Software Development Practices for Climate Models:
What we’ve learned
Steve Easterbrook
University of Toronto & Met Office Hadley Centre

Contents

This presentation covers the following areas

• Current UM Development Practices

• Key issues:
  • Code Management & Coordination
  • Validation and Verification
  • Collaborations with other labs

• Risks and Opportunities

• Future Work
"Get the idea into working software as painlessly as possible"

"Get the working software to run on the available hardware as efficiently as possible"

Assessing Software Quality

Quality in Use
(Does it suit its intended purpose?)

External Quality Attributes
(Does it pass all the tests?)

Internal Quality Attributes
(Is it well-designed?)

Process Quality
(Are we following best engineering practice?)
Hadley study: initial questions

- Correctness
  - How do scientists assess “correctness” of the code in their models?

- Reproducibility
  - How do they ensure experiments can be reproduced?

- Shared Understanding
  - How do they develop and maintain a shared understanding of large complex codes?

- Prioritization
  - How do they prioritize their work?

- Debugging
  - How do they detect (and/or prevent) errors in the software?

Philosophical Status of Climate Models

- Climate is a complex system

- Sources of Uncertainty:
  - Measurement Error
  - Variability in the physical processes
  - Model imperfections

- Imperfection of models is routinely accepted
  - Many different types of model
  - Many choices of resolution, timescale, science
  - Scientists continually select their abstractions
Quality = Fitness for Purpose

- Purpose of Earth System Models:
  - “To test our understanding”
  - “To quantify uncertainty”

All models are wrong, but some are useful
- George Box

Summarizing skill gain
Taylor Diagrams

Time Scales

(Source: McGuffie & Henderson-Sellers, 2005)
Some Conflicting Goals

- Same code used for Weather Prediction and Climate Research
  - NWP: Must be fast, give accurate forecasts
  - CR: Must be fast, reproducible, scientifically valid
- Components with different origins:
  - developed in-house (tightly controlled)
  - consortium models
  - community models (cf open source)
- Code Forking

Code Management Tools

- Subversion - version tracker
- Trac - simplified bug tracker, wiki, and source browser
- Xxdiff - graphical diff and code merge
- Custom UI - simplifies process for branch and merge
- Custom Fortran 9X build system - simplified script to generate makefiles
- Custom code extract system - merge code from different branches and external sources
Coordination “informalisms”

- Core GCM developed in-house
  - Transplanted modules (e.g. MOM) are ‘naturalized’
  - (Is this changing with NEMO, UKCA, Jules,…)
- Single, large, open plan office environment
- Many communication channels:
  - wiki, newsgroups, email, Trac
  - ‘talk to the expert’
  - cross-functional, interdisciplinary teams
  - open meetings, workshops, etc.

Team Organisation
Verification and Validation

- Desk checking
  - Informal unit test, occasional (rare) use of debuggers

- Science Review and Code Review
  - Science review by project managers
  - Code review by designated code owners

- Continuous testing as Science Experiments

- Automated test harness on main trunk

- Bit Reproducibility as strong constraint

Continuous Integration Testing
e.g. Testing HadGEM3

<table>
<thead>
<tr>
<th>N00L35</th>
<th>N00L70</th>
<th>N14L35</th>
</tr>
</thead>
<tbody>
<tr>
<td>atm</td>
<td>atm</td>
<td>atm</td>
</tr>
<tr>
<td>coupled</td>
<td>coupled</td>
<td>coupled</td>
</tr>
<tr>
<td>test</td>
<td>test</td>
<td>test</td>
</tr>
<tr>
<td>ch1</td>
<td>ch1</td>
<td>ch1</td>
</tr>
<tr>
<td>biogenic &amp; sulphur on</td>
<td>biogenic &amp; sulphur on</td>
<td>biogenic &amp; sulphur on</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ch2</th>
<th>ch2</th>
<th>ch2</th>
</tr>
</thead>
<tbody>
<tr>
<td>package A conversion - excluding conversion</td>
<td>package A conversion - excluding conversion</td>
<td>package A conversion - excluding conversion</td>
</tr>
<tr>
<td>diagnosis</td>
<td>diagnosis</td>
<td>diagnosis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ch3</th>
<th>ch3</th>
<th>ch3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(MUCI)</td>
<td>(MUCI)</td>
<td>(MUCI)</td>
</tr>
<tr>
<td>(new boundary layer solver)</td>
<td>(new boundary layer solver)</td>
<td>(new boundary layer solver)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ch4</th>
<th>ch4</th>
<th>ch4</th>
</tr>
</thead>
<tbody>
<tr>
<td>v25h</td>
<td>v25h</td>
<td>v25h</td>
</tr>
<tr>
<td>improved convection diagnosis</td>
<td>improved convection diagnosis</td>
<td>improved convection diagnosis</td>
</tr>
<tr>
<td>chol</td>
<td>chol</td>
<td>chol</td>
</tr>
<tr>
<td>(refined CCA diagnostic)</td>
<td>(refined CCA diagnostic)</td>
<td>(refined CCA diagnostic)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ch5</th>
<th>ch5</th>
<th>ch5</th>
</tr>
</thead>
<tbody>
<tr>
<td>l6/3</td>
<td>l6/3</td>
<td>l6/3</td>
</tr>
<tr>
<td>(include L6/3 mode in L35)</td>
<td>(include L6/3 mode in L35)</td>
<td>(include L6/3 mode in L35)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ch6</th>
<th>ch6</th>
<th>ch6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch7</td>
<td>ch7</td>
<td>ch7</td>
</tr>
<tr>
<td>v25f</td>
<td>v25f</td>
<td>v25f</td>
</tr>
<tr>
<td>(includes L70 states)</td>
<td>(includes L70 states)</td>
<td>(includes L70 states)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ch8</th>
<th>ch8</th>
<th>ch8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch9</td>
<td>ch9</td>
<td>ch9</td>
</tr>
<tr>
<td>chang</td>
<td>chang</td>
<td>chang</td>
</tr>
<tr>
<td>(making up-to-date with HadGEM3-A freezes)</td>
<td>(making up-to-date with HadGEM3-A freezes)</td>
<td>(making up-to-date with HadGEM3-A freezes)</td>
</tr>
</tbody>
</table>


---

‘Virtual’ lab notebook

HadGEM3-A agy yi

Author: Dan Copsey
Owner: Dan Copsey
Predecessor: HadGEM3-A agy yi
Description: Using a large pole filtered zone to try and stop blow ups at 87S.
Current status: Complete
Model start times: From 1 Sep 1978 to 30 Dec 1988 (10 years and 4 months)
Details of changes with respect to predecessor:

Larger polar filtered zone (Terry Daweke):
- See by leed013 diffusion/compressor filter start latitude: 87 -> 85

Notes

Online validation notes and data
10 year short validation note comparing this job with its predecessor is here: http://www. np-lead013.com/lead013/logcat. html

Results of this model versus its predecessor
- Increased sea level pressure over Antarctica in DJF, improving the model.
   Associated with this is reduced surface wind stress in the southern ocean, also improving the model.
- Increased 200 hPa air temperature over Antarctica in DJF, improving the model.

Back to HadGEM3Evolution
- Main Dan/Copsey - 16 May 2008
Model inter-comparison

• Informal model comparisons
  • Used for diagnosing modeling errors
• Model Inter-comparison Projects (MIPs)
• Model Ensembles
  • Models from different labs on a common scenario
  • Variants of a single model to compare schemes
  • Perturbed physics ensembles
  • Single model with varied initial conditions

The Good News
What works, and why it works
Growth (in functionality?) isn’t slowing

Comparators

Financial Software System (Logica)

Open Source Software (Linux Kernel)

(Source: Lehman et al, 2000)

(Source: Godfrey & Tu, 2000)
Software “defect rates”

Some comparisons:

Worst military systems: 55 faults/KLOC
Best military systems: 5 faults/KLOC
“Extreme Programming”: 1.4 faults/KLOC
Apache (open source): 0.5 faults/KLOC
NASA Space shuttle: 0.1 failures/KLOC

Unified Model:

avg of 24 “bug fixes” per release
avg of 50,000 lines edited per release
⇒ 2 defects / KLOC make it through to released code
⇒ expected defect density in current version: 24 / 830,000 = 0.03 faults/KLOC

A more detailed study
Few Defects Post-release

- Obvious errors:
  - Model won’t compile / won’t run
  - Model crashes during a run
  - Model runs, but variables drift out of tolerance
  - Runs don’t bit-compare (when they should)

- Subtle errors (model runs appear “valid”):
  - Model does not simulate the physical processes as intended (e.g. some equations / parameters not correct)
  - The right results for the “wrong reasons” (e.g. over-tuning)
  - Expected improvement not achieved

Critical Success Factors
Which aspects of practice contribute particularly to the successes
Key Success Factors

- Highly tailored software development process (software development is “doing science”)
- Single Site Development
- Software developers are domain experts
- Shared ownership and commitment to quality
- Openness (“Many eyes” validation)
- Benchmarking (e.g. MIPS & ensembles)
- Unconstrained Release Schedule

Highly Adapted Processes
### “Agile” vs “Sturdy”

<table>
<thead>
<tr>
<th>Agile</th>
<th>Sturdy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative ↔ Planned</td>
<td>Planned ↔ Iterative</td>
</tr>
<tr>
<td>Small increments ↔ Analysis before design</td>
<td>Analysis before design ↔ Small increments</td>
</tr>
<tr>
<td>Adaptive planning ↔ Prescriptive planning</td>
<td>Prescriptive planning ↔ Adaptive planning</td>
</tr>
<tr>
<td>Embrace change ↔ Control change</td>
<td>Control change ↔ Embrace change</td>
</tr>
<tr>
<td>Innovation and exploration ↔ High ceremony</td>
<td>High ceremony ↔ Innovation and exploration</td>
</tr>
<tr>
<td>Trendy ↔ Traditional</td>
<td>Traditional ↔ Trendy</td>
</tr>
<tr>
<td>Highly fluid ↔ Upfront design / architecture</td>
<td>Upfront design / architecture ↔ Highly fluid</td>
</tr>
<tr>
<td>Feedback driven ↔ Negotiated requirements</td>
<td>Negotiated requirements ↔ Feedback driven</td>
</tr>
<tr>
<td>Individuals and Interactions ↔ Processes and Tools</td>
<td>Processes and Tools ↔ Individuals and Interactions</td>
</tr>
<tr>
<td>Human communication ↔ Documentation</td>
<td>Documentation ↔ Human communication</td>
</tr>
<tr>
<td>Small teams ↔ Large teams</td>
<td>Large teams ↔ Small teams</td>
</tr>
</tbody>
</table>

### Use of Agile practices:

<table>
<thead>
<tr>
<th>Agile Practices</th>
<th>Sturdy Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Collective Ownership</td>
<td>✓ Process &amp; product quality assurance</td>
</tr>
<tr>
<td>✓ Configuration Management</td>
<td>✓ Project monitoring &amp; control</td>
</tr>
<tr>
<td>✓ Continuous Integration</td>
<td>✓ Project planning</td>
</tr>
<tr>
<td>✓ Feature-driven devl.</td>
<td>✓</td>
</tr>
<tr>
<td>~ Frequent small releases</td>
<td>✓ Refactoring</td>
</tr>
<tr>
<td>✓ Onsite customer</td>
<td>✓ Requirements management</td>
</tr>
<tr>
<td>~ Organization-wide process</td>
<td>~ Retrospective</td>
</tr>
<tr>
<td>~ Organizational training</td>
<td>✓ Risk Management</td>
</tr>
<tr>
<td>× Pair programming</td>
<td>✓ Simple design</td>
</tr>
<tr>
<td>× Planning game</td>
<td>✓ Tacit knowledge</td>
</tr>
<tr>
<td>✓ Peer reviews</td>
<td>× Test-driven development</td>
</tr>
</tbody>
</table>
Shared Conceptual Architecture

Comparison with Open Source Projects

- Release schedule not driven by commercial pressures
- Developers are domain experts
- Core group of code owners control trunk
- Community operates as meritocracy
- Developers have "day jobs" (as scientists!)
- V&V based on extensive use by developers themselves
Challenges and Opportunities
What needs fixing?

Challenges

- Improve coordination across code branches
- Coordination with external users
- Multi-site development
- Make model configurations easier to define and validate
- Improve access to model result datasets
Coordination and Shared Understanding

- Coordinating the teams is a major challenge
  - Keeping your branch up to date
  - Knowing what changes are happening elsewhere
  - Configuration dependencies and hand-edits
- Heavy reliance on informal communication
  - Problems solved by "knowing who to talk to"
- External users using “old” versions
- Other development sites use different processes

Summary & Lessons Learned
Key insights for Software Engineering and for Climate Science
Hadley study: initial questions

- How do scientists assess “correctness” of the code?
  - “correctness” → “model skill”
  - continuous re-assessment of modeling tradeoffs
- How do they ensure experiments can be reproduced?
  - Releases are frozen (no bug-fixes), configs archived
  - Bit reproducibility across architectures and versions
- How do they maintain a shared understanding?
  - Single site, agile practices, many informal comms channels
- How do they prioritize model developments?
  - Organic, bottom-up, code owners provide longer view
- How do they detect/prevent errors in the software?
  - Continuous integration testing, model intercomparisons,…

Future Work
Where next?

• Comparison with other Climate Modeling Centres:
  • e.g. CCCma (very small team)
  • e.g. NCAR (community model)
  • e.g. MPI-M, GFDL, IPSL, …

• Compare Validation processes with other scientific models
  • e.g. economics models used in climate policy
  • e.g. other environmental science models