

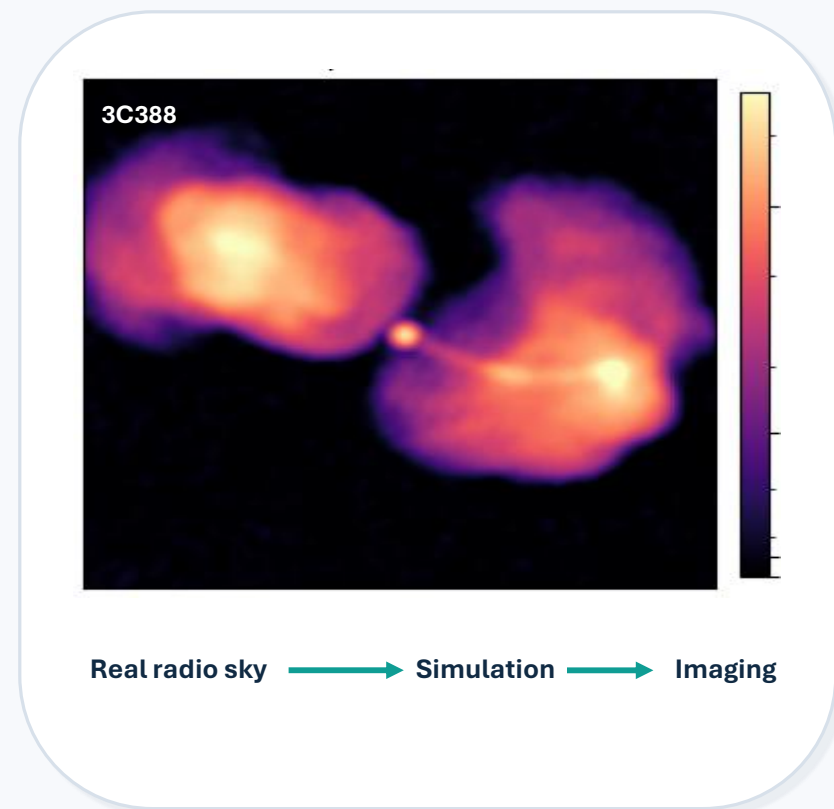
SKA-SDP-ready simulations and imaging for radio astronomy

Karabo Pipeline, current progress and a realistic 3C388 workflow

Massimo De Santis

Haute Ecole Arc / HES-SO

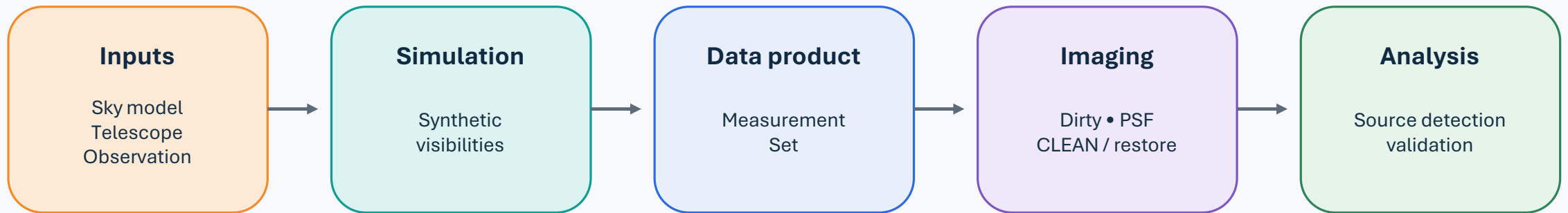
And the Karabo development team



What is Karabo?

A Python framework that connects simulation, imaging and analysis workflows

Karabo provides a common Python layer around radio-astronomy tools and data products.



Open and reusable

Open-source Python workflows, notebooks and standard radio-astronomy data products.

Backend-neutral

Users work through Karabo APIs instead of wiring every backend-specific detail themselves.

Built for experimentation

Useful for simulations, algorithm comparison, pipeline validation and reproducible examples.

Why move away from the RASCIL-based stack?

The migration is primarily about maintainability and the next generation of Karabo

What was holding us back

Old dependency constraints

RASCIL kept Karabo tied to the NumPy 1 generation and made broader upgrades difficult.

Backend-specific complexity

Simulation and imaging paths accumulated implementation-specific glue and special cases.

Portability pressure

Modern HPC environments and requested ARM support need a cleaner, more current dependency stack.



What this unlocks

SKA-aligned foundations

Simulation and imaging now offers the SKA-SDP ecosystem behind Karabo interfaces.

Modern dependency upgrades

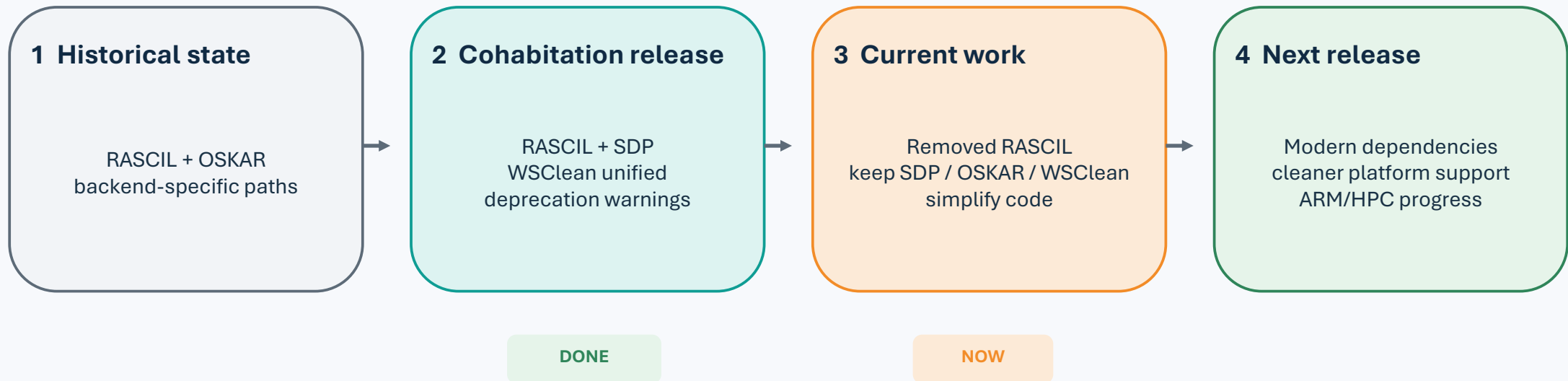
Removing the remaining RASCIL dependency opens the way to NumPy 2 and related upgrades.

ARM / HPC path

A modern stack makes broader platform support much more realistic — without claiming it is finished yet.

Where we are now

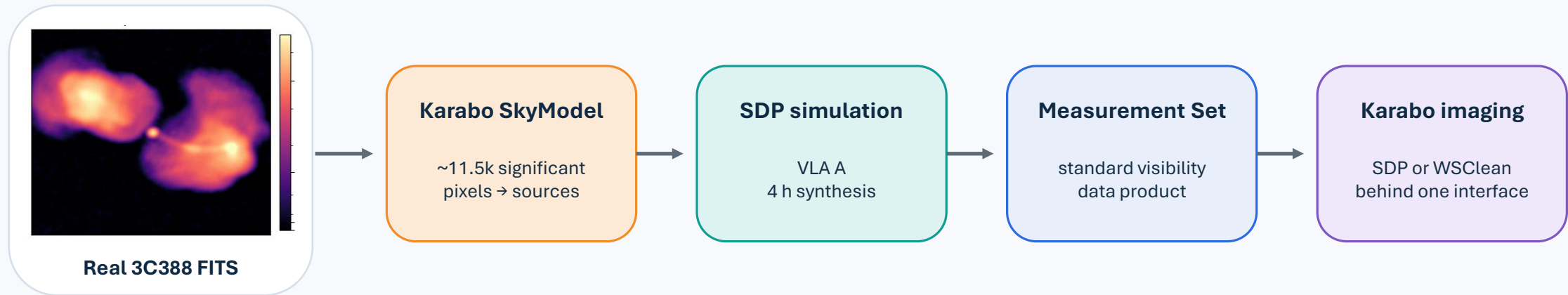
A controlled migration: coexist first, remove later



The important transition principle: existing users were not broken while SDP support was validated.

Realistic end-to-end example: 3C388

A real radio image as the sky → synthetic observation → reconstructed image



Why this is a useful test

The input morphology comes from a real radio source rather than a hand-made synthetic catalogue.

What is simulated

Karabo creates the interferometric observation and visibility data from the sky model.

What we validate

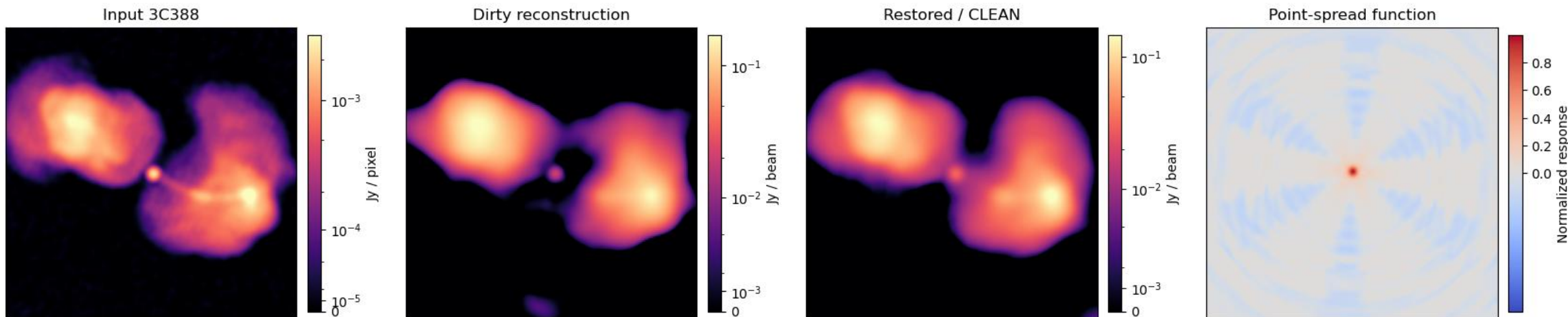
Can the new imaging stack recover the location, angular scale and double-lobed morphology?

Input image: 3C388 at ~1.385 GHz (prepared J2000, Jy/pixel)

The new SDP path reconstructs the source end to end

Input sky → dirty image → CLEAN/restored image → PSF

3C388: real FITS sky → SDP simulation → unified SDP imaging



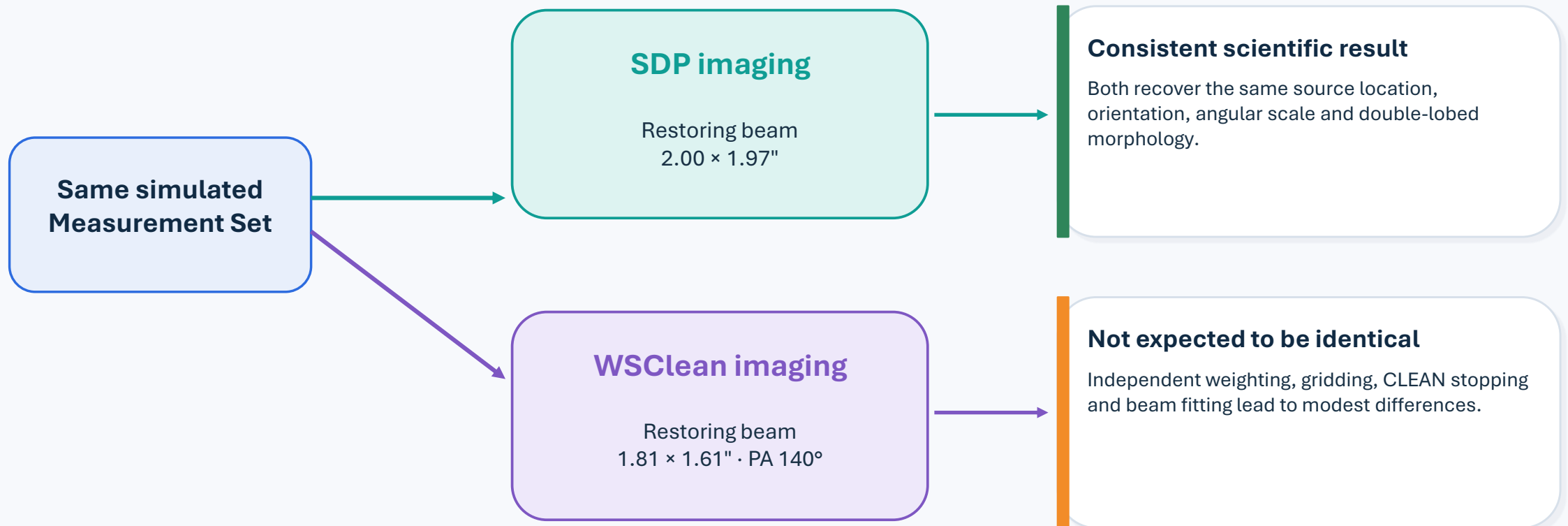
Takeaway

The simulated observation preserves the recognizable 3C388 double-lobed morphology, while the PSF and CLEAN/restoration are produced through the SDP imaging path.

Note: input is Jy/pixel; reconstructed images are Jy/beam, so peak values are not directly comparable.

One Measurement Set, two imaging backends

Karabo now exposes SDP and WSClean through the same imaging abstraction

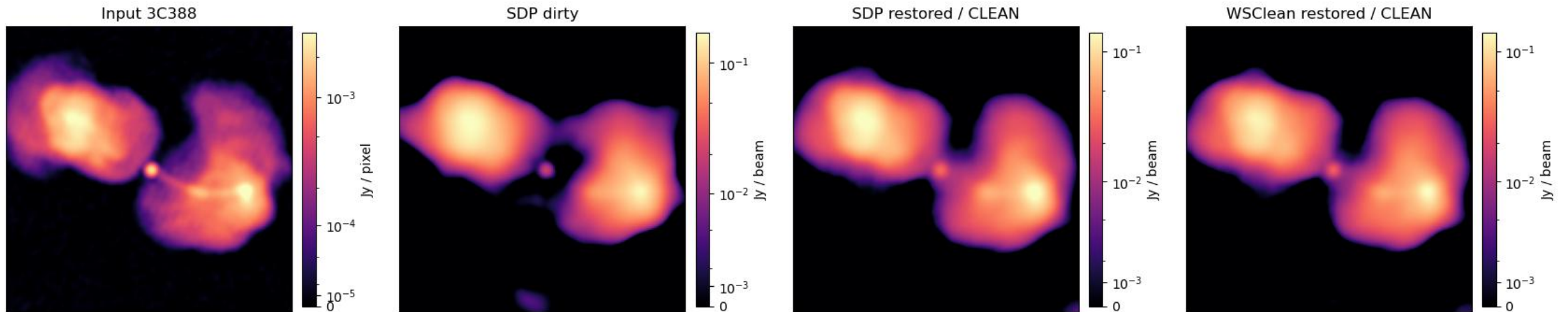


Karabo's value here is the common workflow: the same visibility data can be sent to different imaging implementations without rebuilding the pipeline.

Visual comparison: SDP and WSClean

Same simulated Measurement Set, imaged through the two unified Karabo backends

3C388: one SDP simulation → two unified imaging backends



Both backends recover the same overall morphology and sky structure; the remaining differences are modest and scientifically plausible.

What comes next?

RASCIL is now removed, and the migration is unlocking the next round of Karabo upgrades

Done

- Complete RASCIL removal
- Keep SDP / OSKAR / WSClean workflows stable
- End the temporary cohabitation phase

Now

- Upgrade NumPy and previously blocked scientific dependencies
- Continue package modernization
- Improve packaging, portability and ARM / HPC support

With users

- Test realistic astrophysical workflows
- Validate scientific outputs beyond internal parity tests
- Report missing functionality and platform needs

The next useful validation is not another synthetic unit test, it is real user workflows.

Karabo is moving from a successful backend migration to a more modern, portable and user-driven framework.

Try it • test it • tell us what breaks

github.com/i4Ds/Karabo-Pipeline

Thank you

Thanks to all Karabo contributors and collaborators.
Massimo De Santis • HE-Arc / HES-SO