Best Practices / Lessons Learned from Software Qualification and Model Verification

Greg Flach
Environmental Modeling

Interagency Steering Committee on Performance and Risk Assessment Community of Practice Annual Technical Exchange Meeting
October 19-20, 2016
Outline

• Models and Analyses are more than Codes

• Quality Assurance (QA) versus Quality Control (QC) approaches

• QA / QC Risks and Remedies

• Examples

• Summary thoughts

• Audience participation
Components of a Risk or Performance Assessment Analysis Involving Simulation

Software, Models and Analyses encompass more than Codes:

- **Software** = Code + Documentation
- **Numerical Model** = Input + Code
- **Overall Model** = Concept + Data + Input + Code
- **Analysis** = All of the above

**Quality Assurance** must address the entire **Analysis** to support sound Decisions.
Quality Assurance (QA) versus Quality Control (QC) Paradigms

In my experience:

- QA and QC are complementary
- QA takes forefront w.r.t software
- QC takes forefront w.r.t model or analysis

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Software Quality Assurance (SQA)</th>
<th>Software Quality Control (SQC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>SQA is a set of activities for ensuring quality in software engineering processes (that ultimately result in quality in software products). The activities establish and evaluate the processes that produce products.</td>
<td>SQC is a set of activities for ensuring quality in software products. The activities focus on identifying defects in the actual products produced.</td>
</tr>
<tr>
<td>Focus</td>
<td>Process focused</td>
<td>Product focused</td>
</tr>
<tr>
<td>Orientation</td>
<td>Prevention oriented</td>
<td>Detection oriented</td>
</tr>
<tr>
<td>Breadth</td>
<td>Organization wide</td>
<td>Product/project specific</td>
</tr>
<tr>
<td>Scope</td>
<td>Relates to all products that will ever be created by a process</td>
<td>Relates to specific product</td>
</tr>
<tr>
<td>Activities</td>
<td>• Process Definition and Implementation</td>
<td>• Reviews</td>
</tr>
<tr>
<td></td>
<td>• Audits</td>
<td>• Testing</td>
</tr>
<tr>
<td></td>
<td>• Training</td>
<td></td>
</tr>
</tbody>
</table>

Many people still use the term Quality Assurance (QA) and Quality Control (QC) interchangeably but this should be discouraged.

http://softwaretestingfundamentals.com/sqa-vs-sqc/
Software Quality Assurance / Qualification

Best Practices

• “QA” approach: Check Once, Use Many
  – Simulation software embodies relatively general and static functions
  – Application of systematic software QA activities through formal procedures worth the investment

Experience / Lessons Learned

• Input errors and misuse more common than coding errors
  – Often due to code functionality that is ambiguously, obscurely or not documented
  – Documentation, not code, problem

• The more code use the better (early use, large user base)
• Developer-only testing is often lacking
Example: Retardation Coefficient, R

Conventional definition of R

\[ R = 1 + \frac{\rho_s (1 - n) K_d}{S n} \]

Unusual alternative definition and Porflow default

\[ R = 1 + \frac{\rho_s (1 - S n) K_d}{S n} \]

Diagnosis:
- QA testing had involved only fully saturated cases \((S = 1)\)
- Discovered through code-to-code benchmarking during model abstraction
- User not alerted to non-conventional retardation definition in documentation
Best Practices

• “QC” approach: Check Every Use (in addition to QA for Code)
  – Pre-processing software is embodies specific and undocumented functions
  – Input and/or Output checked every time is more efficient than formal software QA

Experience / Lessons Learned

• Flawed Conceptual Model poses greatest Decision risk
• Input (model setup) errors more common than Code errors
• Independent and thorough technical review is critical
• Independent development efforts are highly effective at identifying Model errors
Example: General Separations Area Groundwater Flow Model (Savannah River)

Quality Control check on groundwater flow field Output:

- Confirm steady-state mass balance cell-by-cell and for domain
- Confirm Darcy’s Law honored at each cell face
- Confirm simulated and measured water levels and stream baseflows agree
## Example: Independent Model Input Summary Tables

<table>
<thead>
<tr>
<th>Material</th>
<th>Time Interval</th>
<th>Start</th>
<th>End</th>
<th>Horizontal Conductivity</th>
<th>Vertical Conductivity</th>
<th>Porosity</th>
<th>Density</th>
<th>Water retention</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKFILL</td>
<td>T101-T160</td>
<td>0</td>
<td>100000</td>
<td>7.60E-05</td>
<td>4.10E-05</td>
<td>0.35</td>
<td>2.631</td>
<td>CoBackfill</td>
</tr>
<tr>
<td>CLEAN_GROUT</td>
<td>T101-T115</td>
<td>0</td>
<td>1106</td>
<td>6.41E-09</td>
<td>6.41E-09</td>
<td>0.58</td>
<td>2.405</td>
<td>fractured_clean</td>
</tr>
<tr>
<td>CLEAN_GROUT</td>
<td>T116</td>
<td>1106</td>
<td>1250</td>
<td>1.68E-06</td>
<td>1.68E-06</td>
<td>0.58</td>
<td>2.405</td>
<td>fractured_clean</td>
</tr>
<tr>
<td>CLEAN_GROUT</td>
<td>T120</td>
<td>1690</td>
<td>2113</td>
<td>3.11E-05</td>
<td>3.11E-05</td>
<td>0.58</td>
<td>2.405</td>
<td>fractured_clean</td>
</tr>
<tr>
<td>CLEAN_GROUT</td>
<td>T121-T160</td>
<td>2113</td>
<td>100000</td>
<td>4.10E-05</td>
<td>4.10E-05</td>
<td>0.58</td>
<td>2.405</td>
<td>fractured_clean</td>
</tr>
<tr>
<td>FLOOR</td>
<td>T101</td>
<td>0</td>
<td>50</td>
<td>1.62E-06</td>
<td>1.62E-06</td>
<td>0.12</td>
<td>2.545</td>
<td>fractured_floor</td>
</tr>
<tr>
<td>FLOOR</td>
<td>T102</td>
<td>50</td>
<td>100</td>
<td>4.86E-06</td>
<td>4.86E-06</td>
<td>0.12</td>
<td>2.545</td>
<td>fractured_floor</td>
</tr>
<tr>
<td>FLOOR</td>
<td>T117</td>
<td>1250</td>
<td>1404</td>
<td>8.59E-05</td>
<td>8.59E-05</td>
<td>0.12</td>
<td>2.545</td>
<td>fractured_floor</td>
</tr>
<tr>
<td>FLOOR</td>
<td>T118-T160</td>
<td>1404</td>
<td>100000</td>
<td>9.11E-05</td>
<td>9.11E-05</td>
<td>0.12</td>
<td>2.545</td>
<td>fractured_floor</td>
</tr>
<tr>
<td>ROOF</td>
<td>T101</td>
<td>0</td>
<td>50</td>
<td>9.07E-07</td>
<td>9.07E-07</td>
<td>0.136</td>
<td>2.558</td>
<td>fractured_roof</td>
</tr>
<tr>
<td>ROOF</td>
<td>T102</td>
<td>50</td>
<td>100</td>
<td>2.71E-06</td>
<td>2.71E-06</td>
<td>0.136</td>
<td>2.558</td>
<td>fractured_roof</td>
</tr>
<tr>
<td>ROOF</td>
<td>T115</td>
<td>950</td>
<td>1106</td>
<td>3.71E-05</td>
<td>3.71E-05</td>
<td>0.136</td>
<td>2.558</td>
<td>fractured_roof</td>
</tr>
<tr>
<td>ROOF</td>
<td>T116-T160</td>
<td>1106</td>
<td>100000</td>
<td>4.10E-05</td>
<td>4.10E-05</td>
<td>0.136</td>
<td>2.558</td>
<td>fractured_roof</td>
</tr>
<tr>
<td>WALL</td>
<td>T101-T160</td>
<td>0</td>
<td>100000</td>
<td>7.60E-05</td>
<td>4.10E-05</td>
<td>0.35</td>
<td>2.631</td>
<td>CoBackfill</td>
</tr>
</tbody>
</table>
Example: Independent Model Input Summary Graphics

![Graph showing time vs. value for different cases: Case_sa_SALTSTONE_Kv, Case_007_SALTSTONE_Kv, and Case_008_SALTSTONE_Kv.](image)
Example: GoldSim and Porflow Benchmarking

- GoldSim model is an abstracted (simplified) version of Porflow model for UQ/SA
- Independently developed models

Updates to the H-Area Tank Farm Stochastic Fate and Transport Model, SRR-CWDA-2014-00060, Rev. 1, Aug 2015
Quality Assurance (QA) Risks and Remedies

Risks

• Assuming software QA testing addressed the full range of capabilities and conditions
• Assuming software and underlying conceptual models are valid analogues of physical reality
• Assuming good Input will always, or ever, produce good Output

Remedies

• Apply some level of “QC” to each Output
  – e.g. cursory look at every Output
• Code-to-code benchmarking
• Prototypic experiments, field observation, natural analogues
Albert Einstein

“An experiment is something everybody believes, except the person who made it.”

“A theory is something nobody believes, except the person who made it.”
"A model is something nobody believes, except the modeler who developed it."
Quality Control (QC) Risks and Remedies

Risks

• Assuming custom software … that produced good Output before … will for the next application

• Assuming software and underlying conceptual models are valid analogues of physical reality

Remedies

• Standardize certain pre- and post-processing tools and bring them under formal software QA control

• Avoid complacency in Each Time “QC” checking of model Output

• Prototypic experiments, field observation, natural analogues
Summary Thoughts

- Models are more than Codes
- Model QA is more than Code QA
- Model Output QC (Check Every Time) is a necessary complement to Software QA (Check Once / Use Many)
- Modelers tend to be overconfident in software tools and conceptual models
- Valuable components of Model and Analysis QA
  - Data!
  - Independent modeling efforts
  - Alternative conceptual models
  - Code-to-code benchmarking
  - Skeptical and dedicated expert reviewers

Your thoughts and experiences?