Memorandum

To: Brian Macpherson (Colorado Water Conservation Board),
Carolyn Kemp (CWCB),
Mary Halstead (Colorado Division of Water Resources)
From: Steve Malers (Open Water Foundation, OWF)
Subject: Notes on PACSM
Date: April 1, 2019

Contents

Introduction

This memorandum provides notes about PACSM (Platte and Colorado Simulation Model), including programming language, development environment, and development team, in order to provide perspective for whether StateMod should be converted to a different language. These notes are based on a conversation between Steve Malers of the Open Water Foundation and Steve McWilliams of Denver Water on March 18 and 19, 2019.

Additionally, the document "Independent Review of PACSM and Options for Further Development" by Dr. Willem A Schreüder was reviewed, with a summary of important points.

Background

PACSM is the basin simulation model used by Denver Water to simulate operations of raw water systems, as input for planning and operations. It was developed by enhancing the BESTSM model (similar to StateMod), specifically to model the Denver Water system.

Technical Notes

The following are notes on specific technical topics.

Programming Language

PACSM is coded in Fortran, with original Fortran 77 code being updated to more recent versions for some syntax elements (for example use of IMPLICIT NONE, renaming variables to be more clear, removing most GOTO statements, and responding to compiler warnings), with work occurring around 2013. The following are important considerations:

- The model has generic rules (similar to StateMod) and 2-3 operating rules that are specific to Denver Water.
- The timestep is daily and the standard dataset has a 61-year period of record.
- The model does not re-operate (based on Steve McWilliams' description).

Operational Environment

The model input is primarily comma-separated-value (CSV) files that are created from a Microsoft Access database. There are approximately a dozen output files, of which 2-3 are binary files and the rest are CSV text. Postprocessors include programs written in VBA for Excel. A C# program creates monthly summaries that are used for decision-making. Previously the environment included MapInfo for geographic information system (GIS). An attempt was made to move to Esri ArcMap postprocessor but this was abandoned. Now the post-processor is C# and focuses on standard reports. Users don't typically see model input files and instead interact with Microsoft Access tables.

User and development documentation exist but "are lacking"

Component Libraries Used

The model is Fortran code. Libraries used in pre- and post-processors were not discussed.

License and Open Source Considerations

The PACSM software is used only by Denver Water with customization for their use. The code is not available outside of Denver Water. Denver Water uses disclaimer and non-disclosure agreements as needed to share modeling datasets and results.

Supported Operating Systems

The model runs on Windows 10.

Development Environment

The development environment uses:

- Commercial Microsoft Visual Studio 2010 and commercial Intel Fortran compiler
- Version control:
    - Dated folders
    - No version control system such as GitHub

Software Testing

Software testing is implemented as follows:

- 5 datasets are run, which exercise features that "touch every file".
- Batch scripts have been developed to run the model and compare output files.
- Custom programs have been written to compare binary output files.
- The open source winmerge software (http://winmerge.org/) is used to perform differences.

Software Performance Optimization

The following are changes that were made to optimize the code:

- Array loop index order was changed to follow best practices for Fortran code optimization.
- Return flow math was updated to skip processing zero return value (perhaps avoiding input/output performance hit).
- The run time has been reduced from 7 minutes to ~1 minute (this may not be accurately recorded from the conversation – seems low).
- Denver Water has a "tree ring" dataset run and the run environment executes multiple runs at the same time.

Development Team

The modeling team consists of:

- A manager.
- Four people that focus on running the model and interpreting results - ~90% of their days.
- Steve McWilliams is the only programmer. 95% of his time is focused on maintaining the software.

Summary of Review Comments for "Independent Review of PACSM and Options for Further Development", March 2012

The following are Steve Malers' review comments on this memo. Note that the recent conversation with Steve McWilliams provided more recent information, such as the move away from MapInfo in recent years. Recent information is largely consistent with the 2012 memo.

1. Executive Summary:
    a. The point is made that if Denver Water were to adopt a third party tool such as StateMod, this might be cumbersome because adding features may be difficult. Comment: Hopefully the OpenCDSS effort has removed barriers to this option, but coordination and skilled developers would be important.
    b. The point is made that if it is necessary to replace or update the PacsmGUI and PacsmMap that it may be necessary to use other technologies such as Qt. Comment: Qt is certainly a popular library for user interface (UI) development, in particular for C++ and Python languages. Other tools such as GIS, can be used; however, integration can be difficult and trade-offs must be considered. The StateMod GUI has not been actively maintained and the choice of language should consider selecting a language that is appropriate for UI. OWF has developed user interfaces in various technologies including Java, Python (Qt), and web technologies, and .NET is also an option.
2. "The Future of Fortran":
    a. The point is made that projects such as CDSS will result in "an adequate pool" of Fortran programmers. Comment: This has clearly not been the case for CDSS given the challenge of finding Fortran programmers for StateMod.
    b. The point is made that a rewrite of a Fortran program in an object-oriented language would be a complete rewrite and not justified. Comment: Many programs have undergone major rewrites in order to implement new technologies and improved design. Such investment may be needed periodically to allow for innovation more nimble and efficient maintenance. In the case of PACSM, the decision has apparently been made to employ a full-time programmer and it is likely that such a person could update the code to a different (e.g., object-oriented) language as part of their duties, perhaps within a year of effort. To some degree, modernizing the language and using an object-oriented design reduces the risks for future maintenance because more developers will be available for the programming language.
3. "Language Standard and Style Guidelines":
    a. A recommendation was made to remove tab characters from code. Comment: StateMod has some of the same issues and such aspects of the Fortran language are at times an irritation and at other times a bug. For example, improper formatting of code can cause truncation of code on right-most part of lines, resulting in bugs. Such things should be eliminated through compiler options, and are in most cases not an issue with other languages.
    b. A recommendation is made to remove GOTO statements to improve logic. Comment: This is also a valid comment for StateMod, with a code search listing over 2500 instances of GOTO. Such logic can be difficult for new developers to understand, although many cases may simply by to jump to the end of a function if an error in input is detected.
    c. A recommendation is made to use modules to replace common blocks. Comment: This is a good recommendation and can make the code more modular. StateMod suffers

from some inherent design considerations that increase the chances of bugs, such as global common blocks.

d. A recommendation is made to use consistent code formatting, indentation, variable formatting, etc. Comment: This is of course a good recommendation and new programmers will likely be tempted to make the code more understandable. Such cleanup would be a part of object-oriented migration.

e. A recommendation is made to remove unused variables, dead code, etc. Comment: Such issues are easily detectable using a modern compiler and integrated development environment (IDE) and changes should occur as part of normal coding. Migrating StateMod to gfortran should point out issues that can be resolved and using additional compiler options can point out further issues for cleanup.

f. Comment: In addition to the above, improvements to StateMod recommended by OWF are to use parameter statements (or equivalent) to isolate constants such as array dimensions and rename subroutines and functions to be more understandable, such as including operating rule and descriptive name in the subroutine name and corresponding source code file.

g. Comment: The PACSM model dataset appears to be simpler than the complex CDSS datasets and discussion of performance using different technologies needs to compare examples of code for the same input and logic (as much as possible).

4. "Review of Modeling Environment"
    a. Comment: Use of PACSM's pre and post processors is equivalent in many ways to the data-centered use of data management interface (DMI) software such as StateDMI and TSTool. TSTool has been enhanced in recent years to write to Excel and StateDMI has also recently been enhanced to include such features (to support new web service integration). Additional visualization can occur and (re)implementation of StateMod GUI could provide useful functionality. Consideration of the full modeling environment is a consideration for whether a new language for StateMod is chosen because the language impacts how much integration can occur with other tools.
    b. PACSM uses Microsoft Access for data processing. Comment: This technology provides useful features but is also limiting in that the resulting code is not very transparent given that it is packaged with the database. Using Access does not appear to be a consideration for CDSS, although use of a database to manage and process model output is something that could be considered in the future.
    c. The point is made that adopting a more integrated language that supports user interface development (such as C++ for RiverWare) might result in a better product. Comment: This is also a consideration for StateMod. The use of Fortran for computation code limits options for implementing an integrated user interface.
5. "Refactoring the Environment in ArcGIS"
    a. The points are made that Esri changes its approach, that Python is typically used to program GIS environments, and PyQt could be used. Comment: Both open source QGIS and Esri ArcGIS Pro provide (and recommend) Python libraries to create custom GIS interfaces. OWF has developed the open source GeoProcessor using a design that supports QGIS PyQGIS and ArcGIS Prop ArcPy Python libraries. If StateMod were coded in Python, then integration with GIS for map-based interfaces would be simpler, although the developer would need to have skills to use the development libraries.
6. "Refactoring the Environment in Qt"
    a. The recommendation is made to consider Qt for user interface, but recognizes the effort to implement. Comment: There is no getting around learning a technology when it is

chosen for a solution and often the effort of implementing a technology takes longer than expected. OWF's approach has been to learn multiple technologies so that an understanding of technologies is considered in design, and experience can be leveraged in programming projects. Qt provides standard user interface tools for C++ (for example used in RiverWare) and Python (used in QGIS) and Qt would be a good choice for UI if those languages are chosen. OWF is using Qt in the open source GeoProcessor software. The adoption of any technology requires planning to ensure that staff are available to support the technology over time.

7. "Review of Alternate Modeling Programs"
    a. Comment: A list of models is presented and compared. It is likely that each of these models is experiencing similar issues with maintenance, with various choices of technologies resulting in different levels of risk in being able support the software. OWF has interviewed the developers of RiverWare and MODSIM to gain perspective of challenges for those tools and has provided the State with summary memos. A common problem is that programmers that are not computer science graduates (or don't have significant experience with coding from different perspectives) may "brute force" their way through coding, resulting in code that is not as easy to maintain. In contrast, a computer science graduate requires time and guidance to understand water resources concepts in order to write code that is understandable and not too abstract. This balance is the challenge that all model programming teams face, with different degrees of success.
    b. There is discussion about the integration of GUI and simulation code for MODSIM and RiverWare that makes it appear that GUI is tightly linked to simulation code. Comment: GUI and simulation code can actually be integrated in a way that allows for batch execution without accessing GUI code, or use GUI code as needed, such as producing output graph products. These design considerations could be easily handled if StateMod were translated to a different language.

A general comment about the 2012 memo and the decision of whether to update StateMod code to another language is whether to move StateMod to an object-oriented design and corresponding language. Moving from a procedural language to object-oriented language is a major paradigm shift. Making this change can result in improved software quality that is easier to maintain. And, adopting an object-oriented language will transform StateMod code into a form that can be understood by any programmer that understands object-oriented concepts. Said differently, using a non-object-oriented language and software design will limit options to find programmers and will result in software that is more difficult to maintain. OWF is currently working on a project to evaluate alternative languages for StateMod and each supports object-oriented design: Java, C#, Python, and new versions of Fortran (2003, 2008, etc.). A migration to object-oriented design is highly recommended, regardless of language. If this decision is made, then secondary criteria should be considered, such as support for user interface, performance, availability and efficiency of programmers, and integration with other components. Fortran has fewer options for UI and integration, although some options are available.