



The Process and Value of Reprogramming a Legacy Global Hydrological Model

Emmanuel Nyenah^{1,2}, Petra Döll^{1,2}, Martina Flörke³, Leon Mühlenbruch³, Lasse Nissen¹, and Robert Reinecke

¹Goethe university frankfurt, Institute of Physical Geography, Hydrology, Frankfurt am Main - Kalbach-Riedberg, Germany (nyenah@em.uni-frankfurt.de)

²Senckenberg Biodiversity and Climate Research Centre (SBIK-F), 60438 Frankfurt am Main, Germany

³Institute of Engineering Hydrology and Water Resources Management, Ruhr University Bochum, 44801, Bochum, Germany

Global hydrological models (GHMs) have significantly advanced in process representation and spatial resolution over the past four decades. These advancements include the inclusion of reservoirs. However, significant needs and opportunities remain to improve these models, particularly for better representing human-environment interactions and reducing model uncertainties by improved integration of model output observations.

As research questions and GHMs become more complex, maintaining and further developing an existing model code in an efficient manner becomes increasingly challenging. Similar to other complex research software, GHMs are developed by scientists with limited software development training, time, and funding, and thus lack the software quality that is required for a sustainable research software. This includes non-modular design, poor variable naming, suboptimal comment density, and a lack of testing frameworks. The sustainability of GHMs could be significantly enhanced by reprogramming them using modern best practices.

While global models such as HydroPy and CLASSIC (a global land surface model) have been reprogrammed, publications on the reprogrammed software focus on evaluating model performance. Details in the reprogramming process, from project management to final software release, are missing. Here, we present the process of reprogramming the GHM WaterGAP to a sustainable research software. This involves rewriting WaterGAP from scratch, introducing improvements such as modular architecture, Python programming, version control, open-source licensing, consistent variable naming, comprehensive documentation, and testing, while maintaining good computational performance. We evaluate the reprogrammed WaterGAP code against software sustainability criteria and FAIR4RS principles.

Reprogramming with best practices requires effort but makes the resulting software easier to use, maintain, modify, and extend. With reprogramming WaterGAP, we aim to facilitate joint code development across multiple locations and by various developer groups, thus establishing a community GHM that is easily understood, used and enhanced by novice users.

