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 artificially 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. Weather simulators exemplify this danger because of the large amount of code they contain, often with insufficient comments; the complex interactions among sections of code modeling different aspects of the domain; and the plausibility that outputs can have even if they contain significant error. 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. sensitivity analyses.

2. CONNECTION TO RELATED WORK

Model Error. Errors in a simulation program can be from errors in the model underlying the system [1], or errors in the implementation of a correct underlying model. The current work deals with the latter, but the two are related.

Software Mutation Testing. This is the process of taking a software system and injecting errors (i.e. adding bugs) to it to see how its behavior changes [4][5][6]. Typically, many different buggy versions will be tested and the responses summarized statistically. The present work tests several thousand versions of MM5, each with one unique bug injected.

Sensitivity Analysis. A software system's sensitivity $s = \frac{\Delta o}{\Delta i}$ is the amount of change in its output Δo that occurs due to a change in input by amount Δi [2][3]. In contrast, examining the change in output due to changes in actual software code (rather than in input parameters) is termed software mutation testing. The work reported here uses both, because one of the questions we focus on is, "How do software mutations affect sensitivity analyses?"

Ensemble Forecasting [7]. A sophisticated form of sensitivity analysis in which the system is run many times, each on its own set of perturbed input parameters, and each therefore giving its own forecast as output. The set of forecasts is then statistically characterized. This enables conclusions such as forecasts that are relatively resistant to inaccurate specification of initial conditions (using *ensemble means*), and forecasts containing distribution

functions describing the probabilities of different values for a forecasted quantity.

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 24-hour forecasts produced by these variants for region of the U.S. midwest with a time step of 4 minutes 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 affected by bugs less or more compared to forecasts (in which a single weather prediction scenario is computed). If less, the dependability of MM5 for studies using sensitivity analysis 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

Several thousand 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 the forecast were recorded. Each mutation was then classified into one of three categories (Figure 1).

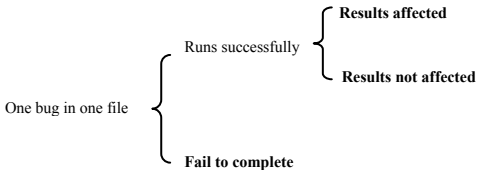

Figure 1. The three possible effects of a mutation (bolded).

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 has no influence on 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 derived from the 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.

Meteorological uses of sensitivity analysis include predicting the effects of interventions on climate, and doing ensemble forecasting. Because the present study relied on 24-hour forecasts, study #2 provides data relevant to ensemble forecasting. The next section summarizes the results.

4. RESULTS

Results related to study #1 on the effects of bugs on forecasts are given in section 4.1. Results related to study #2, which builds on study #1 with additional investigation of sensitivity analyses are given in section 4.2. Some caveats are given in section 4.3.

4.1 The Effects of Bugs on Forecasts

It is useful to be able to compare source code files based on the likelihood of each that an important bug in it will be readily detected. Such comparisons would help focus software testing activities on files for which undetected bugs are most likely to reside. This motivated defining the metric d_c for the dependability of source code file c in section 3.1.

Low values of d_c mean that a bug is relatively likely to allow a simulation to complete but with error in its output. High values of d_c mean that a bug is relatively likely to cause the program to crash without giving output. Thus, a low value of d_c indicates that file c is relatively likely to contain undetected bugs and therefore that file c is a good candidate for careful testing to find bugs. Values of d_c for a number of important files in MM5 are shown in Figure 2.

Conclusion: of the files tested, bugs in exmoiss.f, hadv.f, init.f, mrfpbl.f, param.f, and vadv.f are more likely than bugs in the others to have insidious rather than obvious effects. Hence these files might be expected to benefit from correspondingly thorough testing.

4.2 Effects on Forecasts Vs. Effects on Sensitivity Analyses

MM5 and other weather and climate simulation systems are useful for both forecasting and for sensitivity analysis. It is therefore an interesting question about MM5 which, forecasts or sensitivity analysis results, are more resistant to bugs which, as noted earlier, are undoubtedly present.

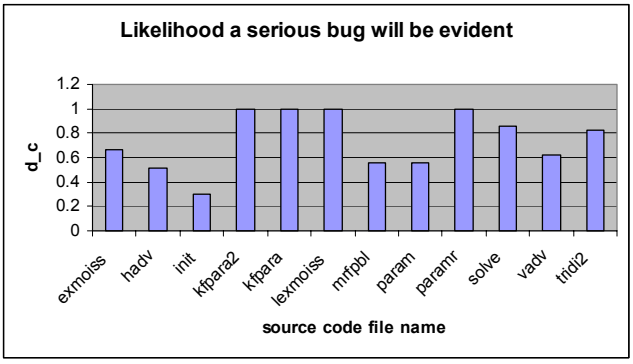

Figure 2. Values of d_c for some MM5 source code files. Lower values suggest files that are good candidates for focusing testing effort on.

To answer this question, the metrics described in section 3.2 were used. Instead of a composite number summarizing 11 output parameters, for this study we used 8 output parameters and analyzed each separately. Because a total of 10,893 different mutations were tested on both the base and perturbed inputs, and 8 output parameters were observed for each mutation, a total of 87,144 different sensitivities (i.e., values of S_m as defined in section 3.2) were observed. Similarly, the same number of forecasted parameter values (i.e., values of F_m as defined in section 3.2) were observed. Although most were unaffected by the mutation, 12,835 were affected (first two rows of Table 1).

Perturbation to the input conditions was done as follows. Some variables in the file init.f were changed by 0.0001%. The variables chosen for this were the prognostic 3D variables (UA, UB, VA, VB, TA, TB, QVA, QVB) for each grid in the domain (these variables are described e.g. in http://www.mmm.ucar.edu/mm5 /documents/mm5-code-pdf/sec6.pdf.). The percentage of perturbation was chosen to be close to the smallest percentage for which the program produced significant changes to the output.

For many mutations and observed output parameters, the sensitivities and forecasted values of an observed output parameter r were exactly the same as for the unmutated program, so for each such mutation m and parameter r, $F_m(r)=S_m(r)$. For other mutations and parameters, the change caused by that mutation to the forecast was *greater* than the change to the sensitivity. For those, $F_m(r)>S_m(r)$. Finally, for the remaining mutations and parameters, the situation was reversed and $F_m(r)<S_m(r)$. In order to determine whether MM5's forecasts or sensitivity analyses were more resistant to bugs, we simply compare the

quantity of parameter/mutation pairs for which $F_m(r)>S_m(r)$ to the number for which $F_m(r)<S_m(r)$. If more pairs have $F_m(r)>S_m(r)$ than have $F_m(r)<S_m(r)$, then forecasts are more likely to be affected by bugs than sensitivity analyses and therefore MM5 is observed to be more dependable for sensitivity analyses than forecasts. If fewer pairs, then the opposite is observed: MM5 would be observed to be more dependable for forecasts. Results will be presented at the meeting.

4.3 Details, Caveats, and Needs for Further Work

Study #1. The results of this study on the effects of bugs on forecasts required mutations to be applied to a number of different source files. A number of different types of mutations were applied, each designed to be plausible as a kind of bug a human programmer might accidentally make. As examples, loops can suffer from "off-by-one" bugs, additions can be programmed into calculations when subtractions should be, multiplication and division can be incorrectly substituted similarly, and so on. Table 2 shows the types of mutations that were used. Each was applied opportunistically to a source file at each point in it where the source code would permit such a mutation. Thus, for example, off-by-one bugs can be applied to points in the code where loop control variables were tested.

This process of using a number of different, seemingly plausible bug types leads to two caveats.

1) The proportion of each bug type in one source file may not match the proportion in another source file. The question this leads to is whether differences observed across source code files (Figure 2) could be due in part to differences in the effects of mutations across different bug types. (Similarly, differences in effects of different bug types could be due in part to differences across the source files containing them.) Appropriate statistical analyses should be able to separate the effects due to source code file from those due to bug type.

2) Although the bug types used have intuitive appeal as mistakes that human programmers might make, there is no claim that all such mistakes are captured by the set of bug types used for mutations in this (or any) work. In particular, humans can make diffuse mistakes that cover a number of lines of code, and these are hard to mimic when generating mutations artificially. Mutation analyses have historically assumed that automatically generated mutations are similar in their effects to human programmer errors, however.

Another limitation of the study is its reliance on a single weather forecasting scenario. While within the range of typical forecasting problems, it is possible that other initial conditions could lead to different results for the dependability metrics of the source files. This could in principle be addressed by seeing if similar results are obtained for a set of diverse forecasting problems.

Study #2. This study comparing the ability of sensitivity analyses and forecasts to resist the effects of bugs relied on a particular perturbation to the input conditions, a particular weather scenario (as in the other study), and a particular time period of 24 hours. Each of these may potentially have an influence on the results.

The perturbation to the input conditions was chosen to be small in order to stay within the linear response region of the simulation system. However weather simulation is well-known to be mathematically chaotic. Thus, there may be legitimate doubt about whether in fact this experiment did stay within the linear response region (or even if trying to do so is worth doing). The questions this raises is whether different input perturbations might lead to different assessments of which resists bugs better, weather forecasts or sensitivity analyses. The solution here is more extensive testing that includes and compares different input perturbations.

The relative abilities of forecasts vs. sensitivity analyses to resist bugs may also potentially depend on the time period of the simulation. While 24 hours incorporates both day and night, thereby exercising varied portions of the system, other time periods are also of interest. This suggests additional testing that incorporates a range of different time periods.

Finally, as in the other study, more extensive testing could usefully include different weather scenarios.

5. REFERENCES

[1] Allen, M.R., J.A. Kettleborough and D.A. Stainforth, 2002: Model error in weather and climate forecasting, *Proc. 2002 ECMWF Predictability Seminar*, ECMWF, Reading, UK

[2] Alpert, P., M. Tsidulko and U. Stein, 1995: Can sensitivity studies yield absolute comparisons for the effects of several processes? *J. Atmos. Sci*., **52**: 597-601.

[3] Berleant, D. and B. Liu, 1997: Is sensitivity analysis more fault tolerant than point prediction? *Simulation in the Medical Sciences: Proceedings of the 1997 Western MultiConference,* pp. 196-199.

[4] Fenton, N.E. and M. Neil, 1999: A critique of software defect prediction models, *Software Engineering* **25** (5): 675-689.

[5] Madeira, H., D. Costa, and M. Vieira, 2000: The emulation of software faults by software fault injection. In *Proceedings of the International Conference on Dependable Systems and Networks*, IEEE, pp. 417-426.

[6] Voas, J. and J. Payne, 2000: Dependability certification of software components, *Journal of Systems and Software* **52** (2-3): 165-172.

[7] Workshop on Ensemble Weather Forecasting in the Short to Medium Range, Quebec, Sept. 18-20, 2003. http://www.cdc.noaa.gov/~hamill/ef_workshop_2003.html.