**HF Stability Working Group**

13 Dec 2021

**Present**: Adam Bergmark, Chris Dornan, Ben Gamari, Mikolaj, Konarski, Trevis Elser, Alex Bernauer, Bodigrim, Dmitrii Kovanikov, Neil Mitchell, Oleg Grenus, Simon PJ

**Working documents:**

* [The Stability Working Group Charter](https://docs.google.com/document/u/0/d/13tPObTEVhhjYIDvK9JmyLSq29nnX2879eWwpj50hHOE/edit)
* [Managing Change in the Haskell Ecosystem: a draft proposal](https://docs.google.com/document/d/13YqDmtfvVNpum5BLLLFF6Hq_c7hLOOrrLYNyezPEQjo/edit#heading=h.qhz5p8r8z60z)

**Question 1:** in the draft SWG charter, is the proposed purpose of the group sensible, well expressed?

General assent. Encouragement to everyone to edit (or comment on) the draft Charter to make it easy for everyone to understand.

Discussion around the question.

* Ben: language/ecosystem makes it hard to evolve libraries without breakage. Question: are there things we could do within the language to make it support evolution better?
* Neil: changes to ecosystem/setup is more disruptive e.g. “cabal puts something in a different place”.
* Adam: Let’s emphasize tooling, not process. Technical solutions over bureaucratic solutions.
* Viktor: encourage/guide library authors to
  + use pattern synonyms more vigorously to reduce API change. (E.g. worked in bytestring when the length field was dropped.) Template Haskell is often used with shims to reduce the impact of change.
  + use record field names more extensively
* Dmitrii. Main problem is (a) how breaking changes are **communicated** and (b) people’s differing **expectations**.
* Chris: we should (a) track and (b) review breaking changes. Danger: an extra layer of process/bureaucracy to Get Stuff Done.
* Mikolaj: for Cabal we don’t have enough developers anyway, so adding new process would be a burden. But we have a few major breaking changes to come (e.g. remove v1 cabal) -- here, it would be helpful share responsibility for the timing and nature of breaking changes.
* Bodigrim: not so keen on becoming an advisory board, more a central knowledge base. Adding burdens on implementors might make them break fewer things!
* Victor: slender resources make it harder to avoid breakage. (SPJ: More resources might not improve matters on this front.)
* SPJ: in Haskell we do try to actually fix problems, rather than regard them as now un-changeable, so that the language and its ecosystem gradually accrete a historical raft of compromises. Bodigrim: let’s articulate this principle more clearly.
* Ben: Rust does a better job of making it clear what is stable and what is not.
* Trevis: for application developers, breaking changes in tooling can be more disruptive than breaking changes in tooling. e.g.
  + change to ?? tool, which required a new version of GHC, which led to a chain of consequences.
  + where a path was changed which led to link errors (path wasn’t in CI cache).

Library changes are less disruptive. But slowness of getting libraries updated (even trivially) is quite hard.

* Viktor: reverse dependencies would be useful for library authors to know. Can we do finer grain, so we know who will be affected by a change in function ‘foo’.
* Syd: as a package author, I’m very happy if someone fixes my package for me. SPJ: we need a way for these fixes to land as a release of the package.
* Alex: if we had clearer communication about (a) nature and (b) timing of changes, would that help? Syd: a bit, but not much.
* Trevis: Stackage LTSs have helped a lot: we know that a particular LTS is known to work together. Syd: we update one LTS at a time -- big jumps.
* Pepe: has upgraded SC; and then Facebook; and is in the HLS team (a big project with a lot of dependencies).
  + Industry don’t really care about breakage. Upgrade every couple of years, use head.hackage, Stackage LTS, etc
  + Open source developers are much more sensitive. Big problem is getting dependencies updated, cannot rely on head.hackage for this. And its my own time. And maintainers often go AWOL.
* Alex: is the delay in getting access to new features a problem? Syd: no, all the important stuff is there already. Trevis: yes a bit.
* Pepe: big users have their own fork of GHC and/or packages.
* Mikolaj: would like to be able to get help with package upgrades.

General classes of solution

* Better communication
* Better tooling
* Less breakage, period. (Or maybe less frequent.)

**Here is a place to collect ideas for what to do, to mitigate the cost of changes**

* Syd’s idea (because SPJ proposed we write down ideas by next time): Help the people who work for the betterment of the ecosystem for free
* Fewer GHC releases (yearly instead of bi-yearly)
* More support for Stackage LTS - at least make sure that it doesn’t stop existing
* Introduce language and compiler features which facilitate evolution of libraries without breaking users
* Extend hackage
  + Hackage could allow for maintainers to be notified when a dependency has been uploaded that falls out of the range specified. So maintainers of package X that depends on Y <1.2 would be notified when 1.3 is uploaded. This could help the situation where maintainers do not know they need to take action.
    - Discussed was if hackage could show consumers of a particular export (function, type, etc). I believe the answer was that yes it could be extended to do so. If hackage had that ability then you could as a package maintainer (or in a similar automated notification system) see the impact of a change. Suppose that you have a release candidate of some package, you could see the impact it would have on the consumers in hackage and use that to inform the release process for your package. I imagine this could be used by the GHC developers as well to inform if compatibility shims should be released or not.
  + Visibility could be enhanced if packages could opt-in to some kind of PVP enforcement, similar to what Elm has done, but optional as to not break backwards compatibility, and displayed somewhere on the package information. Then as a consumer you can feel somewhat more assured about how the versioning of the package is put together.
* Dmitrii's idea:
  + Write in some official place (Haskell.org or at least haskell.foundation) something like "One of the Haskell goals is to constantly evolve and this implies having breaking changes from time to time".
    - This statement helps to align community expectations with common goals and values. Additionally, this reduces burnout of maintainers because they at least know what to expect (to some degree).
  + Introduce a separate group of people responsible for maintaining packages in the ecosystem. They should have rights to contribute patches, accept them and release new versions to Hackage. Maintainers opt-in for the process of having such support from the group.