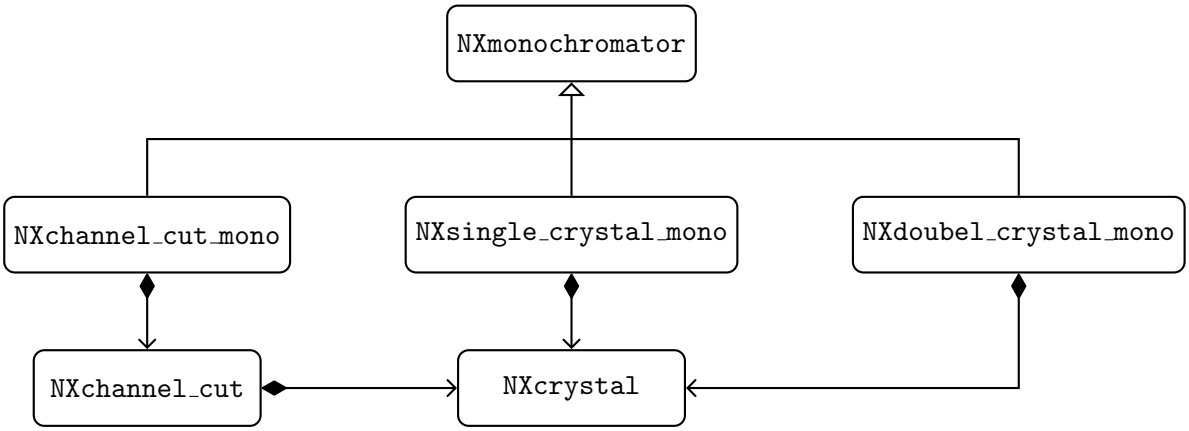

Figure 4: The new monochromator classes for hard x-rays once `NXcrystal` has been removed as a child class of `NXmonochromator`.

3 A radical solution

The best solution, however, would be to entirely strip of all monochromator specific attributes from `NXcrystal` and transform it into a generic class to store crystal information. In this case it could be used whenever information about a crystal structure is needed. The monochromators (at least for hard x-rays) would become

- `NXsingle_cyrstal_mono` which replaces `NXcrystal`
- `NXdouble_crystal_mono` a monochromator with two crystals
- `NXchannel_cut_mono` a monochromator with a channel cut

The relationship between these new monochromator classes and `NXmonochromator` and `NXcrystal` is depicted in Fig. 4. As channel cuts are not only used in monochromators but also in analyzers a new class `NXchannel_cut` has been added.

As `NXcrystal` is no longer a child class of `NXmonochromator` one could even use `NXcrystal` now as a member type for `NXmonochromator`. This would allow to construct entirely new monochromator types which are not covered by the three types presented here.