natESM

# Newsletter

ISSUE 13 / JULY / 2026

## Community workshop 2026

Our Community Workshop in Leipzig once again brought together researchers, developers and RSEs from across the community. With more participants than ever before, the workshop was characterized by lively discussions, many new connections and an exceptionally active exchange throughout both days.

A highlight was the keynote by Peer Nowack on the future role of AI in Earth system modelling, which sparked many discussions that continued well beyond the session itself.

This year's programme also featured two breakout groups on Impact Modelling and Palaeoclimate Modelling, providing dedicated opportunities for scientific exchange within these parts of the natESM community. In addition, participants could learn more about the new ComIn Plugin Interface in a dedicated technical session.

👉 All presentations are available on our website.

## natESM on tour

This spring, natESM was represented at several international conferences.

In April, Process Coordinator Iris Ehlert presented natESM at the Baltic Earth Conference, introducing the national Earth system modelling initiative to the wider Baltic Earth community.

A few weeks later, many members of the natESM community met again at the EGU General Assembly in Vienna.

We were delighted to welcome some of you to our EGU session ESSI3.4. Within the session, our RSEs presented their latest work and ideas on...

- Wilton: The natESM Journey for Improving Software Quality in Earth System Modelling
- Aleksandar: Choosing an I/O approach for Earth system models: lessons learned from a modular I/O server for MESSy
- Aparna: Navigating legacy Earth System Model software

We would like to thank everyone who contributed to the lively discussions throughout the session.

👉 Find the abstracts of all presentations on EGU's website.

## First natESM Summer School almost ready to start

In September we will welcome 22 participants to our first natESM Summer School.

During the week, participants will work with a dedicated natESM configuration that has been developed specifically for the Summer School and reflects our long-term goal of providing the community with a growing portfolio of standard configurations.

During the second phase of natESM, we plan to develop four community-supported reference configurations addressing different scientific application areas, making it easier for researchers to get started with natESM and to build upon well-tested configurations.
The Summer-School configuration represents one important building block along this path and has been specifically designed for the needs of this year's participants.
We are especially grateful to Stephan Lorentz, Vera Schemann, Helmuth Haak, Kerstin Hartung, Julia Nabel and Birgit Hassler for helping us prepare the scientific programme.

👉 More info on our website

## Save the date: Community workshop 2027

🗓️ February 17-18, 2027 | 📍 KUBUS, Leipzig

Our next Community Workshop will take place on 17–18 February 2027 at KUBUS Leipzig.

Following the many spontaneous discussions between community members and our RSEs during this year's workshop, we will introduce a new service: on the afternoon of Day 2 you will be able to book one-on-one consultation sessions with our RSEs to discuss your code, technical challenges, or future sprint ideas.

Depending on how this new format is received, we will further develop it in future workshops.

More information and registration will follow towards the end of this year.

## Our RSE team keeps growing

We are very happy that our RSE team has continued to grow and now consists of seven Research Software Engineers, allowing us to support an even broader range of community activities.

### Vladyslav Pushenko

Joined in March at JSC and currently supports the ICON-Land Sprint together with Sergey Sukov.

### Christian Tica

Joined in June and is leading the ARTEX1 sprint.

### Julia Duras

Recently joined the team and is supporting the preparations for the summer school and the natESM configurations.

### Samin Hamidi

Joined in April as our AI-focused RSE.

We are currently exploring how AI can best support our community.

If you are interested in AI-related support, we'd greatly appreciate your feedback in our short survey until 12 August.

👉 Link to our AI survey

## natESM featured

Curious to learn more about natESM? DKRZ recently published a feature article introducing the project, its services and long-term vision.

If you're interested in the broader Research Software Engineering perspective, you may also enjoy the recent F1000 Research article "Establishing Central Research Software Engineering Units in German Research Institutions", which highlights natESM as a best-practice example of community-driven Research Software Engineering (Section 3.9).

## Join us at the ComIn training

🗓️ When: November 25, 2026
📍 Where: DKRZ, Hamburg

Following the strong interest expressed in our recent community survey, this year's natESM Technical Training will focus on ComIn.

The program begins with an overview of ComIn and its basic concepts, followed by hands-on sessions covering setup and the technical building blocks in greater depth. The day will conclude with a showcase of a complex ComIn application, a Q&A session and an outlook.
A preliminary agenda is already available on our website. Registration will open in due course.

👉 More info

## Not goodbye just yet...

Although Aparna Devulapalli recently joined the ESiWACE project, she will continue to support the preparation and delivery of the ComIn training and will join us again at DKRZ in November. Afterwards, she hopes to pursue a PhD in Meteorological Sciences. We thank Aparna for her valuable contributions to natESM and wish her every success for this exciting next step.

## Invitation to all with ideas

natESM thrives on its sprints. They are where collaboration becomes concrete, where ideas turn into code, and where the community really comes alive. So please don't hesitate to submit sprint-check requests – whether you already have a clear project in mind or are still exploring possibilities. We are very much looking forward to new sprint proposals.

👉 How to request a sprint check — and all details of the process — can be found on our website.

deep dive

Personal stories from natESM trailblazers

# More than code

## Building scientific software that lasts.

**When Patrick Jöckel was 16 years old, he wrote to atmospheric chemist Paul Crutzen asking for information for a school project. A few days later, a parcel arrived, filled with articles, reports and background material. That unexpected reply not only helped Patrick finish his assignment—it also sparked a fascination that would shape his entire scientific career. Today, MESSy forms the software backbone of a long-standing scientific community working on atmospheric chemistry and Earth system modelling. In this Deep Dive, Patrick Jöckel reflects on more than twenty years of MESSy, the value of building communities rather than just software, and why the hardest challenges were never technical.**

Interview by Iris Ehlert

### The Beginning

**Can you remember when you first became interested in climate research?**

I was around sixteen years old when I wrote a letter to Paul Crutzen. At the time, I was preparing a school assignment on climate research and simply asked whether he might have some material that could help me.

A few days later, a parcel arrived, filled with articles, reports and background information. I was completely overwhelmed. I think that moment really sparked my fascination with climate research.

**And from that moment on, the path was already clear?**

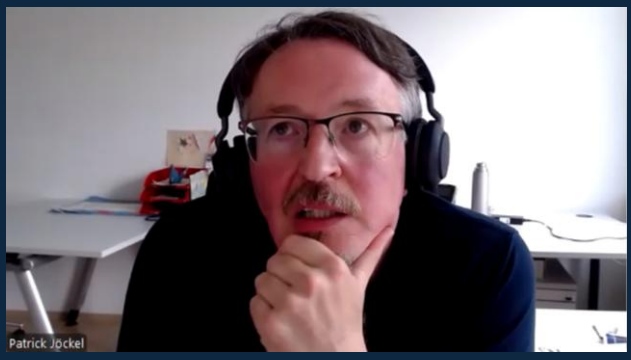
No. Towards the end of school, I had to decide what to study. I chose physics, specializing in nuclear, atomic, and plasma physics. After finishing my diploma, I applied to the Max Planck Institute for Chemistry in Mainz. Even then, the topic of my PhD wasn't fixed. It was completely open whether I would end up in laboratory work or in modelling. That decision came almost by chance.

**So, becoming a modeller wasn't the original plan?**

Not at all. But there were already five people working in the lab, so at some point I thought: "Well, there are already enough people measuring—I'll do the modelling instead."

It wasn't a strategic career decision. It just felt like the place where I could contribute most.

**Today, many people associate your name with MESSy.**




**For readers who may not know it yet: what exactly is MESSy?**

Many people think MESSy is an atmospheric chemistry model—or even an aerosol model. I completely understand why, because atmospheric chemistry was where everything started.

But that's actually a misconception. MESSy itself is neither a chemistry model nor an aerosol model. Instead, it is a software framework for assembling different components of an Earth system model into one coherent system. On the one hand, it provides the infrastructure and interfaces that allow model components to interoperate with each other. On the other hand, it already contains a large collection of scientific submodels that make use of this framework.

**» MESSy is not messy. «**

Over the years, it has grown far beyond its original purpose.

## When software became the challenge

**MESSy has been evolving for decades. Was there a master plan from the beginning?**

Not at all. Originally, we simply wanted to extend an atmospheric general circulation model with atmospheric chemistry. At the time, that was the scientific challenge we were interested in.

What we didn't anticipate was that the software itself would become the much bigger challenge. Atmospheric chemistry is not just one process. It involves emissions, transport, kinetics, aerosols, clouds, radiation, deposition—many tightly coupled processes that constantly exchange information with each other.

But there was another problem. The underlying atmospheric model, ECHAM, was still under heavy development. Every few weeks, new versions appeared with major structural changes. We knew that simply copying the code and maintaining our own version would eventually isolate us from future developments. So instead of asking "How do we add chemistry?" we started asking a different question:

"How can we make our own developments as independent as possible from software we don't control?" That question shaped everything that followed.

**When did you realize that software architecture had become the real challenge?**

Surprisingly early. Within the first few years we realized that the chemistry itself wasn't the limiting factor. We already had the scientific expertise in the team. The real challenge was organizing all these processes in a way that remained maintainable over many years.

**» Technically, almost everything is possible. The real challenges lie in human communication. «**

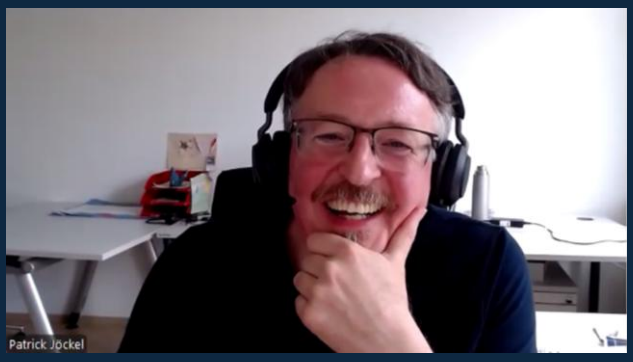
By the time we wrote our first concept paper in 2005, many of the ideas that today we would simply call modular software architecture were already there.

**Looking back, that early decision to invest in modularity rather than quick solutions seems to have shaped MESSy ever since.**

I think so. We learned very early that software has to be designed for change. Models evolve, scientific questions evolve and technology evolves. If your software cannot evolve with them, it won't survive. That's why we invested so much effort in clean interfaces and modular structures. It required more work in the beginning, but it paid off over the long term.

There were situations where we deliberately chose the longer development path because we knew it would result in a more sustainable solution. We often heard that our approach was too complicated or too expensive in terms of performance. But in the end, we always found a way to satisfy the scientific requirements without compromising the architecture or even performance. It sometimes took longer—but I think that has been one of the reasons why MESSy is still around today.

**How did MESSy actually get its name?**




Quite spontaneously, actually. We were travelling by train to a workshop and realized that we still didn't have a proper name for the project. So, we started playing around with acronyms until one colleague suddenly came up with "MESSy".

Everybody liked it immediately. Of course, we were aware that "messy" isn't exactly a flattering word in English, which is why our website has always said: "MESSy is not messy.“ It was a bit self-ironic. After all, we were trying to replace the spaghetti code we had inherited with something much cleaner and more structured.

At one point there was even the suggestion to rename the project—but by then the whole developer community was already attached to the name. So MESSy stayed MESSy.

## From code to community

**Was there a moment when you realized that MESSy had grown beyond its original scope?**

There wasn't one defining moment. It happened gradually.

One milestone I remember particularly well was our first release workshop in 2006. We invited colleagues from the atmospheric chemistry community to see what we had built and to try it themselves. We expected some interest—but we certainly didn't expect that so many groups would adopt MESSy so quickly.

That was probably the first time we realized that MESSy had grown beyond our own project. As more institutes joined, it also became clear that we needed a different way of working

together. That's why we founded the MESSy Consortium in 2013—not as a formal organization, but as a framework for collaboration.

**Many scientific software projects struggle when people move on. MESSy has been around for over two decades. What made the difference?**

I think we realized quite early that software alone isn't enough. The real challenge is communication. Different institutes naturally have overlapping scientific interests. If you don't address that openly, people can easily end up duplicating work—or even competing without realizing it.

**» Plan first. Code afterwards. «**

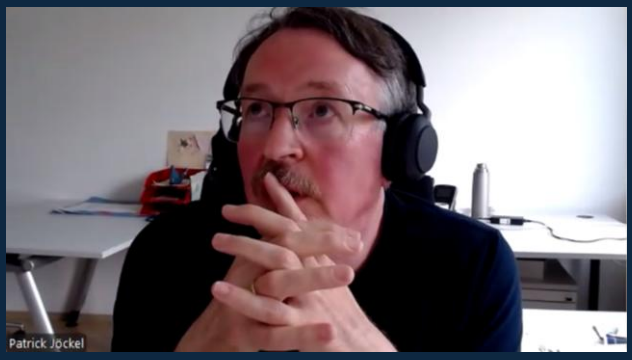
That's why we introduced simple mechanisms like Letters of Intent for new groups. Not because we wanted to control anyone, but because we wanted to connect people. If two groups are working on similar ideas, why not let them collaborate from the beginning? In almost every case, that created added value for everyone involved.

**At this year's EGU, you looked back on more than twenty years of continuous software development. What has been the biggest lesson for you personally?**

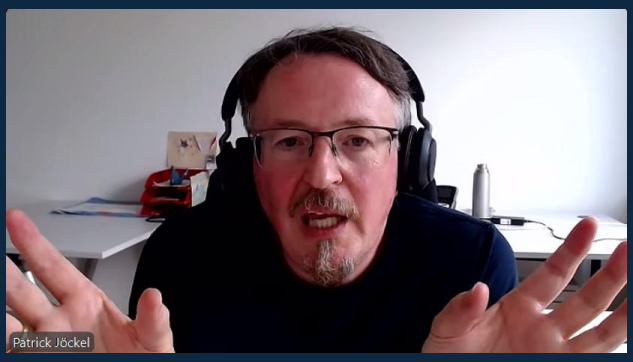
Technically, almost everything is possible. The real challenges lie in human communication. That has probably been my biggest lesson over the past twenty years.

People often assume that the difficult part is writing the code. In my experience, that's rarely true. The difficult part is communication, openness and keeping the bigger picture in mind. Once people trust each other and genuinely want to work together, technical solutions usually follow.

**If that's the biggest lesson, what advice would you give someone starting a scientific software project today?**




Plan first. Code afterwards. It sounds almost too simple, but good software engineering begins long before the first line of code is written.




## Opening MESSy

**MESSy is now preparing for its open-source release after more than twenty years of development. How does that feel?**

Perhaps a little bit like watching your children leave home. You spend decades building something, shaping it, taking care of it—and then, suddenly, it starts its own life.

Of course, preparing for open source also means going through the code one more time. We reviewed licenses, cleaned up the infrastructure, harmonized coding conventions and even took the opportunity to further improve the framework itself.

**» Preparing MESSy for open source feels a little bit like watching your children leave home. «**

But the technical work isn't what I'm most curious about. I'm looking forward to seeing what other people will do with it. I hope new institutions will join, that more people will contribute, and that the community will increasingly help itself instead of relying on a handful of core developers. That would be the nicest outcome.

## Building software together

**MESSy has completed three natESM sprints over the past years. What difference did that collaboration make for your project?**

It made quite a difference. One challenge throughout the entire history of MESSy has been that the software itself was never directly funded. We always developed it alongside scientific projects. Of course, the scientific work was funded, but the software infrastructure largely evolved "on the side".

That means there are always important software-engineering tasks that are difficult to justify within a scientific project—cleaning up code, improving interfaces, modernizing the infrastructure or preparing an open-source release.

The natESM sprints gave us dedicated time to work on exactly these kinds of tasks. Things that had been on our to-do list for years suddenly became possible because software engineering itself became the focus. I think that kind of support is extremely valuable for scientific communities.

## From MESSy to ComIn

**Today, many people associate your name not only with MESSy, but also with ComIn. How did that development come about?**

It actually started with a familiar problem. When we began working with ICON, we realized that we were facing almost exactly the same situation we had encountered twenty years earlier with ECHAM. The model was evolving rapidly, interfaces kept changing, and anyone extending the model constantly had to adapt their own code.

As more and more communities encountered the same challenges, colleagues from the German Weather Service (DWD) approached us and asked whether we would be interested in developing such an interface together. Of course we said yes.


By then, we had accumulated almost twenty years of experience with MESSy. Many of the design principles we had developed there—clean interfaces, modularity and independence from the core model—naturally found their way into ComIn. I should also say that the real implementation was very much a team effort. Florian Prill at DWD and I shared the project leadership, while most of the actual development work at our institute was carried out by Kerstin Hartung and Bastian Kern together with colleagues from DWD, the German Climate Computing Center (DKRZ), and the Jülich Research Center (FZJ). Today, they know the framework far better than I do and are really the experts behind its ongoing development.

That's exactly how I think scientific software should evolve. Projects become successful when knowledge spreads beyond a handful of people. Today, it's exciting to see how widely ComIn is being adopted. The number of plugins is growing rapidly, and I think many people recognize that this kind of interface has become a real game changer.

## One final question

**If you could go back and give one piece of advice to the young Patrick who had just started his PhD in Mainz, what would it be?**

Don't take things personally.

When you're just starting out, you tend to assume that criticism is directed at you as a person. Over time, I realized that's usually not the case. People have different opinions. They argue because they care about getting things right—not because they want to attack each other. Once you understand that, collaboration becomes much easier.

**» Don’t take things personally. «**


*In 2005, Patrick Jöckel (left) and Rolf Sander (right) received the Heinz Billing Award for the Advancement of Scientific Computation from Prof. Kurt Kremer.*
*A milestone early in MESSy's development.*

*Source: GWDG, Research and Scientific Computing, Report No. 69.*

## From the community

The Max Planck Institute for Marine Microbiology is currently recruiting three Scientific Programmers for its Ocean Biosphere Dynamics Department.

👉 View the vacancies.

# Sprint status

| Sprint title | Inst. | Service description |
|---|---|---|
| **ICON-ART** | KIT | Analysis of ART code for GPU porting → Sprint report |
| **ICON-mHM-YAC** | UFZ | Online coupling mHM into ICON using YAC → Sprint report |
| **FESOM** | AWI | Port FESOM 2.1 to JUWELS booster and Levante-GPU → Sprint report |
| **ParFlow** | FZJ | Port ParFlow to AMD GPUs, Performance Analysis → Sprint report |
| **MESSy** | FZJ | Optimize data transfers between host and device → Sprint report |
| **ESMValTool** | DLR-PA | Updating remaining non-lazy preprocessor functions → Sprint report |
| **HAMOCC** | MPI-M | Concurrent HAMOCC on GPU → Sprint report |
| **MESSy-ComIn** | DLR-PA | Couple MESSy to ICON via ICON Community Interface → Sprint report |
| **LAGOOn** | FZJ | Develop concept of Lagr.-transport-modeling framework → Sprint report |
| **IQ** | MPI-BGC | Stepwise port of IQ code to GPUs → Sprint report |
| **modLSMcoup** | FZJ | Develop proof-of-concept for modular coupling → Sprint report |
| **CLEO** | MPI-M | Coupling CLEO to ICON with YAC → Sprint report |
| **PALM** | Uni Hannover | Porting PALM modules related to urban processes to GPUs → Sprint report |
| **MESSy-ComIn2** | DLR-PA | ComIn integration time loop → Sprint report |
| **PDAF2GPU** | AWI | Porting PDAF to GPUs → Sprint report |
| **PISM-AsyncIO** | MPI-GEA | Resolve the issue with the I/O library for asynchronous output → Sprint report |
| **CLEO2** | MPI-M | Uniting CLEO's domain decomposition with two-way coupling to ICON → Sprint report |
| **ICON-XPP** | DWD | Optimization of ICON-XPP for DWD NEC Aurora vector computer → Sprint report |
| **FESOM-C** | AWI | Performance sprint (profiling, analysis, optimization roadmap) → Sprint report |
| **WAM** | Hereon | Full NetCDF I/O and performance optimization |
| **MESSy IMPORT** | FZJ | Revise data-import function of MESSy for ICON/MESSy |
| **COFARE** | AWI | Coupling FABM and REcoM3 |
| **HAMOCC2** | MPI-MM | Consolidate and stabilize HAMOCC on GPUs |
| **ICON-Land** | MPI-BGC | Document readiness of ICON-Land for natESM system |
| **ARTEX1** | KIT | Prepare technical foundation for a future-proof ART |
| **FESOM-C_2** | AWI | Implementation of performance-optimization strategy developed during Sprint 1 |

On hold · Running · Reporting · Finished

Additional information from the sprints, beyond what is covered in the sprint reports, is available in our GitLab wiki.