Reproducibility in Modeling

Methods implemented in the ABM Simulation Framework FAME

Christoph Schimeczek[1,]*, Felix Nitsch[2]

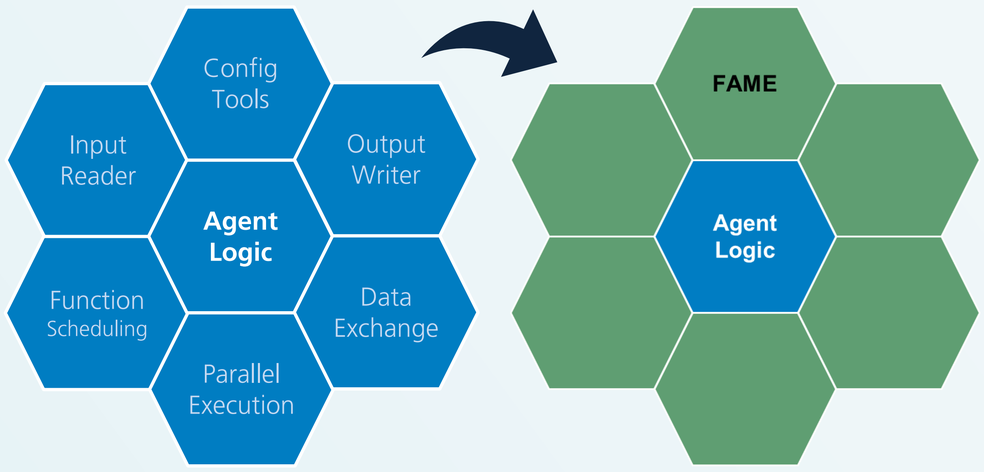
[1]German Aerospace Center (DLR), Institute of Networked Energy Systems, Curiestr. 4, 70569 Stuttgart, Germany
[2]BOKU University, Institute of Sustainable Economic Development, Feistmantelstr. 4, 1180 Vienna, Austria
* Christoph.Schimeczek@dlr.de, fame@dlr.de

deRSE26
March 3rd – 5th 2026, Stuttgart, Germany

FAME

*open **F**ramework for **A**gent-based **M**odelling of **E**nergy systems*

Aim: Reduce overhead code



- Support **model development** of energy-related Agent-Based Models (ABM)
- Provide **fast execution** of models at low overhead
- Assist **all project stages**: from rapid prototypes to large & complex models
- Grant **scalability**: from laptop to High-Performance Computing (HPC) cluster

In a nutshell

Write less code

FAME organises
– Data exchange
– Order execution
– Inputs & Outputs
– Parallelisation

Configure, not code

FAME offers flexible
– Input
– Output
– Agent interactions

Scale to your needs

FAME is portable
– Windows / Linux / Mac
– Laptop / Server / HPC

No hidden costs

FAME is open
– Apache 2.0
– REUSE complaint

Full model support

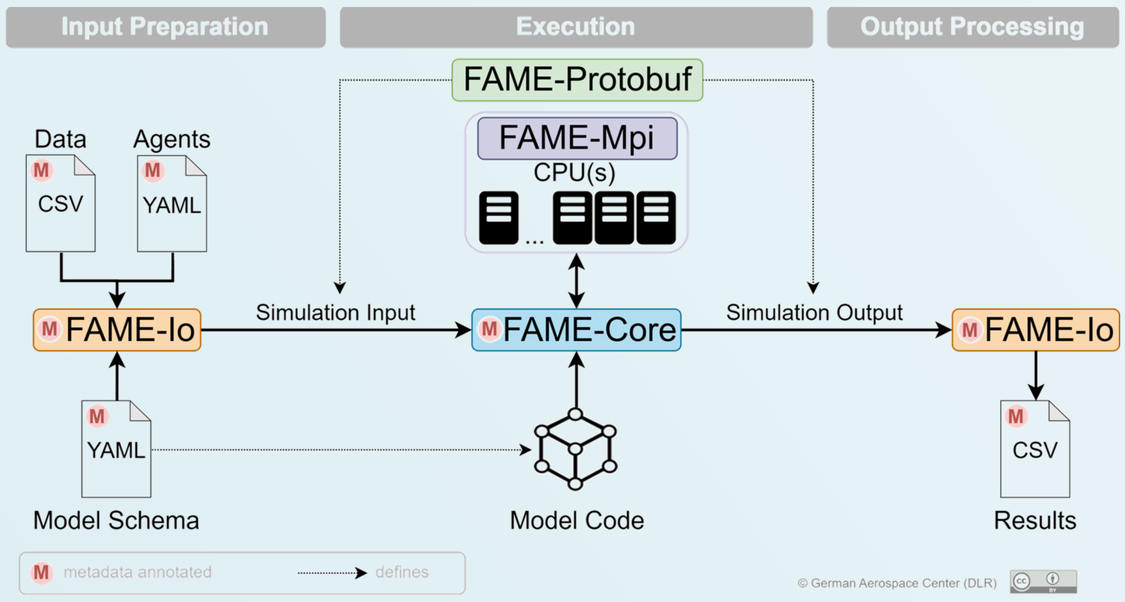
FAME helps developers/users
– Coding models
– Configure scenarios
– Run simulations

https://gitlab.com/fame-framework

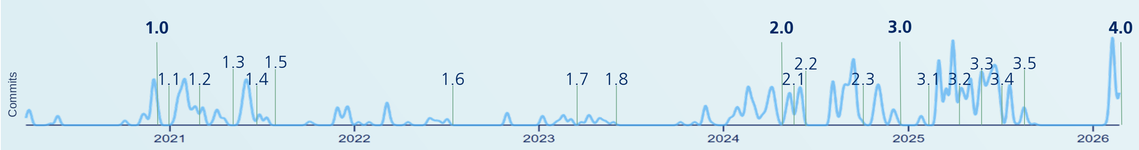
Components & workflow

FAME is divided into several components, each addressing a specific task:

FAME-Core: Provides methods to create & run agent-based simulations (Java)
FAME-Io: Feeds input data to & extracts results from simulations (Python)
FAME-Mpi: Coordinates processes in multi-core mode using MPI (C++)
FAME-Protobuf: Defines file formats & message types (Google Protocol Buffers)
FAME-Gui: Drag & drop configuration of FAME-based models (Python)



Activity: FAME-Io



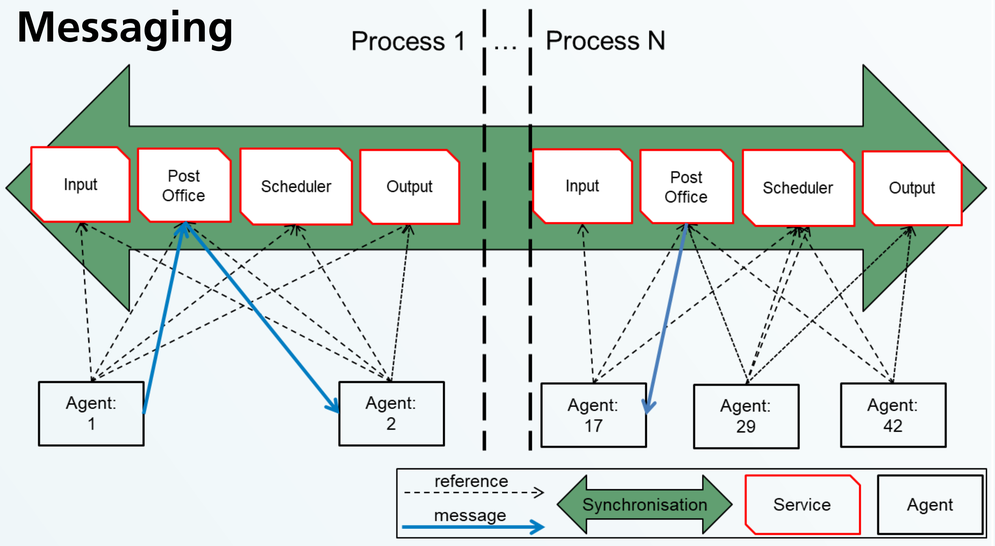
1191 commits | Last commit ≈ **2 days ago**

Concurrency

Basic idea

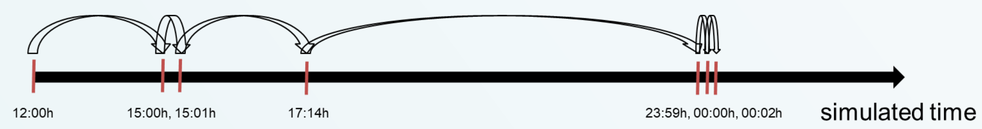
Zero knowledge: Modellers do not need to consider parallelisation
Parallelisable by default: every FAME model can be run concurrently

Messaging



Services: Each process has a set of Services that connect to local *Agents*
Synchronisation: *Services* update each other every tick
Agents: Have a unique ID; *Agents* don't share any objects
Messages: Are used to exchange data between *Agents*; 1 tick delay

Scheduling



Agents: Ask *Scheduler* to schedule *Actions* at specific simulated *Time*
Scheduler: Determines *Time* of next tick; calls *Actions* of non-idle *Agents*
Actions: Methods of *Agents* executed at specific *Time*
Simulated time: Non-continuous timestamp chain; 1s minimal resolution
Contracts: Define repeated *Actions* and *Message* deliveries

Result Reproducibility

Random number generation

Dedicated service: Provides random number generators to *Agents*
Seed: Derive seeds from configuration: Main seed + Agent ID + Counter
→ Random numbers are independent of process count involved

Comprehensive outputs

Single output file: includes model input, output, metadata
Log tool versions: of involved FAME-tools and models
Rerun capability: Rerun a simulation using output file as input

Metadata support

Everything: can have a metadata description
Describe: Models, agents, files, inputs, outputs
Free Format: Format of metadata is choice of modeler
Template Processor: Automatically extract metadata and assign to model output files; default: OEO metadata template

Outlook

Go Python: Enable *Agents* to be programmed in Python
Reduce boilerplate: Autogenerate message classes from schema file
Enhance verifications: Check contract consistency during configuration

Use FAME

See example: fully-fledged open energy market simulation AMIRIS: https://helmholtz.software/software/amiris
Ask questions: unsure how to use FAME? Contact us at: FAME@dlr.de
Get started: Create your own FAME model! Start from our example: https://gitlab.com/fame-framework/fame-demo

Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

German Aerospace Center (DLR)
Institute of Networked Energy Systems
Energy Systems Analysis
www.dlr.de/ve