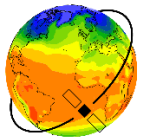
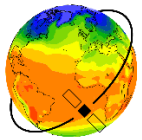


ESMValTool

Earth System Model Evaluation Tool

Developing our strategy on backward incompatible changes

Discussion points for June 2022 workshop



ESMValTool

Earth System Model Evaluation Tool

How the strategy is being developed

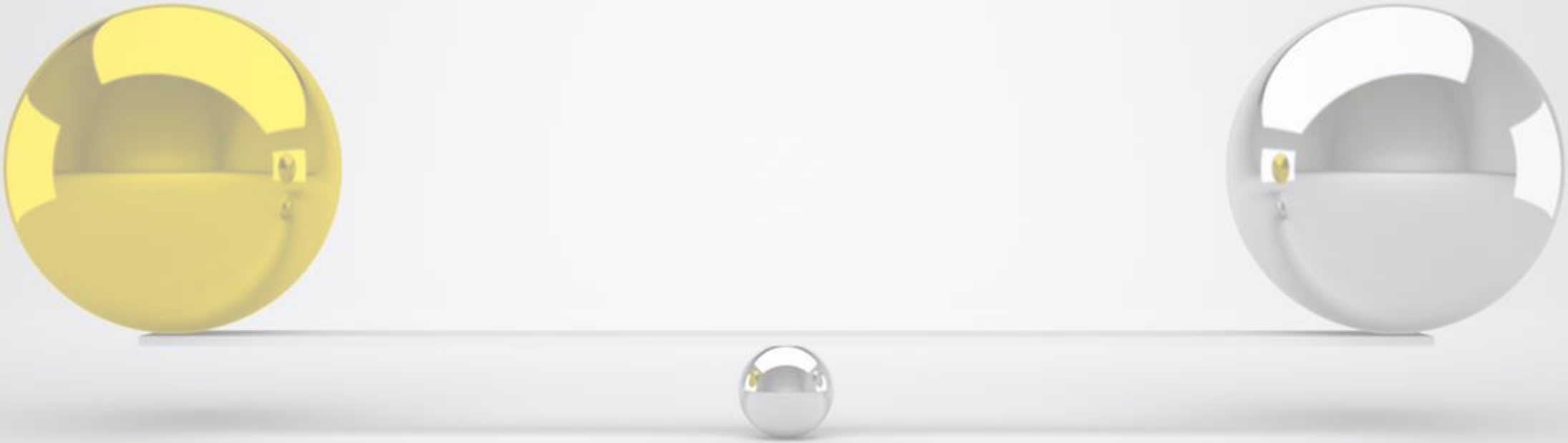
1. Discussion session at Nov 2021 workshop
2. Discussion document circulated to community
3. “Proposal team” will develop the contributed ideas and put a proposal to the community. The team is Alistair, Axel, Bouwe, Emma.
4. Community feeds back and we iterate, and aim for general consensus on a v1 strategy.
5. Implement, monitor & review

Definitions

- **Scientific consistency**: results are scientifically equivalent to those from previous release, such that scientific conclusions would not be affected. An upgrade may maintain scientific consistency for a given recipe even if results are not identical.
  - We don't currently have the capability to automatically test for this
- **Breaking change**: a change which prevents an existing recipe from running.
  - Note that a non-breaking change (e.g. a bug fix) may still alter results, and hence not preserve scientific consistency
- **Breaking release**: a release containing one or more breaking changes

Key points

(this and following slides contain my *personal* view of discussion thus far)



- Backward compatibility policies strike a balance between:
  - Effort for users to upgrade existing recipes/diagnostics to a new version
  - Effort for developers to introduce new capability without breaking recipes
    - (need to consider novice developers as well as core developers)
- Where do we as a community want to sit on this scale?

Where do we want to sit on this scale?

(for now)

- ~= How often should we have a breaking release?
  - …approximately (it could be flexed in light of specific issues)

- How should we signal to users that a release will break some recipes?
  - Breaking release = major release?

Helping users to upgrade

- General advice on upgrading in tutorial, e.g.
  - only use released versions
  - what to expect for breaking and non-breaking releases
  - importance of checking release notes
  - even when code appears to run OK, to check the *scientific consistency* of their output between the old and new versions
- Release-specific guidance in release notes, e.g.
  - for each breaking change include “how to” on adapting recipes/diagnostics
  - more prominent communication that a release will break some recipes
    - use multiple channels: release notes, mailing list, +++

Helping developers to help users

- Make it easy for developers provide the “how to” information for release notes
- Give (more) guidance to developers, including (more) on deprecation
  - Note that deprecation adds effort and is not a silver bullet
- Reviewers also need guidance
- Testing: help developers know as early as possible whether their change might break something