A Better Understanding of the Effects of Software Defects in Weather Simulation

Dongping Xu
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
xdp@iastate.edu

Daniel Berleant
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
berleant@iastate.edu

Gene Takle
Department of Geological and Atmospheric Sciences
Iowa State University
Ames, IA 50011
gstakle@iastate.edu

Zaitao Pan
Department of Earth & Atmospheric Sciences
St. Louis University
St. Louis, MO 63103
panz@eas.slu.edu

ABSTRACT

We investigate the impact of bugs in a well-known weather simulation system, MM5. The findings help fill a gap in knowledge about the dependability of this widely used system, leading to both new understanding and further questions.

In the research reported here, bugs were artifially added to MM5. Their effects were analyzed to statistically understand the effects of bugs on MM5. In one analysis, different source files were compared with respect to their susceptibility to bugs, allowing conclusions regarding for which files software testing is likely to be particularly valuable. In another analysis, we compare the effects of bugs on sensitivity analysis to their effects on forecasting. The results have implications for the use of MM5 and perhaps for weather and climate simulation more generally.

1. MOTIVATION

Computer simulation is widely used, including in transportation, digital system design, aerospace engineering, weather prediction, and many other diverse fields. Simulation results have significant impact on decision-making and will continue to in the coming years. However complex simulation programs, like other large software systems, have defects. These adversely affect the match between the model that the software is intended to simulate, and what the software actually does. Simulation programs produce quantitative outputs and thus typify software for which bugs can lead to insidious numerical errors. Such faulty outputs can differ from what the model implies without appearing unreasonable. These incorrect results may escape notice even as they influence the decisions that they are intended to support. This is an important dependability issue for complex simulation systems. Hence, investigating the effects of bugs in a simulation system can illuminate the robustness of its outputs to the presence of bugs. This in turn yields better understanding of important quality issues, like trustworthiness of the outputs, and important quality control issues, like software testing strategy.

The artificial generation of bugs and observation of their effects is termed *mutation analysis*. We have done a mutation analysis of MM5, an influential weather simulator available from the National Center for Atmospheric Research (NCAR). The results obtained illuminate important aspects of this software system. We focus on (1) what sections of the code are most likely to have undetected bugs that cause erroneous results, and hence are particularly important to test thoroughly; and (2) the relative dependability of the software for sensitivity analysis as compared to point prediction, and consequently for forecasting vs. predicting the effects of interventions.

2. RELATED WORK

2.1 Sensitivity Analysis

2.2 Ensemble Forecasting

2.3 Software Mutation Testing

3. APPROACH

Two related studies were performed. In the first, simulation results were obtained from numerous variants of MM5. Each variant had a different mutation ("bug"), deliberately inserted into code within a selected subset of its Fortran source code files. The set of simulation results from these variants allows comparison of the abilities of the source code files to resist erroneous outputs despite the presence of bugs. That in turn has implications for software testing strategy.

In the second study, the same mutated variants were used but the initial conditions were slightly different. The simulation results for this perturbed initial state were compared to the results obtained in the first study for each variant. This gives evidence about whether sensitivity analyses, in which the ratio of output change to input change is computed, tend to be more affected or less by bugs compared to forecasts, in which a single weather prediction scenario is computed. If less, the dependability of MM5 for studies related to intervention, which for the time scale investigated would be weather modification, is increased relative to its dependability for forecasting. If more, MM5 would be relatively more dependable for forecasting.

3.1 Study #1: Effects of Bugs on Forecasts

Over 7,000 different mutations were tested. For each, the MM5 weather simulator was compiled and run using a typical American midwest weather scenario for initialization. Effects of the mutation on the final state of a 24-hour forecast were recorded. Each mutation was then classified into one of three categories (Figure 1).

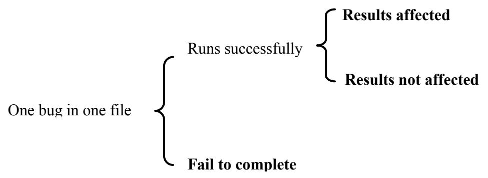

Figure 1. The three possible effects of a mutation (bolded).

The “Fail to complete” category includes cases where (1) the simulator terminated (“crashed”) prior to providing a 24-hour forecast, or (2) the simulator did not terminate. The “Results affected” category includes cases in which one or more members of a set of 11 important output parameters had a different forecasted value than it had in the original, unmutated version of MM5.

Each source code file c in which mutations were made was analyzed as follows. Define

r_c = # of mutations in the “Results affected” category; and
f_c = # of mutations in the “Fail to complete” category.

A dependability metric, d_c, for rating source code file c was defined as

$d_c=f_c/(r_c+f_c)$.

Value d_c estimates the likelihood that a bug inadvertently introduced during software development will be detected, therefore removed, and hence not affect results subsequently. Thus low d_c for a file suggests a need to compensate with extra effort in testing and debugging.

The “Results not affected” category, which does not influence d_c, contains mutations for which (1) the mutated code cannot have an effect (meaning the code, despite its presence, has no function), (2) the mutated code would fall into one of the previous two categories given other initial conditions, or (3) the mutation affected some output but not the output parameters we examined for effects. Mutations in case (1) do not impact the dependability issues of interest here, and therefore are ignorable. Of mutations in case (2), there is no reason to expect that their effects, when triggered by other initial conditions, would cause other than random changes to the various d_c. Those in case (3) are in a gray area. If their effects can be considered insignificant relative to effects on the output parameters that were analyzed, they can be ignored. Otherwise, had they been detected, d_c would have been lower. Thus the d_c values calculated in this work are upper bounds relative to an alternate view of the software outputs that classifies all outputs as equally important.

3.2 Study #2: Comparing Effects on Forecasts to Effects on Sensitivity Analyses

The results of study #1 provide data on the effects of bugs on forecasts. By introducing a perturbation to the initial data and then obtaining data parallel to the data of study #1, there will be two sets of data that can be compared. The perturbation in the initial data can be summarized as a number Δi. The resulting change in the output of the *unmutated* software can be summarized as a number Δo.

For each mutated version m that runs to completion under the two input conditions, there are two corresponding output scenarios whose difference can be summarized as a number Δo_m. Sensitivities (changes in outputs divided by changes in inputs) can now be calculated.

Let s be the sensitivity of the original, unmutated software:

$s=\Delta o/\Delta i$.

Let s_m be the sensitivity of the software as modified by mutation m:

$s_m=\Delta o_m/\Delta i$.

To compare the effect of a mutation m on forecasting to its effect on sensitivity analysis, we must define measures for the magnitudes of its effects on forecasting and on sensitivity analysis, and compare those measures. We define the magnitude of its effect on *forecasting* as

$$F_m = \frac{|o_m - o|}{o}$$

where o_m is a number summarizing output parameters of the software as modified by mutation m, and o is an analogous number for the unmutated software. Thus, F_m describes the change in the forecast due to mutation m, as a proportion of the nominally correct output o.

The magnitude of mutation m’s effect on *sensitivity* is analogously defined as

$$S_m = \frac{|s_m - s|}{s}$$

where s and s_m are as defined earlier.

For a given mutation, if $F_m>S_m$, then the mutation affected the forecast more than the sensitivity analysis. On the other hand, if $F_m<S_m$ then the opposite is true. Considering the mutations m collectively, if $F_m>S_m$ for most of them, this suggests that the MM5 software resists the deleterious effects of bugs on sensitivity analysis better than it resists their effects on forecasting. On the other hand, if $F_m<S_m$, that suggests the opposite.

The uses of sensitivity analysis include predicting the effects of interventions on weather and climate. Thus study #2 provides data on the relative dependability of MM5 for use in forecasting vs. interventions. The remainder of this paper gives many more details concerning the two studies, followed by some caveats and directions for further investigation.