Software Specification WaterGAP

ReWaterGAP workshop participants

March 5, 2025

Document version 0.1

Contents

1 Introduction

1.1 Purpose

WaterGAP is a state-of-the-art global-scale water resources and use simulation model. In numerous studies, WaterGAP was used to support sustainable development of the Earth system by assessing water scarcity for humans, drought hazard, ecologically-relevant streamflow characteristics, the impacts of human water use and dam construction as well as freshwater-related scenarios of the future. Here, the focus has been on quantifying the impact of climate change on the global freshwater system, including the streamflow regime, groundwater recharge, floods and droughts. The existing WaterGAP software consists of five independent water use models that compute time series of consumptive water use and water withdrawals (abstractions) for the water use sectors irrigation, livestock, households, manufacturing and cooling of thermal power plant (note that the irrigation model GIM only computes consumptive water use). The water use time series are input to a submodel GWSWUSE that computes, for each sector, time series of sectoral consumptive use and gross abstractions from groundwater and from surface water, and, simulating return flows, time series of net abstractions from groundwater and surface water bodies. Both net abstractions are input to the WaterGAP Global Hydrology Model WGHM that computes, with a daily time step, water flows and storages (or storage anomalies for some compartments) on the continents, driven by meteorological variables. Flows include actual evapotranspiration, total runoff, groundwater recharge and streamflow. Ten water storage compartments are differentiated, including snow, soil, groundwater and diverse surface water bodies. Water storages and flows related to glaciers are not simulated, but output of an external glacier model has been integrated into WGHM and can be taken into account optionally (in WaterGAP 2.2e only). Depending mainly on available meteorological input, WaterGAP generally covers the time period 1901-2100.

WaterGAP covers all land areas of the globe except Antarctica. Currently, there exist two model variants, WaterGAP 2 with a spatial resolution of 0.5° latitude by 0.5° longitude and WaterGAP 3 with a spatial resolution of 5' (arc-minute), with partly different algorithms and parameters. For a detailled description of WaterGAP model version 2.2d, please refer to Müller Schmied, H. et al. (2021): The global water resources and use model WaterGAP v2.2d: Model description and evaluation. Geosci. Model Dev., 14, 1037–1079. doi: 10.5194/gmd-14-1037-2021.

The purpose of this software specification is to support the re-programming of WaterGAP 2.2e (publication in preparation) excluding the five water use models. The output files of the five water use models will be read in by the new WaterGAP software, similar to how they are read into GWSWUSE in WaterGAP 2.2e.

The water use models will not be re-programmed but will be made open source (except the thermal power plant water use model as it includes proprietary data). However, the documentation of the water use models needs to be strongly im-

proved such that they can be run by external researchers. This is particularly relevant for the Global Irrigation Model GIM that needs to be re-run if new climate data are available. There exists a version of GIM for 0.5° using time series of irrigated areas (HID, Siebert et al. 2015) that differentiates only rice/non-rice, and another version of GIM for 5' that does take into account HID but more crops). Not sure if this can be harmonized within the project period.

1.2 Software Scope

The overall project goal is to completely rewrite the software WaterGAP with a modular structure using a modern programming language and providing extensive documentation such that the resulting software is to be testable, maintainable, extensible, and usable while maintaining the current computational performance. It should have a modularity that allows altering the model behaviour in a reduced amount of time. In particular, the new software should enable that new researchers who are not computer scientists and need to analyze, modify and improve the WaterGAP model by adding code spend less time understanding the code and writing new code. This includes researchers inside and outside the core development groups. The software is to be thoroughly tested (100% test coverage). The new code will be open source.

Such a code and its documentation will make it possible for other researchers to run our global hydrological model software by themselves, to reproduce our results or investigate the impact of data and algorithm modifications on the results. By enabling researchers from other groups to investigate the internal structure of the code, the reprogrammed research software is to improve the comparability to other model results and structures and to enable the research community to check the consistency and accuracy of our computational approach and find possible errors in the software more easily.

For a quick and reliable reporting of such errors we intent to make use of established platforms like github, and established software engineering techniques like automated testing and an agile development process combined with benchmark scenarios. This will not only enable others to use the research software more efficiently and make scientific results more reliable; it will allow WaterGAP, which is one of only two German global hydrological models, to remain one of the best global hydrological models world-wide.

1.2.1 Software architecture

The software consists of two "layers" (like in case of G3M and PCR-GLOBWB):

1. Framework: program as flexible software to simulate hydrological processes

2. Specific model with selected spatial resolution and all input data that can produce output

1.2.2 Programming language

Python, using C-libraries. Checks should be done later if certain parts need to be translated to C to achieve faster run times.

Python is more widely known among hydrologists and easier to write and read, C++ is faster due to available compilers. There are many python libraries the are written in C++, e.g. to handle NetCDF files. For ensemble runs (e.g. 20,000), Python code can be run without problems in an HPC environment. Also the coupled WaterGAP-G3M should run fast enough with Python.

1.2.3 Major specific objectives for the code

Flexible spatial resolution. The code should allow to flexibly change the spatial resolution of the model which is currently not possible. Minimum requirement for project: The model user should be able to select either a 5 arc-min or a 0.5 degree (30 arc-min) spatial resolution, by reading in alternative land masks, but the coding should enable other spatial resolutions to be implemented later. This should enable any code modifications done for one spatial resolution to be usable at the other resolution. This is not the case now when we have separate codes for 0.5° (WaterGAP 2) and 5' (WaterGAP 3).

There is a need to consider

- which algorithms cannot be the same at the two resolutions (e.g., lakes and reservoirs?)
- which datasets must differ and are lacking at the two resolutions
- which datasets (e.g., soil, land cover) could be upscaled automatically from 5' to 0.5°? (learn from mHM hydrological model of UFZ?)

Parallelization. Basins or continents can be run in parallel of different processors (already implemented in WaterGAP 3). Problems: Basins that cover more than one continent need to be stitched together in case of "continents". Water abstractions from outside the basins cannot happen in case of "basins".

Re-start. The model runs can be stopped at any day. Then, the model parameters and storages are saved such that they can be read in and manipulated by another software. Then, reading in the modified parameters and storages, the model run continues for another day (or any other time period, e.g. 10 days or one month). Ideally, an online/in-memory coupling with PDAF is possible. PDAF (The Parallel Data Assimilation Framework) is a software environment for ensemble data assimilation; PDAF simplifies the implementation of the data assimilation system with existing numerical models (pdaf.awi.de). Currently, WaterGAP 2.2d can be re-started each month, but some errors still occur despite extensive debugging such that the output of runs with restart are not identical to the output of "continuous" runs.

Coupling to G3M. G3M is a gradient-based global groundwater model with a spatial resolution of 5 arc-min that is in the process of being coupled to the current WaterGAP software (Sebastian Ackermann, Robert Reinecke). The

re-programmed WaterGAP should be able to run 1) in its current form, with groundwater simulated as a linear box model and 2) with the 0.5° WaterGAP coupled to the 5 arc-minute G3M. If time allows, also coupling to 5 arc-min WaterGAP should be enabled.

1.3 References

"List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location."

2 Overall Description

2.1 Terminology

The new WaterGAP software will use a new terminology for its components (Figure 1). WaterGAPCore includes the WaterGAP 2.2e components WGHM and GWSWUSE. WaterGAPUse includes the five sectoral water use models. WaterGAPTools includes preprocessing (input-generation) and standard calibration (against mean annual streamflow).

The name of the new software will be WaterGAP-n or WaterGAP-neo but in publications etc. we will continue to use just WaterGAP. *If we do not re-use any code pieces of the existing WaterGAP (which we will not), there is no legal connection to the old WaterGAP code.*

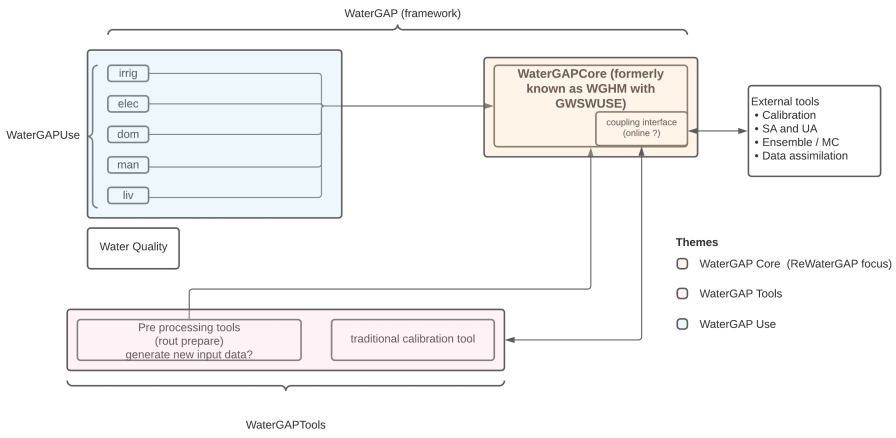

Figure 1: Overview of the WaterGAP components and how they link to the redevelopment themes.

2.2 Product Perspective

See Sect. 1.2

"Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful."

2.3 Product Functions

The major function of the software is to generate (monthly or daily) time series of various (vertical and lateral) water flows and of water storage in various compartments as driven by climate forcings, human water use and the existence of man-made reservoirs. The time series are generated for each grid cell, which is part of a drainage basin. These time series depend on the selection of model parameters that are adjustable by model calibration to observations of model output variable. In an alternative model variant, the gradient-based groundwater model G3M is to be coupled to WaterGAP, with additional outputs of time series of groundwater table elevation and surface water table elevation as well as exchange flows between groundwater and surface water bodies and capillary rise.

Figure 2 provides a schematic of the temporal sequence of computations in WGHM, as described verbally below.

Current WGHM calculation structure (*written by PD in workshop*)

- Initialize storages of lakes, wetlands to maximum, groundwater to zero, plus other storages (or prescribed) (WaterGAP.cpp)
- Read inputs
- Annual initialization: new reservoirs or regulated lake, unsatisfied use, glacier area, resulting in updated land area fraction and storages (vertical storages canopy, snow, soil) (in routing.cpp)
- GLWD units read in every year
- Monthly water use (NAg and NAs) distributed to daily values (in routing.cpp) and aggregating to GLWD units
- Simulation of consecutive days (or smaller time steps); if the day is a certain day, e.g., the first of a year or a month, then further things happen, like reading in climate data, initializing new reservoirs, etc. (Yearly loop, monthly loop, daily loop, loop over grid cells)

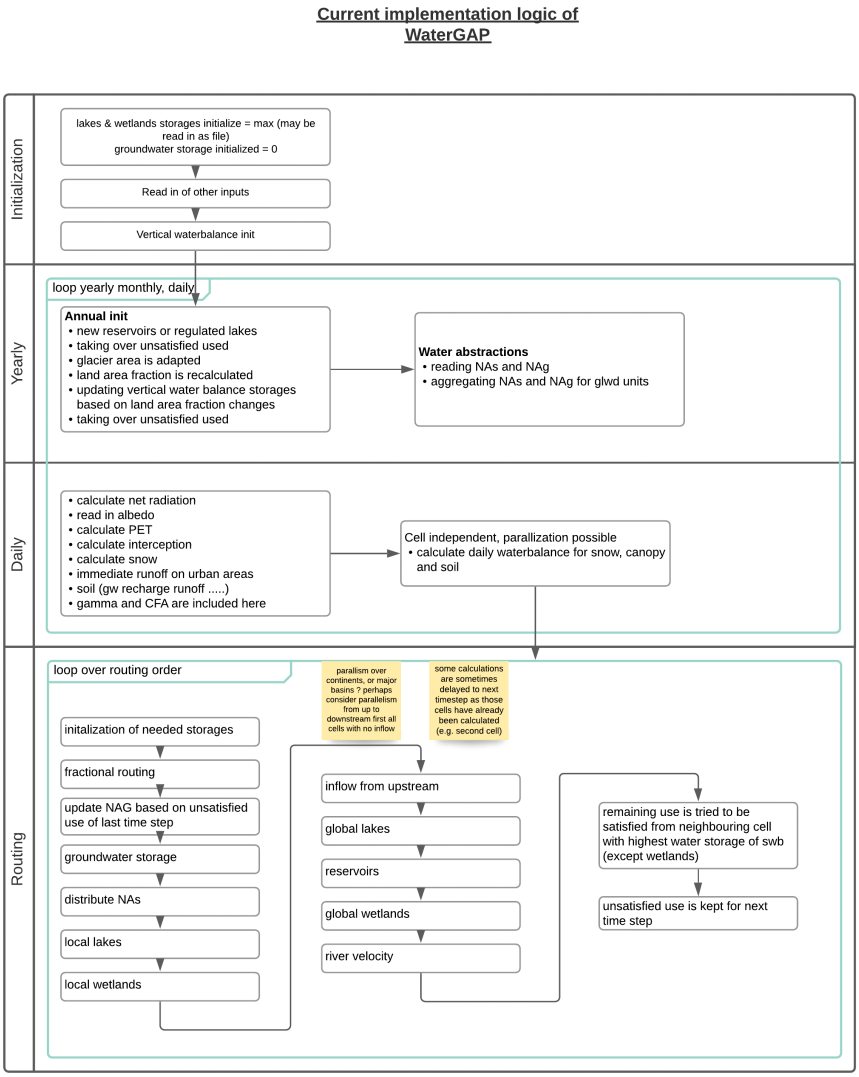

Figure 2: Overview of the current logical structure of WGHM

- For one day and all grid cells independently (Vertical water balance): in the sequence of grid cells Compute PET, 1 canopy, 2 snow, 3 immediate runoff for buildup areas 4 soil (output surface runoff and groundwater recharge). Include gamma and areal cell correction factor (to modify AET and runoff). (could be done in 5-7 parallel pieces) (daily)
- Lateral water balance in routing; could be parallelized per basin if there

is no second cell option for water use, or per continent. Or run all cells without inflow in parallel, the those with just one inflow, etc.

- Initialize, use routing order of cells from upstream to downstream (route.prepare computes routing order if necessary but also defines things relevant for calibration)
- Do various calculations on fractional routing, distinguishing inland sink etc.
- Compute gw balance (including NAg) Adjust NAg based on unsatisfied NAs of last time step.
- Distribute NAs over GLWD unit grid cells
- Compute water balances of 1) local lakes, 2) local wetlands, get inflow from upstream, 3) global lakes, 4) global reservoirs, 5) rivers (compute river velocity before). Order of NAs: global reservoirs/global lakes, rivers, local lakes, Update of reduction factor of maximum surface water body area (lakes, wetlands, reservoirs) in each water balance.
- Subtract remaining use from second cell with highest total sum of river, global lake, local lake and reservoir storage.
- Unsatisfied use is kept for next time steps
- Updating of land area fraction
- Writing out flows and storages

2.4 User Classes and Characteristics

- Users that do not want to change the code but are interested in its detailed functionality (G1)
- Users that actively develop the code (G2)
- Users that would like to change small parts of the code (G3)
- Users that use the model as a black-box, e.g. to do calibration and data assimilation (G4)

Requirements of these groups are described in sections 4, **??** and **??**.

2.5 Operating Environment

"Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist."

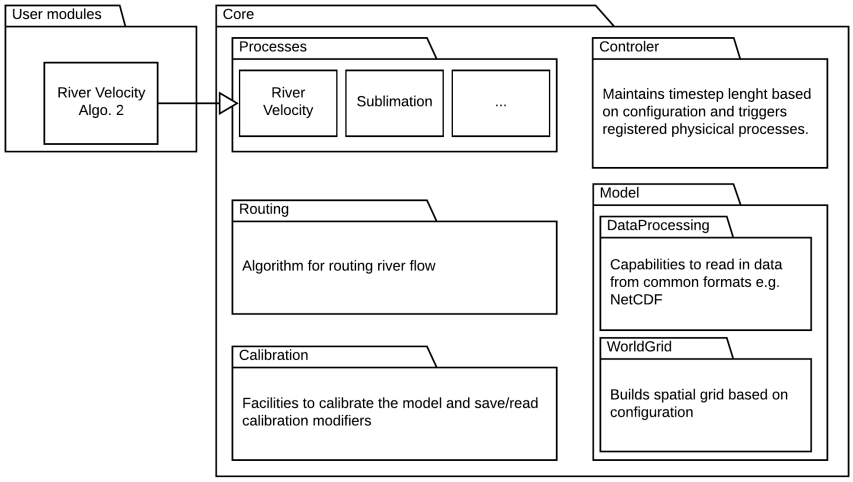

Figure 3: Possible package structure as outlined in the project proposal for DFG.

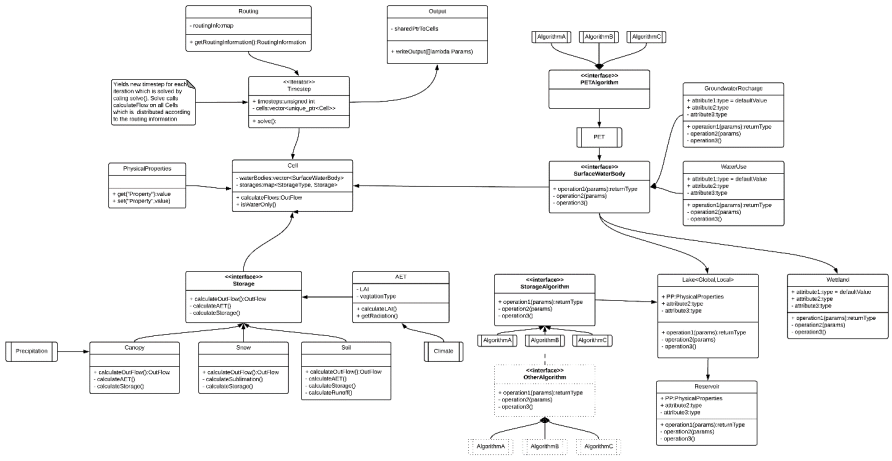

Figure 4: Possible architecture of a new WaterGAP.

2.6 Design and Implementation Constraints

"Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design

conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software)."

How to do Sprints in Re-WaterGAP

- Product owners: Martina, Petra
- Scrum Master: Robert
- Development team: Emmanuel, Jenny, Tim, Hannes, (Sebastian, Mohammad)
- One sprint takes one month.
- Every first Monday: Sprint review and planning meeting (Emmanuel, Martina, Petra)
- (Start in third month; in the first two months, Robert will get Emmanuel up to speed in software design)
- Every week (team in WaterGAP dev meeting)
- Sprint retrospective meeting (team): once per month

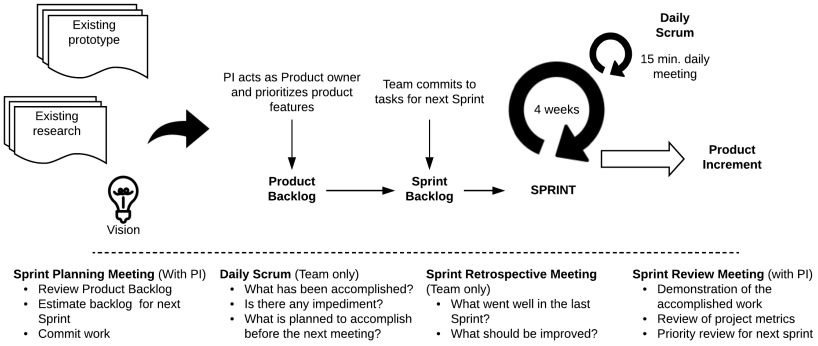

Figure 5: Outline of the development process during the reimplementation.

2.7 User Documentation

"List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards"

2.8 Assumptions and Dependencies

"List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan)."

3 External Interface Requirements

With "interface" the following refers to an outside connection or interaction with a user (User Interfaces) or a program (Software Interfaces).

3.1 User Interfaces

The user interacts with WaterGAP through a comandline interface (CLI), log files, and multiple configurations files that can be used to change in- and outputs of the model.

3.1.1 CLI

WaterGAP has no graphical user interface (GUI). A later implementation of such an interface should be possible but is not in the scope of this project. Interaction with the program are to follow well established standards e.g., `https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap12.html`. Arguments are to be parsed by using best practises e.g., through the *argparse* library. CLI arguments can be extended easily and are documented in the user manual.

The CLI will be designed in a way to also facilitate automated executions of WaterGAP through scripts and other programs. It is possible to start the program with a non-verbose flag that writes all output to a logfile to facilitate this use case.

3.1.2 Logging

During the execution of the program the user is informed about the general status of the program such as:

- Start of the simulation
- Current time frame of the simulation
- Criticals and Errors (see below)
- Warnings and Debug information if debug flag was used

- End of the simulation and quality metrics that summarize the simulation e.g., time to compute, water balance

This information shall be short and precise. All other information is written to a logfile. All messages inside the program are separated into five standard categories:

- DEBUG Detailed information, typically of interest only when diagnosing problems.
- INFO Confirmation that things are working as expected.
- WARNING An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.
- ERROR Due to a more serious problem, the software has not been able to perform some function.
- CRITICAL A serious error, indicating that the program itself may be unable to continue running.

The category of each level is noted in the log file together with a timestamp. Each logfile name automatically is named with the simulation date and version of WaterGAP. It should also summarize the inputs and parameters that were used.

3.1.3 Configuration

Additionally to the CLI, the user can interact with the program through configuration files. The following should be configurable via configuration files:

- Paths to input data (meteorological forcing, parametrization and model files (input+routing, parameters.json))
- Runtime options (simulation options, simulation period, timestep configuration)
- Output variables and files

The configuration files should be in json format, as they are configurable by human in text editors and are support by a wide range of libraries in established programming and scripting languages. Some configurations can not be combined with others, such conflicts should be checked either in WaterGAPCore but better in the configuration files themselves, possibly through a json schema `https://json-schema.org/`. Ann additional advantage of json files is the flexible integration of these files into a web application or GUI.

3.2 Software Interfaces

Software interfaces describe how the program interacts with other tools, libraries or databases. Since WaterGAP is a scientific software its main software interfaces are related to scientific tasks such as (1) coupling to another model, (2) sensitivity analysis (SA), (3) calibration, (4) and data assimilation.

In the following we distinguish between *online* and *offline* coupling. *Offline* coupling (or calibration etc.) describes the process in which the model is connected to another model or script in a fashion that does not integrate the two into one executable program but rather necessitates a control script that executes the model and then transfers information from one component to another. For example, a script could define an objective function that is set out to be optimized by varying input parameters to WaterGAP - a calibration scheme. This script executes WaterGAP, reads the outputs, calculates the objective function and then modifies the parameters.

In comparison, *Online* coupling (or calibration etc.) refers to an in-memory execution of the whole system as one. Using the example of a calibration scheme that means that the calibration includes WaterGAP as a library and is able to read and write properties (its in and outputs for example) of the model in memory which can be much more efficient than writing output data to a file first. The same mechanism can then be used to couple WaterGAP, for example, with a crop model in an efficient way.

The new WaterGAP program is offering a well-defined interface of abstract methods that need to be implemented by a possible coupling script/program to properly use WaterGAP as a library like module. See `https://realpython.com/python-interface/` as an example.

Model coupling, sensitivity analysis, model calibration, and data assimilation are similar and should be made possible through the same interface for online coupling.

Data in Any possible data that WaterGAP can receive as inputs through files and user input (including configurations) can also be provided through the coupling interface.

Data out Any possible data that WaterGAP can write as output can be accessed through the coupling interface instead.

3.2.1 Model coupling

Purpose Model coupling should allow to couple WaterGAP to any other scientifically meaningful model in a online or offline fashion. Coupling means that outputs of one model are used as inputs for another model and possibly, but not necessarily, vise-versa. WaterGAP should be able to be either the controlling model that uses a coupled model as additional module or be used as a library like module by another model. It provides comprehensible and reusable facilities for both use-cases.

3.2.2 Sensitivity Analysis

Purpose Sensitivity analysis refers to the systematic variation of model parameters (model inputs, model resolution, timesteps, constants etc.) to investigate the impact on the model output. The model is agnostic to whichever method is used to achieve that.

3.2.3 Model calibration

Purpose By defining one or multiple objective functions the model can be calibrated by using multiple in and outputs. WaterGAP is shipped with its currently used calibration method as default but it is possible to implement other calibration schemes in a flexible way through the coupling interface.

3.2.4 Data assimilation

Purpose Similar to calibration, data assimilation allows to optimize the output of the model during simulation time by modifying parameters of the model during runtime. If only an offline coupling is possible, WaterGAP offers the possibility to be "restarted" at any given simulation time. That means the model can start from any arbitrary point in time, if the necessary input data is available.

4 Functional Requirements

The functional requirements of WaterGAP are driven by the four user groups outlined at the beginning of this document:

- Users that do not want to change the code but are interested in its detailed functionality (G1)
- Users that actively develop the code (G2)
- Users that would like to change small parts of the code (G3)
- Users that use the model as a black-box (G4)

The following summarizes the main requirements for these user groups for each of the main WaterGAP components (these directly link to the epics in the user stories):

- WaterGAPCore
- WaterGAPTools
- WaterGAPUSe

4.1 WaterGAPCore

This is the core application of the modeling framework. The hydrologic simulation software that computes fluxes and stores of the hydrologic cycle.

4.1.1 R1 Run the calibrated WaterGAP as a prescribed variant (ant, nat)

UserGroup: G1, G4
The user just needs to make small changes to the options file. It is not necessary to recompile the application.

4.1.2 IO RX the input of WaterGAPCore is interchangeable within prescribed formats

Input to WaterGAPCore like climatic or static input can be exchanged.

4.1.3 CSA R3 Interface for calibration and sensitivity analysis

UserGroup: G1, G4 WaterGAPTools and user should be able to modify model parameters in order to calibrate WaterGAP and run sensitivity analysis.

4.1.4 R4 Data assimilation is possible

UserGroup: G1
Target is mainly the program PDAF, but in principle it should be possible with different assimilation approaches. Requires capability to restart every month or any day, e.g. produce daily snapshots and restart with updated parameters and storages, with online coupling/no reading in and out files

4.1.5 IO R5 Change the gridded static input data

UserGroup: G1
The user should be able to exchange several files or change files in already existing files, e.g. DEM, DDM, Soil Information, ... Therefore WaterGAPCore should be programmed in a robust and dynamic manner (no/low hard-coding) and checks should be implemented to check if input data is correct (value ranges, No Data, ...). Note that metadata of files is mandatory so the users can change files or create files in a reasonable way (so WaterGAPCore can use them.)

4.1.6 R6 Recalibrate WaterGAP in the standard way based on different climate input

UserGroup: G1
TODO

4.1.7 R7 Run the model for specific basins for which different data for for soil texture, climate data etc. is available and calibrate

UserGroup: G1, G4
TODO

4.1.8 R8 It is easy to replicate/reproduce previous model results

UserGroup: G1, G4
Model versions need to be linked directly with specific commits in the open repository using tags. The code and configuration file setup should facilitate the replication of results. See also R14 checks.

4.1.9 R9 Run an ensemble of models

UserGroup: G1
It should be as easy as possible to automatically run a multitude of WaterGAP applications. Specifically, it should be easy to run the application in a cluster environment with shared storage.

4.1.10 R10 Run the model in operational mode with new climate input on selected days

UserGroup: G1, G4
This should enabling operational runs with new climate input and automated pre/post-processing scripts for e.g. every 5. of the month or every five days.

4.1.11 R11 Read in a different drainage direction map

UserGroup: G1
At either 0.5° or 5 arc-min. Users have to be warned that other files like reservoirs files need to be adapted. This may a check if the standard input files have changed. Possibly a flag is necessary that allows for checking md5 sum or similar for all inputs.

4.1.12 I/O RX Read in and write out NetCDF files

UserGroup: G1
netcdf should be the standard format for spatio-temporal input and output.

4.1.13 R13 Calibration the fraction of (sectoral) water withdrawals from groundwater/surface water

UserGroup: G2
TODO This requires integration of GWSWUSE into WHGM.

4.1.14 R14 Run WaterGAP at 5 arc-min and 0.5 degree spatial resolution

UserGroup: G2, G3
Links to R14. It should be possible in principle to implement other spatial resolutions. Focus on 5 arc-min and 0.5 degree. Requires reading in of different land masks and flow direction maps. Ideally prepare input data automatically for new drainage direction maps at different spatial resolutions.

4.1.15 R15 Possible to implement dynamic land cover

UserGroup: G2, G3
TODO (e.g. HYDE data) (requires integration of GWSWUSE into WHGM) low priority

4.1.16 R16 Either simulate groundwater flow with a linear box model or with WaterGAP (,5 degree or 5 min) coupled to 5 arc-min 3GM (online coupling)

UserGroup: G2, G3
TODO

4.1.17 R17 Simulate the Gregorian calendar (including leap years)

UserGroup: G2, G3
The user should simulate the actual time period, so 29.02. is simulated if there is a leap year. So number of days in a month should not be hard-coded. Instead the model should use the Gregorian calendar.

4.1.18 R18 Account for different model structures (processes, algorithms) per grid cell (e.g. arid or humid, or basin-wide)

UserGroup: G2, G3
TODO flexible process structure (e.g. define specific processes/algorithms for specific regions)

4.1.19 R19 It is possible to add an alternative algorithm for a specific process and enable selection of different defaults

UserGroup: G2
TODO

4.1.20 R20 Artificial water transfers from grid cell to grid cell

UserGroup: G2
TODO

4.1.21 R21 Prepare for an implementation of a floodplain inundation

UserGroup: G2
TODO low priority

4.1.22 R22 Prepare for higher temporal resolution than daily

UserGroup: G2
TODO low priority

4.1.23 R23 Possible to implement new lakes dataset (Lehner et al.)

UserGroup: G2
TODO

4.1.24 R24 Modify reservoir algorithm so that it includes a calibration parameter

UserGroup: G2
TODO

4.1.25 R25 Check that area types in one grid cell add up to 100%. Define other essentials for consistency check without making consistency checks too run-time consuming.

UserGroup: G2
TODO

4.1.26 R26 Analyze log file, and user selects what level of information s/he wants to have on the screen and what checks to do

UserGroup: G2
TODO

4.1.27 R27 Run in parallel on a cluster for basins/continents

UserGroup: G2
TODO

4.1.28 R28 Determine actual water use per sector (using allocation rule) (possible as we include GWSUSE in WGHM)

UserGroup: G2
TODO

4.1.29 R29 Run WaterGAP in calibration mode, with a script

UserGroup: G2
TODO

4.1.30 R30 Run (and calibrate) for a specific basin with modified input

UserGroup: G4
Links to R9. TODO

4.1.31 R31 modify (by calibration) fraction of "sectoral" water withdrawals from groundwater / surface water GWSWUSE

UserGroup: G3
TODO

4.1.32 R32 add a specific equation / process and to be able to select from different options

UserGroup: G3
TODO

4.1.33 IO CSA R33 restart from any daily snapshot or prescribed state

WaterGAPCore is able to start from prescribed state. For that it must be able to read and write out the state at any specified date. **UserGroup: G3**

4.1.34 IO CSA RX enable interface to PDAF

It is necessary to have a interface to PDAF in a "online-coupling".

4.1.35 R34 artificial water transfer

UserGroup: G3
TODO see R20

4.1.36 R35 consider desalinization

UserGroup: G3
TODO low priority

4.1.37 R36 smart logging

UserGroup: G3
TODO

4.1.38 R37 consistency checks

UserGroup: G3
TODO This should include physical unit checking as well.

4.2 WaterGAPTools

4.2.1 R1 Vary selected model parameters from range of predefined calibration parameters

UserGroup: G1
The user needs to be able to change the parameter file adding/deleting predefined parameter and changing/defining parameter spaces (min/max). (A warning should be displayed which files are parameter-dependent, e.g. bankfull flow, and which options are recommended to avoid the usage of parameter-dependent files, e.g. constant flow velocity, reservoirs as global lakes, ...)

4.2.2 R2 Vary selected model parameters for a sensitivity analysis

UserGroup: G1, G4
The user needs to be able to change the parameter file adding/deleting parameter and changing/defining parameter spaces (min/max). (A warning should be displayed which files are parameter-dependent, e.g. bankfull flow, and which options are recommended to avoid the usage of parameter-dependent files, e.g. constant flow velocity, reservoirs as global lakes, ...)

4.2.3 R3 Calibrate the models with different parameters and approaches

UserGroup: G1
The user needs to be able to change the parameter file adding/deleting parameter and changing/defining parameter spaces (min/max). (A warning should be displayed which files are parameter-dependent, e.g. bankfull flow, and which options and steps are recommended to avoid the incorrect usage of parameter-dependent files)

4.3 WaterGAPUse

We have not yet defined any requirements here.

4.4 Additional features - overarching stories

We have not yet defined any additional requirements.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

The runtime needs to be small enough to use WaterGAP enable the following use cases:

- Use a virtual laboratory to run experiments, e.g., change equations, data etc.
- Calibrate the model in a reasonable time (a week with the standard approach)
- Run Sensitivity Analysis and Data Assimilation in reasonable time-frames (less than a month)

Currently the model takes no longer than 4 minutes (closer to 3) per year in standard mode on our computing units (ipg 80 or 50). The memory footprint is around 1-2 GB. The future footprint should be lower than 8 GB. Target system platform: Linux.

5.2 Software Quality Attributes

See also proposal document for metrics and details.

5.2.1 Adaptability

Every software system changes over its lifetime. Some of the changes are predictable, and systems can be designed to be robust to such changes by considering those future changes beforehand. However, it is impossible to predict all the future changes and possible concerns. Anticipating the various concerns is hard due to the diversity in client requirements and the rapid advances in the enabling technologies. Because unanticipated changes often require many parts of the system to be modified or redesigned, they are very costly most of the time. Therefore, it is necessary to engineer adaptability into software systems in order to meet various future requirements. Text from `https://doi.org/10.1117/12.434864`

WaterGAP as a research software needs to take into account possible future developments. The list of requirements already lines out some of these developments.

5.2.2 Correctness

The software needs to behave according to the scientific publications in which it has been described.

5.2.3 Maintainability

See Reusability. Maintainability and Reusability are understood here as one attribute that is to be achieved.

5.2.4 Reusability

The implementation of new features, e.g., algorithms needs to be as easy as possible. Change is embraced as part of the system design.

5.2.5 Testability

All algorithms and computations within the simulation software need to be testable.

6 Other Requirements

6.1 Legal requirements

The software will be licensed as OpenSource from the start of development.

6.2 Community requirements

Code and documentation will be made available to the public already during the development process. The project is guides by best practices for Open Source projects. See also project proposal.

7 Epics and User stories

"This section summarizes all currently known user stories (US). USs are combined into epics (E) that describe a bigger common goal that its containing USs are pursuing. E in turn are summarized in themes (T) that describe a high level goal or software component. Ts are not altered during the development process and describe a common vision of the product owners."

Themes

- WaterGAPCore T1
- WaterGAPUse T2
- WaterGAPTools T3
- Overarching stories T4

"The development phase of the DFG project ReWaterGAP mainly focuses on T1. Some Es and USs may outline future wishes that are important for the design of the current software architecture, these are marked with a ++ Symbol. US numbering is unique to each E."

7.1 WaterGAPCore T1

7.1.1 E1 Handling I/O

U1 Specifying file outputs
As a user, I want to be able to specify what storages and fluxes or other variables of the model are written to disk.

Acceptance criterion: Results are in netcdf and have proper meta information (comparable to isimip output).

UX Run the calibrated WaterGAP with different climate change scenarios as input
4.1.24.1.12 As a user, I want to make changes to the options / configuration file. When changing climate input it is necessary to rerun GIM with that specific climate (warning message should be provided).

Acceptance criterion: A warning should be displayed that re-running GIM is necessary. Changing the code should not be necessary.

UX save model configuration information
As a data user, I want to have easy accessible information on which model configuration and source code has been used to produce the results.

Acceptance criterion: model configuration (e.g. time, paths, climate forcing, water use, code version, settings in options and output options) are stored successfully in a reasonable format (netcdf or json)

UX Reading in standardized spatially and temporally distributed climate input data
4.1.24.1.12 As a user, I want to be able to read in well defined NetCDF(3-4) as forcing and map it on the model structure to WaterGAPCore.

Acceptance criterion: The reading of 10 years globally takes 3 minutes (no hard limit).

Comments: Product owner will provide well defined climate forcing data as example.

UX Reading in JSON configuration files
4.1.2 As user I want to change the model configuration based on JSON configuration files (see 3.1.3).

Acceptance criterion: configuration information is correctly read in.

UX Gregorian calendar
As a user I'm expecting WaterGAP to be using the exact Gregorian calendar. This will help me to get output files that also take into account leap years.

Acceptance criterion: I have simulated output of 29. of February.

7.1.2 E2 Reimplementing simulating storages and fluxes

UX Canopy storage
As a user I want that the canopy storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: canopy storage and related fluxes are producing plausible outputs.

UX Snow storage
As a user I want that the snow storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: snow storage and related fluxes are producing plausible outputs.

UX Soil storage
As a user I want that the soil storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: soil storage and related fluxes are producing plausible outputs.

UX Groundwater storage
As a user I want that the groundwater storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: groundwater storage and related fluxes are producing plausible outputs.

UX Local lake storage
As a user I want that the local lake storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: local lake storage and related fluxes are producing plausible outputs.

UX Local wetland storage
As a user I want that the local wetland storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: local wetland storage and related fluxes are producing plausible outputs.

UX Globl lake storage
As a user I want that the global lake storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: global lake storage and related fluxes are producing plausible outputs.

UX Reservoir storage
As a user I want that the reservoir storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: reservoir storage and related fluxes are producing plausible outputs.

UX Global wetland storage
As a user I want that the global wetland storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: global wetland storage and related fluxes are producing plausible outputs.

UX River storage
As a user I want that the river storage and related fluxes are functionally implemented, based on the documentation of 2.2d and 2.2e.

Acceptance criterion: river storage and related fluxes are producing plausible outputs.

7.1.3 E3 Flexibility in implementing new algorithms for existing processes and new processes

Following user stories are simple examples for new algorithms.

UX Implementation of a new PET algorithm
As a user I want to be able to implement a new PET algorithm to be used in the model. For this I need to be able to read in new input data and specify the PET algorithm.

Acceptance criterion: Implementation of new PET algorithm is feasible and the new algorithm is used in new runtime.

UX want to add a new process to the code e.g. active vegetation
As a user i want to be able to introduce a new process to existing code.

Acceptance criterion: WaterGAPCore is written modular and is well documented, including guidelines for implementing new processes.

7.1.4 E4 Long distance water transfer schemes for urban water supply (existing) and irrigation

UX long distance water transfer schemes for urban water supply
As a user I want to consider long distance water transfer schemes for urban water supply. (ask WaterGAP3 people).

Acceptance criterion: process is integrated and works as expected

UX long distance water transfer schemes in general
I want to use in WaterGAPCore long distance water transfer based on prescribed information in databases.

Acceptance criterion: process is integrated and works as expected

7.1.5 E5 Consistency and quality checks

see also 3.1.2

U1 Physical units
As a user I want to know which variable has which physical unit and guarantee that conversion is physically correct.

Acceptance criterion: The conversion of units in the code is checked automatically.

U2 Waterbalance check
As a user I want a short information after running WaterGAP on the global / basinwide (depending on the type of run) waterbalance error.

Acceptance criterion: Loginformation on waterbalance error is shown.

UX input data availability check
As user I want that WaterGAPCore cancel its run, when input data are not available (data not accessible behind path or time period not included).

Acceptance criterion: model stops running

UX Checking plausible ranges for input data
As user I want to be able to select if WaterGAPCore checks all or selected input data on physical plausibility ranges (e.g. to prevent negative precipitation)

Acceptance criterion: proper warning messages with inconsistent input data.

UX Implementing configuration check on ability to run prior runtime
As a user I want WaterGAPCore to check if the current configuration is runable and be warned about (all paths are set, options are set or have default value).

Acceptance criterion: model does not start running with missing configuration settings.

UX Checking configuration file (model options) on logical conflicts
As a user I want WaterGAPCore to check if model options are logically meaningful (e.g. to prevent last year of reservoirs used earlier than first year of reservoirs used).

Acceptance criterion: model gives out a proper warning in case of logical conflicts

UX ranges for variables
As a developer and user I want to specify resp. know which range of values a variable can get and that WaterGAP throws an error if a value out of range is tried to be set.

Acceptance criterion: Warning message is written to the logfile in case of corrupting variables.

7.1.6 E6 Implementing floodplain algorithm

7.1.7 E7 Guide users to use other input data (content) to run WaterGAPCore

U1 tutorial how to run the model with modified input (e.g. climate, landuse etc.)
As a user I want to replace existing well-defined input data to run the model.

Acceptance criterion: Having a documentation with definition (description, ranges) of input data. Having an example tutorial how to generate a new input data set.

U2 tutorial to re-calibrate
As a user I want to be able to re-calibrate WaterGAP in the "standard" way (calibrating up to three parameters on long-term average observed streamflow).

Acceptance criterion: Tutorial and/or a tool to re-calibrate WaterGAP.

7.1.8 E8 flexible spatial resolution with first target of 5 and 30 arc minutes

U1 model structure runs in different spatial resolution
As a user I want to be able to run the model in the selected spatial resolution.
Acceptance criterion:

U1 higher spatial model resolution can be run with coarser climate forcing
As a user I want to be able to run a high(er) spatial resolution even though with coarser resoluted climate forcing.

Acceptance criterion:

U1 Generation of input files (except climate forcing) independently of scale
As a user I want

Acceptance criterion: ?

7.1.9 E9 Enhancing groundwater representation

U1 Using G3M as groundwater representation
As a user I want to be able to use the coupled G3M model instead of the linear groundwater model for the whole time period.

Acceptance criterion: G3M is working properly.

U2 Using G3M as groundwater representation for specific time steps

Due to performance reasons it might be necessary to use a coupling only for specific steps in time (e.g. every 2 weeks) and the linear groundwater model in between.

Acceptance criterion: G3M runs at given time intervals and in between the linear groundwater models

7.1.10 E10 Integration of temporally dynamic land use

U1 Annual land use dataset can be considered
As a user I want to select if annual varying land use data can be used instead of a static land use data. This requires some conceptual thoughts (e.g. what to do with storages once the land use classification switches, if it is still valid to use one dominant land use class (or to use a more distributed approach) and if the transition between land uses (e.g. over the year break) should be smoothed (e.g. over a period of 2 months instead of 1 day)). Not on a high priority.

Acceptance criterion: The model runs with a dynamic land use dataset

7.1.11 E11 Implementing interfaces to calibration, sensitivity and uncertainty, PDAF runs

U1 Coupling interface
As a user, I want to be able to couple the model to any other program through a transparent interface that can be implemented by the coupling model. I would benefit through the flexibility and speed.

Acceptance criterion: The model has a coupling interface through which all outputs of the model can be accessed in a "online" matter.

U2 Modify (by calibration) static fraction of "sectoral" water withdrawals from groundwater / surface water GWSWUSE
As a user I want to be able to modify the static fractions of how much water is withdrawn from groundwater or surface water per sector for running GWSWUSE.

Acceptance criterion: GWSWUSE runs properly with modified static fractions.

U3 Modify (by calibration) dynamic fraction of "sectoral" water withdrawals from groundwater / surface water GWSWUSE
As a user I want to be able to use temporal dynamic fractions of how much water is withdrawn from groundwater or surface water per sector for running GWSWUSE, e.g. for scenario runs.

Acceptance criterion: GWSWUSE runs properly with temporal dynamic fractions.

U4 Modifying calibration parameters
As a user
Acceptance criterion: ?

U5 model parameter modification
As user I want to change any model parameter(except physical constants) without touching the code, to get flexibility. Note that this model parameter does not necessarily needs to be a calibration parameter.

Acceptance criterion: Any model parameter content is subject to change from outside the core

U6 Snapshot
As user, I want to start the model from flexible starting points in time with described (already calculated) storages (and additional information as needed), in order to be able to do e.g., forecast simulations or similar

Acceptance criterion: model runs properly from a snapshot (similar behaviour compared to a transient run)

U7 Calibration
As a scientist, I want to calibrate traditional WaterGAP calibration parameters (gamma, cfa, cfs) to new climatic input with lta discharge observations so that I am able to use new climate input.

Acceptance criterion: Calibration runs in two weeks

U8 Calibration parameters
As scientist, I want to modify predescribed model parameters (in a predefined range), in order to run sensitivity runs or calibrations etc.

Acceptance criterion: There is documented api to modify those model parameters (e.g. JSON)

U9 Data assimilation
As a scientist, I want to change water storages and model parameters in runtime of WaterGAP and continue the run, in order to have online coupling capabilites.

Acceptance criterion: ?

U10 reservoir algorithm calibration parameter
As a user i want to have a calibration parameter in the reservoir algorithm.

Acceptance criterion: ?

7.1.12 E12 WaterGAP can simulate individual basins also using basin-specific input data

U4 enabling to run single river basins
As a user ..

Acceptance criterion: ? TODO define USs

7.1.13 E13 Monitoring (near real time WaterGAP runs) and forecasting

TODO define USs

7.2 WaterGAPUse T2

7.2.1 E1 Considering desalination

++

7.2.2 E2 Run GIM consistent with WaterGAPCore

U1 Modifying allocation schemes for downscaling domestic and manufacturing sector

As a user I want to ... ?
Acceptance criterion: ?

U2 Modifying time series for non-irrigation water use sectors

As a user I want to ... ?
Acceptance criterion: ?

U3 Re-Running GIM with new climate information
As a user I want to ... ?
Acceptance criterion: ?

U4 Re-Running GIM with new irrigation areas
As a user I want to ... ?
Acceptance criterion: ?

7.3 WaterGAPTools T3

7.3.1 E1 implementing alternative drainage direction map

TODO define USs

7.3.2 E2 Calibration to long term average streamflow (standard WaterGAP calib)

TODO define USs

7.3.3 E3 preparing input files for WaterGAPCore

UX reading CSV, GeoJSON, JSON, shapefile
As user I want to read in CSV, GeoJSON, JSON, shapefiles to read in information and data to generate input files.

Acceptance criterion: No information is lost. Comment: Product owner will provide sample files for those formats.

UX Integration of surface water body data
As user I want to convert a well defined shapefile with lake polygons to generate input files to WaterGAPCore.

Acceptance criterion: ?

Waterbalance check
outside of watergapcore, with model output

7.3.4 E4 preparing configuration files

UX generation of a model configuration file
As user I want to have one configuration json file, where i can define input file paths, model options, time configuration and outputfile.

Acceptance criterion: all required information is stored in a readable (human, machine) and structured way.

UX GUI for model configuration file
As user I want to be able to click a model configuration together in a GUI.

Acceptance criterion: GUI works in all common operating systems

7.4 Overarching stories T4

7.4.1 E1 Example and tutorials

TODO define USs

7.4.2 E2 Enable prescribed variants of WaterGAP in a self-contained environment

TODO define USs

A Glossary

"Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS."