AT on GitHub
Development and integration

Laurent Farvacque

AT HISTORY

- ~2000: The Accelerator Toolbox is initiated at SLAC by A.Terebilo
- 2009: The Matlab code is set as a collaborative open-source project on SourceForge (Boaz Nash)
- 2017: Introduction of PyAT (Will Rogers)
- 2017: Move from SourceForge to GitHub

What do we get from GitHub:

- Storage of the source code
  - Keeping a full history of modifications (git)
- Tools for collaborative development:
  - Handling of conflicts, (“Pull requests”),
  - Automated testing, (GitHub actions),
  - Reporting of problems, (“Issues”),
  - Discussions,
  - Many more that we don’t use yet…

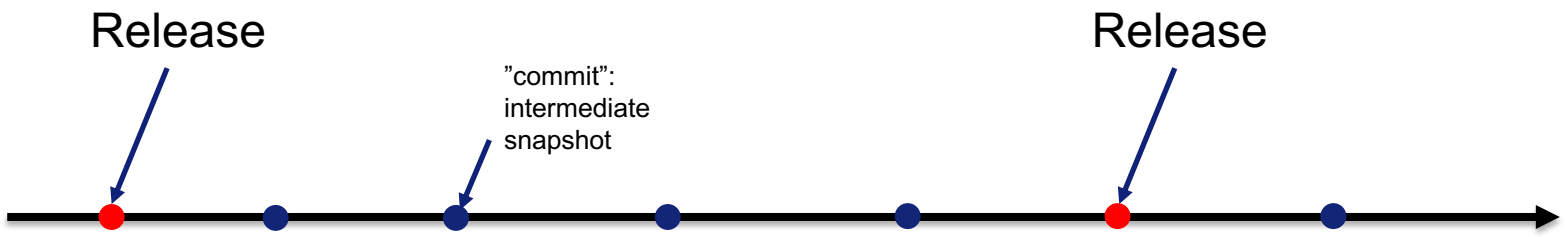
What I will show may be

- Either too technical for those more interested in using AT rather than developing it,
- Or obvious for those familiar with collaborative development.

The hope is to motivate more of you to contribute to AT by:

- Simply reporting problems,
- Proposing new features,
- Of course contributing to the code!

DEVELOPMENT CYCLE



Periodic releases, distributed as pre-built packages

Simplest and safest way to install AT: no system requirement

- Matlab: distribution on Matlab File Exchange
- Python: distribution on the Python Package Index (PyPI)

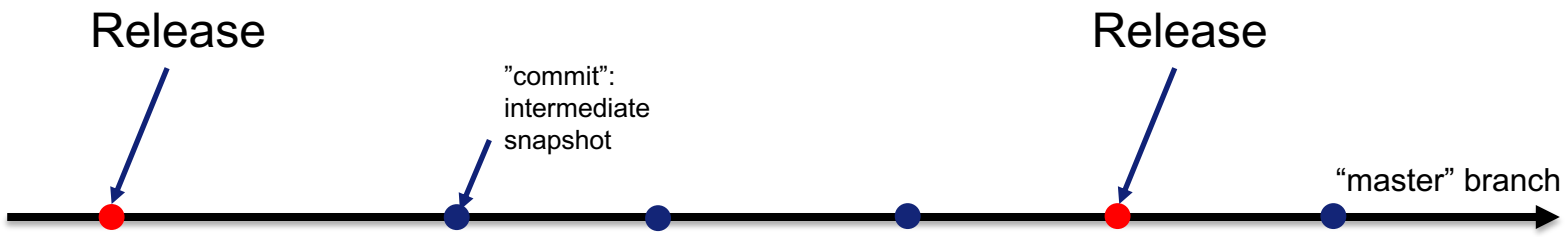
Each release is accompanied by release notes summarising all the modification since the previous one.

A new release is issued every 6th month, Matlab and Python releases are not synchronised.

Intermediate snapshots

- Allows access to the latest developments,
- Needs git and development tools (C compiler) to be used,
- Available by cloning the GitHub repository and building from the source.

DEVELOPMENT CYCLE



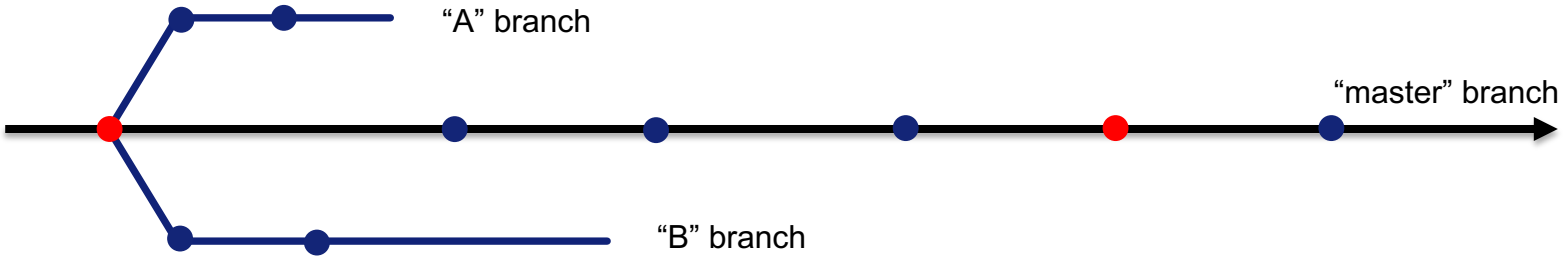
Any development is done in its specific branch, so that it does not interfere with the “master” branch

DEVELOPMENT CYCLE

Any development is done in its specific branch, so that it does not interfere with the “master” branch

- Branch “A” is started

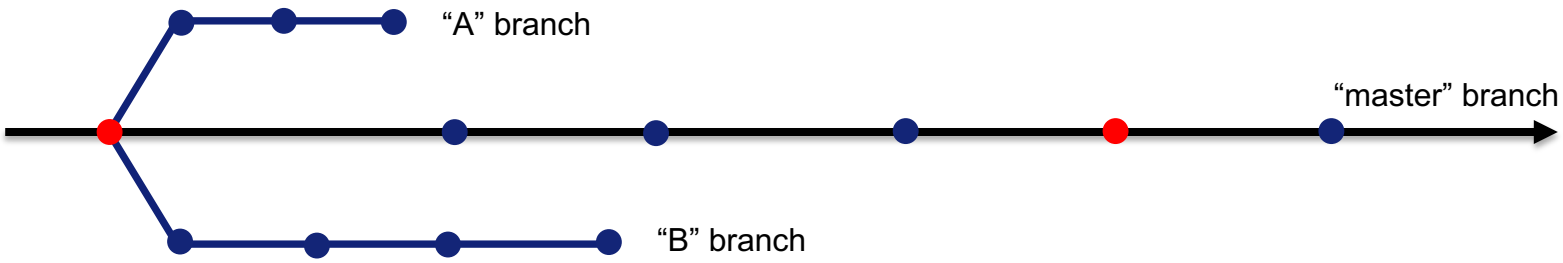
DEVELOPMENT CYCLE



Any development is done in its specific branch, so that it does not interfere with the “master” branch

- Branch “A” is started
- Branch “B” is started

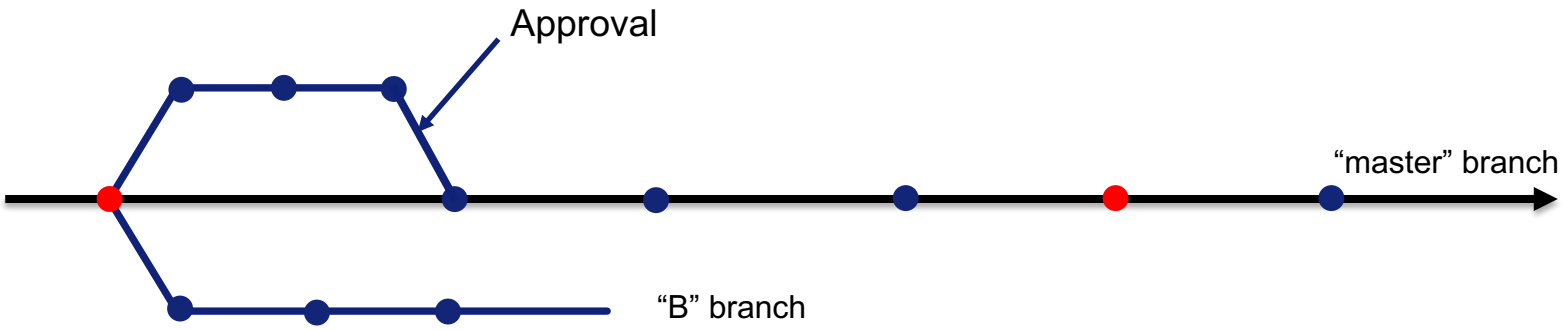
DEVELOPMENT CYCLE



Any development is done in its specific branch, so that it does not interfere with the “master” branch

- Branch “A” is started
- Branch “B” is started
- Development runs in parallel

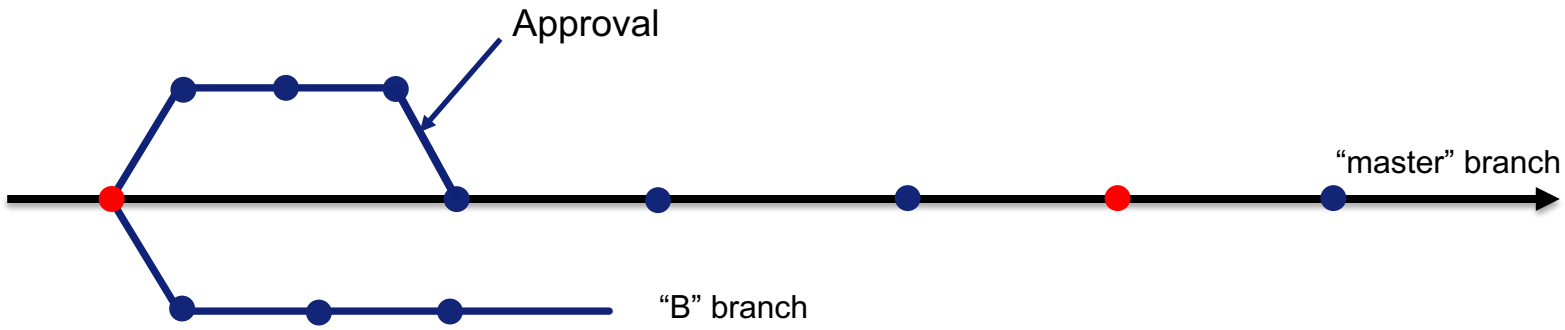
DEVELOPMENT CYCLE



Any development is done in its specific branch, so that it does not interfere with the “master” branch

- Branch “A” is started
- Branch “B” is started
- Development runs in parallel
- Branch “A” is approved and merged into the main branch

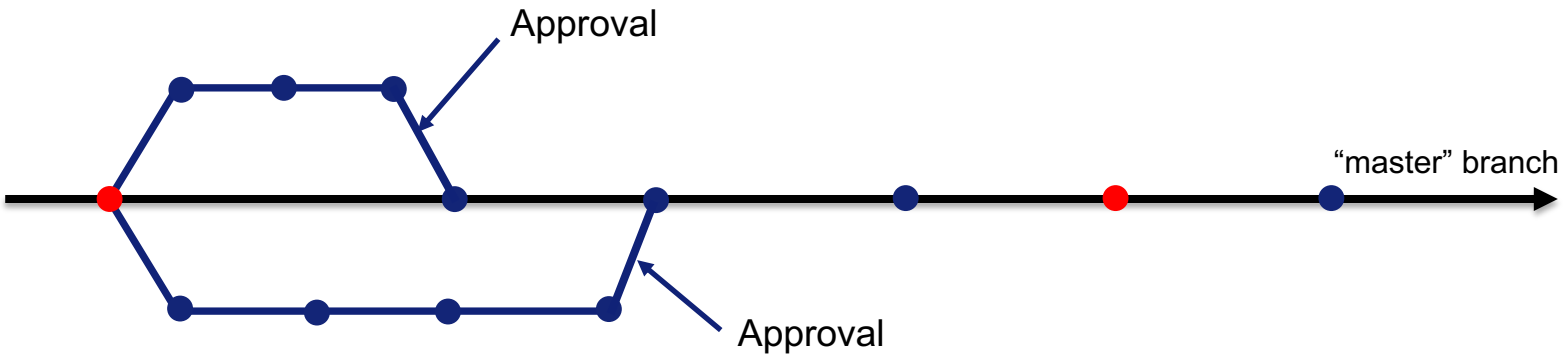
DEVELOPMENT CYCLE



Any development is done in its specific branch, so that it does not interfere with the "master" branch

- Branch "A" is started
- Branch "B" is started
- Development runs in parallel
- Branch "A" is approved and merged into the main branch. Approval:
  - Check that there is no conflict with the master branch,
  - A series of tests is run automatically on each commit: both python and Matlab tests, including comparison of the results of both versions,
  - At least one reviewer must approve.

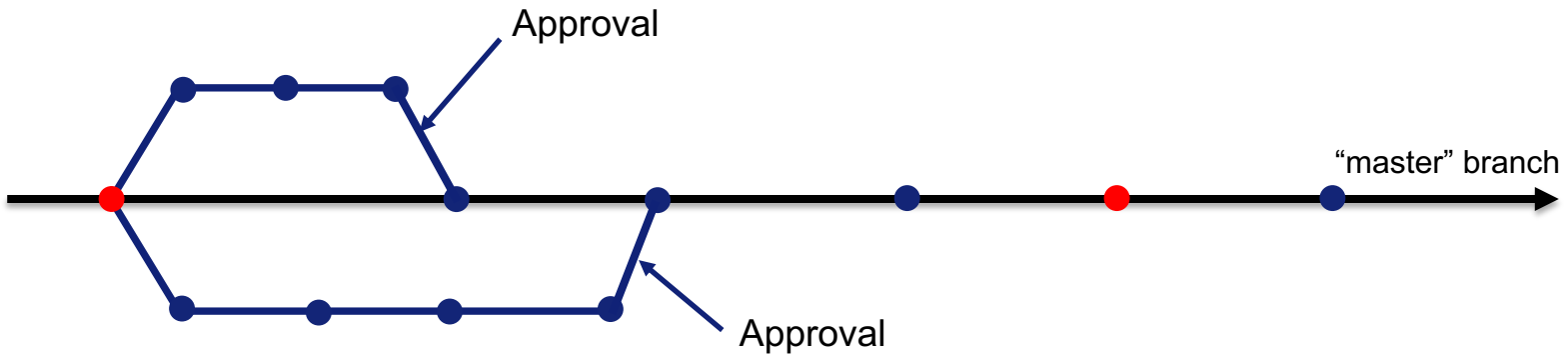
DEVELOPMENT CYCLE



Any development is done in its specific branch, so that it does not interfere with the "main" branch

- Branch "A" is started
- Branch "B" is started
- Development runs in parallel
- Branch "A" is approved and merged into the main branch
- Branch "B" is approved

DEVELOPMENT CYCLE



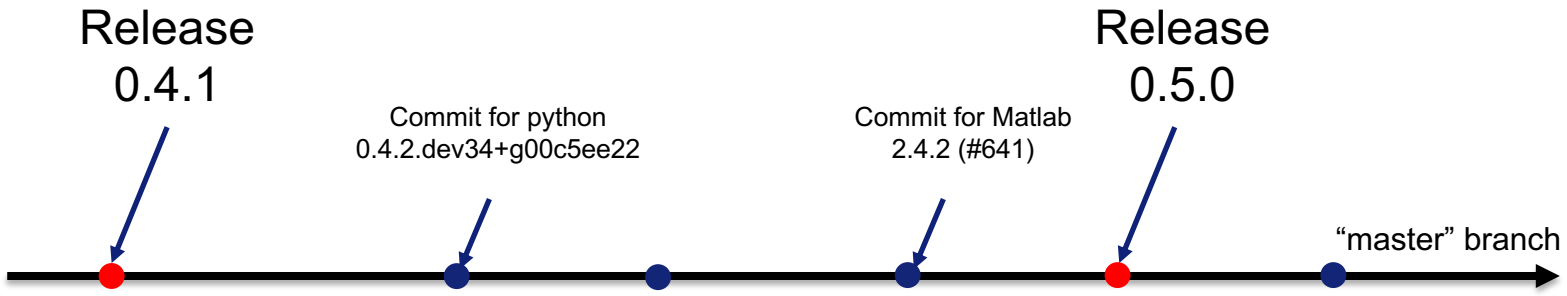
Any development is done in its specific branch, so that it does not interfere with the "master" branch,

The "master" branch is considered "safe" for users (no accident identified so far),

All branches are public: anyone can clone the "master" branch or any development branch,

The creation of new branches is allowed for a set of declared "contributors". However, anybody can instead create a "fork" and propose his/her contribution to be merged.

VERSIONING



AT is still evolving fast, so version numbers are useful when reporting problems

Release version number: major.minor.patch

- Major: New features, possibly breaking the compatibility
- Minor: new features, keeping the full backward compatibility
- Patch: bug fixes

The release notes are available on GitHub

GitHub also keeps the history of releases

Commit version numbers:

- For python, the versioning is handled automatically
- for Matlab is must be done manually (less reliable)

GOVERNANCE

Still vague…

Role:

- Ensure the stability
  - Check the full backward compatibility,
  - Check the correctness.
- Ensure the consistency
  - Check the relevance of new proposals,
  - The manipulated objects, notions, quantities are consistent across all functions or classes,
  - The same notion is referred to by the same name everywhere.

Status:

- 16 members of the project (not all active)
  - 12 “owners”: all permissions
  - 4 “members”
- A few non-member contributors
- Very few identified reviewers (~5)
- More users active in reporting issues, raising discussions, asking for new features

GOVERNANCE

From the survey results:

- Is the continuous development approach used until now satisfactory?
  - No: 11%
  - Maybe: 57%

⇒ There is obviously a large potential for improvement !

Difficulties:

- Review process: what is expected from a reviewer:
  - Checking the involved physics? Checking the results? Checking the code? Checking the documentation? Testing?
  - Depending on the choice, it may take a lot of time,
  - 2 languages, missing Matlab reviewers.

- Should we be more formal? Or less?
  - Defining and assigning responsibilities?
  - Imposing coding style?

Documentation

Both languages prompt to document any object inside the code.

But even if each function/class is documented, that's not enough: if you don't know which function you need, it does not help . We need:

- Tutorials,
- User guides,
- How-to documents,
- …

- Python

  The Web documentation is automatically generated:

  - All packages, modules, classes, functions are documented on the Web site,
  - Some manually written sections are available (primer, how-to), but we need more.

- Matlab

  The documentation is both on the web site and in the standard Matlab help, but

  - Everything has to be done manually.
  - So it's late compared to Python

WEAK POINTS

Code maintenance

The Matlab part is cluttered with a lot of unusable code (missing references…), but cleaning is very difficult…

Testing

- Testing with GitHub actions is very powerful. It runs on:
  - Linux, Windows and macOS platforms,
  - All the supported python version,
  - Recent Matlab versions.
- It proves daily very useful,
- But writing relevant tests is a delicate task.

License

AT has no license. Is it useful? Which one?

UPCOMING NEW FEATURES

Non-relativistic tracking

- Tracking: well delimited task
- Derived computations: spread around in many places.

"Exact" pass methods, based on E. Forest's book (PTC)

- Drifts: `ExactDriftPass`
- Straight multipoles: `ExactMultipolePass`
- Rectangular and sector bending magnets: `ExactSectorBendPass` `ExactRectangularBendPass`

Parametrised lattices

- Can assign a scalar parameter to one or several scalar attributes of an element (or items of array attributes)
- Then the parameter can be freely varied (for matching, parameter scans,…)
- Python only, because of the limitations of the Matlab language

CONCLUSION

I am confident that this workshop will give many ideas for new features to be added to AT…

But I hope that it will also contribute in improving the organisation and governance of the project.

And that it will motivate many people to actively participate by:

- Contributing to the code,
- Being involved in the reviewing process.
- Or simply reporting bugs, ambiguities, incomplete or wrong documentation,…

https://github.com/atcollab/at