Software Development Tools for SWAT

Seshu Tirupathi
IBM Research - Ireland

2017 International SWAT Conference - Warsaw
June 28, 2017
Acknowledgments : Prof. Gosain, Prof. Srinivasan
DevOps - Software Development and IT Operations
  SWAT Model Development
  Constraints and Opportunities

Code Development Tools
  Documentation
  Unit Test Framework
  Build Tools

Development Environment
  SWAT Requirements
  Virtualization - Vagrant
  Containers - Docker

Conclusion
Outline for section 1

DevOps - Software Development and IT Operations
SWAT Model Development
Constraints and Opportunities

Code Development Tools
Documentation
Unit Test Framework
Build Tools

Development Environment
SWAT Requirements
Virtualization - Vagrant
Containers - Docker

Conclusion
Outline for section 1

DevOps - Software Development and IT Operations
SWAT Model Development
Constraints and Opportunities

Code Development Tools

Development Environment

Conclusion
• Fortran source code.
• SWAT to SWAT+ to gSWATCloud.
• Primarily based on Intel Fortran Compiler (ifort), Visual Studio IDE.
• Open source code in the true sense. Open source standards?
DevOps - Software Development and IT Operations

SWAT Model Development

Constraints and Opportunities

Code Development Tools

Development Environment

Conclusion
Constraints

Visual Studio → Code → gSWAT Cloud → Production
Opportunities

Visual Studio

Code ➔ Unit tests ➔ Integration ➔ Integration tests ➔ Ready for deployment ➔ Production

gSWAT Cloud
Opportunities
Outline for section 2

DevOps - Software Development and IT Operations
  SWAT Model Development
  Constraints and Opportunities

Code Development Tools
  Documentation
  Unit Test Framework
  Build Tools

Development Environment
  SWAT Requirements
  Virtualization - Vagrant
  Containers - Docker

Conclusion
Outline for section 2

DevOps - Software Development and IT Operations

Code Development Tools
  Documentation
  Unit Test Framework
  Build Tools

Development Environment

Conclusion
Easier to use and maintain the code.
User-friendly approach to understand code logic.
Graphical visualization of interdependencies.
Auto-generating documentation packages for SWAT.
Examples:
- ROBODoc
- Doxygen
- sphinx-fortran
- FORD
- doctran
**aquifer_module Module Reference**

**aquifer components module**

More...

Collaboration diagram for aquifer_module:

Collaboration graph

**Data Types**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aquifer_header</td>
<td></td>
</tr>
<tr>
<td>aquifer_data_parameters</td>
<td></td>
</tr>
<tr>
<td>aquifer_database</td>
<td>type for reading aquifer input files rows More...</td>
</tr>
<tr>
<td>aquifer_dynamic</td>
<td></td>
</tr>
<tr>
<td>aquifer_state_parameters</td>
<td></td>
</tr>
</tbody>
</table>

**Public Member Functions**

- type(aquifer_dynamic) function aqu_add(aqo1, aqo2)
- type(aquifer_dynamic) function aqu_div(aq1, const)
Public Attributes

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer iaq</td>
<td></td>
</tr>
<tr>
<td>integer iaqdb</td>
<td></td>
</tr>
<tr>
<td>type(aquifer_database),</td>
<td>read from the aquifer</td>
</tr>
<tr>
<td>dimension(), allocatable</td>
<td>database file named</td>
</tr>
<tr>
<td>aqudb</td>
<td>aquifer.aqu</td>
</tr>
<tr>
<td>type(aquifer_data_parameters),</td>
<td></td>
</tr>
<tr>
<td>dimension(), allocatable</td>
<td>aqu_prm</td>
</tr>
<tr>
<td>type(aquifer_state_parameters),</td>
<td></td>
</tr>
<tr>
<td>dimension(), allocatable</td>
<td>aqu_st</td>
</tr>
<tr>
<td>type(aquifer_dynamic),</td>
<td></td>
</tr>
<tr>
<td>dimension(), allocatable</td>
<td>aqu</td>
</tr>
<tr>
<td>save aqu_m</td>
<td></td>
</tr>
<tr>
<td>type(aquifer_dynamic),</td>
<td>save aqu_y</td>
</tr>
<tr>
<td>dimension(), allocatable</td>
<td></td>
</tr>
<tr>
<td>save aqu_a</td>
<td></td>
</tr>
<tr>
<td>type(aquifer_dynamic)</td>
<td>aqu_z</td>
</tr>
<tr>
<td>type(aquifer_header)</td>
<td>aqu_hdr</td>
</tr>
</tbody>
</table>

Detailed Description
## Detailed Description

**aquifer components module**

**Author**
- Jeff Arnold

**Version**
- 0.0.0

**Date**
- 06 28 2016

This module includes the aquifer database parameters

**REVISION HISTORY:**
- 2016.24 test

## Member Function/Subroutine Documentation

```plaintext
/* type (aquifer_dynamic) function aquifer_module::aqu_add ( type (aquifer_dynamic), intent(in) aq01,
  type (aquifer_dynamic), intent(in) aq02
  ) */

/* type (aquifer_dynamic) function aquifer_module::aqu_div ( type (aquifer_dynamic), intent(in) aq1,
  real, intent(in) const
  ) */
```
### Member Data Documentation

<table>
<thead>
<tr>
<th>Type</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>type (aquifer_dynamic), dimension(:), allocatable aquifer_module::aqu</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_dynamic), dimension(:), allocatable, save aquifer_module::aqu_a</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_header) aquifer_module::aqu_hdr</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_dynamic), dimension(:), allocatable, save aquifer_module::aqu_m</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_data_parameters), dimension(:), allocatable aquifer_module::aqu prm</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_state_parameters), dimension(:), allocatable aquifer_module::aqu_st</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_dynamic), dimension(:), allocatable, save aquifer_module::aqu_y</code></td>
<td></td>
</tr>
<tr>
<td><code>type (aquifer_database), dimension(:), allocatable aquifer_module::aqdb</code></td>
<td></td>
</tr>
</tbody>
</table>

read from the aquifer database file named aquifer.aqu

**See Also**

- `aqu read`
SWAT Documentation

aquifer_module::aquifer_database Type Reference

type for reading aquifer input files rows More...

Collaboration diagram for aquifer_module::aquifer_database:

Public Attributes

character(len=16)  aqunm = **
real     flo = 0.05
         [FLO] groundwater flow [mm H2O] More...
real     stor = 0.
         [STOR] depth of water in shallow aq [mm H2O] More...
real     hgt
         [HGT] groundwater height [m] More...
real     no3 = 0.
         [NO3] nitrate conc in shallow aq converted to kg/ha [ppm NO3-N] More...
real     minp = 0.
         [MNP] mineral P concentration [mg P/L] More...
real     orgn = 0.
Outline for section 2

DevOps - Software Development and IT Operations

Code Development Tools
  Documentation
  Unit Test Framework
  Build Tools

Development Environment

Conclusion
Overview

- Test Driven Development (TDD).
- When/if done wisely, avoids glaring mistakes during code development.
- Easier integration of contributions.
- Easier maintenance of code for owners/administrators.
<table>
<thead>
<tr>
<th>Name</th>
<th>XUnit</th>
<th>MPI</th>
<th>SWAT</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRUIT&lt;sup&gt;1&lt;/sup&gt;</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>Ruby + Fortran, User friendly&lt;br&gt; Last update: 2016-10-23.&lt;br&gt; Discontinued</td>
</tr>
<tr>
<td>FLIBS&lt;sup&gt;2&lt;/sup&gt;</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>Tcl Make, simple to implement</td>
</tr>
<tr>
<td>ObjexxFTK&lt;sup&gt;3&lt;/sup&gt;</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>Python + Fortran, Not open source, paid service</td>
</tr>
<tr>
<td>FRUITPy&lt;sup&gt;4&lt;/sup&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Python + Fortran, limited documentation. Last update 1 year back.</td>
</tr>
<tr>
<td>pFUnit&lt;sup&gt;5&lt;/sup&gt;</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Python + Fortran, extensive documentation, regular updates</td>
</tr>
</tbody>
</table>
pFUnit Example

- Usage given in http://pfunit.sourceforge.net/page_Usage.html
- Write preprocessor input files which describes the modules required from pFUnit. Also includes the "assert" statements.

```
!testTheta.pf
@test
subroutine testTheta()
    use pfunit_mod
    implicit none
    @assertEqual(1.0,theta(1.0,1.0,20)) !r20*thk**(tmp-20.)
end subroutine testTheta
```

- Create a file called 'testSuites.inc' which includes information on the tests that are to be checked.

```
ADD_TEST_SUITE(testTheta_suite)
```

- Makefile to automate the build and test process.
Outline for section 2

DevOps - Software Development and IT Operations

Code Development Tools
  Documentation
  Unit Test Framework
  Build Tools

Development Environment

Conclusion
Build Tools

- Operating System independent build tools.
  - **Makefile**: Dependency resolution is very difficult. Some successful efforts are available online. Not a viable long-term solution.
  - **CMake**: More viable approach for dependency hierarchy.
  - **FoBiS.py**: Automatic dependency hierarchy. Simple to implement.
  - **Other Options**: Meson, Waf, fake etc

- Integrated Development Environment (IDE) beyond Visual Studio.
  - **Photran**: Eclipse Plugin + Fortran. Universal IDE.
  - **IntelliJ**: Incredible success for Java, Python, C++ etc. IntelliJ IDEA plugin for Fortran available since May 2017.
  - **Code::Blocks**: Customized version for Fortran.
Outline for section 3

DevOps - Software Development and IT Operations
SWAT Model Development
Constraints and Opportunities

Code Development Tools
Documentation
Unit Test Framework
Build Tools

Development Environment
SWAT Requirements
Virtualization - Vagrant
Containers - Docker

Conclusion
Outline for section 3

DevOps - Software Development and IT Operations

Code Development Tools

Development Environment

SWAT Requirements

Virtualization - Vagrant

Containers - Docker

Conclusion
SWAT Requirements

- Future of SWAT? Single platform rigid development process or platform independent process.
- Framework for integration with other open-source libraries.
- Multiple code development environments.
- Code deployment/Cloud Ready/Live in production.
Outline for section 3

DevOps - Software Development and IT Operations

Code Development Tools

Development Environment
  SWAT Requirements
  Virtualization - Vagrant
  Containers - Docker

Conclusion
Virtualization - Vagrant

Current Process

- Standard solution is to create manual instructions for getting the code to work.
- Lots of constraints and unknowns.
- Time consuming process.

Vagrant - Developers tool for creating a virtual environment using command line.

- Development environment isolated to ensure that SWAT code works.
- Easier to get started with the core development of the code.
Example Vagrant File Snippet

```ruby
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "1024"
    vb.cpus = 1
  end

------

apt-get update
apt-get install gfortran
echo "fetch swat repo"
wget http://swat.tamu.edu/media/115510/rev664_source.zip
echo "unzip swat source code"
unzip rev664_source.zip
```
DevOps - Software Development and IT Operations

Code Development Tools

Development Environment
  SWAT Requirements
  Virtualization - Vagrant
  Containers - Docker

Conclusion
Containers - Docker

- Containers - Application’s components + dependencies + binaries + libraries
- Lightweight deployments that can run on any computer, infrastructure or cloud.
- hub.docker.com

Repositories (10)
FROM alpine:latest

MAINTAINER John Doe "johndoe@blah.com"

RUN apk add --update \ 
    python \ 
    gfortran

WORKDIR /swat
ADD requirements.txt /swat
RUN pip install -r requirements.txt
Outline for section 4

DevOps - Software Development and IT Operations
  SWAT Model Development
  Constraints and Opportunities

Code Development Tools
  Documentation
  Unit Test Framework
  Build Tools

Development Environment
  SWAT Requirements
  Virtualization - Vagrant
  Containers - Docker

Conclusion
Appendix

1. https://sourceforge.net/projects/fortranxunit/