Future for Geant4 Python

Ben Morgan

Why discuss the future of Geant4 Python bindings?

- *NB: Playing **devil's advocate** throughout to stimulate discussion - it is **not a criticism** of work that has/is being done (I'm supportive of bindings!!)*

- Fundamentally a question on **available FTE** for development and user support
  - *Geant4Py only has a fraction-of-a-fraction of Koichi's and my time*
  - *The amount of work needed should **not be underestimated** - it's effectively a **whole extra system to test, validate, provide examples for, and document**.*
  - *Also consider **compiler/platform/Python version differences**…*
- **Can we therefore maintain Geant4Py at *production quality, long term*?**
  - *i.e. all features from C++ functional, documented alongside C++ version, Pythonic versions of at least basic examples, validated for physics, performance*
  - ***I think the answer in the short term is "no"***

Not a unique problem

- Career structure means we do lose developers - our “bus factor” can be very low for components, especially over longer periods
  - *Technology also moves on, so tools/libraries/syntax can also be lost through obsolescence or lack of upstream support*
- Discussing the bus factor issue this week, but will inevitably have points where this drops to zero for parts of Geant4 due to FTE/technology loss
  - *What do we do with the affected code then?*
- Not suggesting it’s immediately expunged from the repo, equally it shouldn’t remain indefinitely…
  - ***I think we are very good at adding (good!) code/capabilities, less so at removing it when it is becoming obsolete/unusable/no longer a requirement***
  - ***Should recognize that for users, code presence can => full, long term, support for it***
- All this is really saying is we can and should be blunter about deprecating and then removing unused/unmaintain(ed/able) components.
  - ***Easy*** *to mark code as such to provide early warning to users*
  - ***Also a way to identify ongoing requirements (as users may complain!), or to identify new contributors (you want it, you maintain it!), or to make case for support to funders.***

What, then, is the future for Geant4 Python?

- Perhaps the first thing is to regauge the user/stakeholder requirement for having a Python (*or any other language du jour*) binding
    - *Actual level of interest beyond "nice to have/expected/don't know C++"*
    - *Use cases, though I think this boils down to "write example B1 in Python"*
    - *More importantly, **find people who would be willing and able to contribute***
- **Already have a couple of places to start:**
    - *Chats with GATE developers in early 2021 as they had looked at Python, but little time to follow up since (but see Susanna's presentation on Monday)*
    - *Recent (2020) geant4_pybind project on GitHub, which works very nicely with full "example B1 in Python"*
- **Any other contacts/interested parties you know of?**

Wider community engagement?

- Broaden discussion through, e.g. **HSF and other user communities**.
  - *Why not have a topical meeting, say ½ day, on "Python/Julia/etc for Simulation"...*
  - *… or use the next Technical Forum?*
  - *… or a user survey/questionnaire?*
- COVID has meant **we haven't had a User's Meeting in a long while**
  - *Equally, rise and familiarity of virtual meetings could make this easier to organise than ever…*
- Again, the totally selfish aim here is to see if…
  - *… there is a significant demand for Python/etc bindings to Geant4*
  - *… there is a pool of contributors sufficient to develop and support these bindings at **production quality over a non-trivial timescale***

Development Model?

- *Could* be contributor level ***within*** Geant4 to begin with
    - *I think we are probably talking about scrapping Geant4Py and starting from something like geant4_pybind as a base, depending on discussions/input with developers/users*
- Alternately, and perhaps ***more realistically***, we support community efforts to provide the functionality, somewhat like we do for CAD interfaces,
    - *One possibility here would be to host any future "Geant4 community Python" on GitHub under our organization: https://github.com/geant4*
    - *Not Geant4 itself, and only building on **public release** code, so no concern over IP/physics validity of underlying toolkit?*
    - *Equally, how to mark as "community provided", ensure validation, avoid specialization for specific project etc?*
- ***All of these, and preceeding, points are for discussion…***