Impact of modern software engineering practices on the capabilities of an atmospheric general circulation model

Ravi S. Nanjundiah* and U. N. Sinha†

*Centre for Atmospheric and Oceanic Sciences, Indian Institute of Science, Bangalore 560 012, India

†Flosolver Unit, National Aerospace Laboratories, Kodihalli Campus, Bangalore 560 017, India

In this note, we discuss the relevance and impact of a software engineering effort at NAL on the forecast model in operation at the National Centre for Medium Range Weather Forecasting (NCMRWF). The code has been re-written exploiting the features of Fortran 90. As a direct consequence of appropriate reengineering efforts on the code, it is both easy to comprehend and modify. The reengineered code is appreciably shorter (number of lines in the code reduced by 55%) and can run on a variety of computing platforms including a PC, without the need for any further modifications to the code.

A General Circulation Model (GCM) used for numerical weather prediction is among the most complex computer codes from the view-point of mathematics as well as software engineering. Mathematically, a GCM encompasses an ensemble of sub-models to describe various physical processes (as different as radiation and turbulence) that occur simultaneously within the atmosphere. From the view-point of software engineering, all these diverse sub-models need to be integrated correctly and perform in unison to generate a forecast. Therefore, numerical weather prediction (with the associated atmospheric modelling) has been considered as one of the grand challenge problems for high performance computing. Hence it is not surprising that all over the world, the most powerful computers have always been used by weather forecasting centres. Most weather forecasting software codes presently in use were originally developed in the mid 1980s when memory and computational power were at a premium, compilers were not as highly developed as the present day ones and software engineering (with emphasis on ease of usage) was not considered a major issue. The emphasis therefore was on exploiting the power of the then available high performance computing platforms. Most present-day weather forecasting codes have evolved from those developed in the 1980s and still retain most features of their original codes from a software engineering view-point. In the process, such codes used for weather forecasting lack transparency and are very cryptic and difficult to comprehend, thus making modification and experimentation extremely difficult. This has resulted in numerical modelling of the atmosphere to become the exclusive domain of a few select centres worldwide (and even fewer within the country). However, with the improved power of microprocessors, the evolution of more powerful programming languages and a better appreciation of modern software engineering techniques, it is feasible to reengineer this traditional high performance computing application to make the code more comprehensible, facilitate research and development and to exploit the burgeoning power of microprocessors. In this note we discuss the impact of applying modern software engineering practices to GCMs, with specific reference to the GCM used by the National Centre for Medium Range Weather Forecasting (NCMRWF), New Delhi.

The model used at NCMRWF has a spectral resolution of T-80 (eighty modes with triangular truncation) and has 18 levels in the vertical. It integrates the equations of mass, momentum and energy conservation (in addition to equations specifying boundary conditions) to generate forecasts for the entire globe. Details of this model are discussed in by Kalnay *et al.*[1]. It was provided to the National Aerospace Laboratories (NAL), Bangalore, in 1993, as part of a national initiative on use of parallel computers for weather forecasting. The original code was successfully implemented in the parallel mode on the Flosolver Mk3 (ref. 2). Subsequently, another version of the model was successfully implemented on the SP2 parallel computer in the Super computer Education and Research Centre (SERC) at the Indian Institute of Science (IISc)[3] and successfully used in a climate mode (i.e. for simulations extending over time period of several years).

During these efforts it was strongly felt that the code lacked transparency and was very difficult to comprehend and experiment with, and therefore not suitable as a research tool. This lead to a major effort in re-engineering of the code exploiting features of Fortran 90 (ref. 4).

The objectives of the reengineering project were: (i) to make the code transparent and easy to comprehend and modify; and (ii) to remove limitations of coding practices due to Fortran 77 by rewriting it in Fortran 90.

The procedure followed to attain these objectives can be broadly summarized as follows:

1. Redundant code was removed, especially in the repetitive application of the same code for computing different quantities.
2. Common blocks, the bane of traditional Fortran programming were eliminated. Memory allocation for global variables was made modular.
3. For optimal memory utilization, the Fortran 90 features of dynamic allocation and deallocation of memory, and pointers (not available with older versions of Fortran) were exploited.
4. Iterative statements using hard-coded numbers (for specifying the range) were replaced with statements incorporating variables/parameters as the range-specifiers.
5. Conditional statements (IF and GO TO) were simplified. An example of the differences between old and new codes is shown in Appendix A.
6. Names of variables/routines were made transparent and comprehensible so that their functionality becomes clear. The older version of the code used the older Fortran standard of utilizing only seven characters to identify a variable which resulted in the variable/procedure names to be terse and cryptic. A typical example of this is given in Table 1.

The same array (SYN) was used for divergence, temperature and log(surface pressure). In the new version of the code, the names have been made more descriptive. The new name Di_Fourier clearly suggests that this variable is used for storing divergence in the Fourier–latitude space.

As a result of this effort, *the rewritten code reduced from 40,000 lines to 18,000 lines*. This reduction occurred due to the fact that similar (but not identical) computations (such as summation of spectral coefficients, etc.) were conducted in several routines. Due to the reengineering efforts, these could be consolidated into fewer routines and the size of the code was thus reduced. The modular structure of memory assignment also helped in shrinking the size of the code.

The reengineered code should have a significant impact on the atmospheric modelling. The significant implications are:

- We have a code that is completely portable across platforms. The reengineered code has been successfully tested on diverse platforms such as the IBM RS6000/595 (the basic processor for the IBM SP2), the MIPS R10000 (the processor used in SGI Origin) and a PC based on Pentium II. It is noteworthy that no changes were required in the code to run on any of these platforms.
- Using PCs, almost any college or university can now offer training in numerical weather modelling. Therefore, the lack of computational infrastructure will no longer be a major bottleneck for manpower training. As a consequence a larger pool of trained manpower can be generated.

Table 1. Comparison of names of variables in the old and new versions of the code

Old name	New name
SYN	Di_Fourier
SYN	Te_Fourier
SYN	Q_Fourier

- We have a powerful research tool. Unlike the older version of the NCMRWF model, the reengineered code is easy to comprehend and modify. The resolution can be changed by changing just three lines in a single memory module. In contrast, the original NCMRWF model required more than hundred changes in different segments of the code (and one could not still be sure that all the required changes had been incorporated). Additionally, the hard-coding in spectral transform computations made it almost impossible to increase resolution. On the contrary, the reengineered code can now be used for weather forecasting (high resolution, short integrations), climate modelling and simulation (medium resolution, long integrations) and can be an ideal ‘hands on’ tool in classroom teaching (very low resolutions).
- Ease of modification can facilitate research by different groups with diverse interests and seamlessly incorporate improvements into the code. In the Indian context, this could mean more cost-effective research in the field of monsoon modelling and simulation. Simulation of monsoon is considered to be one of the major challenging problems for GCMs, as no model has been able to capture all the features of the monsoon even on the seasonal scale[5]. Improving the monsoon simulation would require researchers from diverse fields such as radiation, turbulence and boundary layer modelling and numerical methods to incorporate their contributions to the model. The present reengineered model would be an ideal vehicle for such experimentation.

In conclusion, it appears that, if used properly, the reengineered code holds out the prospect of revolutionizing the way atmospheric modelling is conducted within the country.

Appendix A

An example of replacement of the old code by a more readable and transparent code is given below.

Old code:

```
DO 100 I = 1, IX

DW(I)=max(RWFL*SOLWT(I),0.)

DW(I)=min(DW(I),1.)


IF (SLMSK(I).NE.1.0) THEN

DW(I) = 1.0

ENDIF


IF((SLMSK(I).EQ.1.0).AND.(SNOCOV(I).GT.0.0)) THEN

DW(I) = 1.0

ENDIF

100 CONTINUE
```

New code:

```
Dw(:)=Rwfl*Solwt(:)

Where(Dw(:) < 0.0 )Dw(:)=0.0

Where(Dw(:) > 1.0 )Dw(:)=1.0

Where(Slmsk(:) /= 1.0)Dw(:)=1.0

Where(Slmsk(:)==1.0 .And. Snocov > 0.0 )Dw(:)=1.0
```

Another example where GO TO statments were removed leading to a greater transparency of the code was in the calculations of model dates (this small piece of code had 13 GO TO statements). The old and new versions of the code are shown below.

Old code:

```
IYEAR=IYR

NDAY=JD-JDOR

IF(FJD.GE..5 E 0) NDAY=NDAY+1

61 IF(NDAY.LT.1462) GO TO 62

NDAY=NDAY-1461

IYEAR=IYEAR+4

GO TO 61

62 NDIY=365

IF(MOD(IYEAR,4).EQ.0) NDIY=366


IF(NDAY.LE.NDIY) GO TO 65

IYEAR=IYEAR+1

NDAY=NDAY-NDIY


GO TO 62

65 IF(NDAY.GT.INT(DY(2))) GO TO 66

IM=1

ID=NDAY


GO TO 67

66 IF(NDAY.NE.60) GO TO 68
```

```
IF(NDIY.EQ.365) GO TO 68
IM=2
ID=29

GO TO 67
68 IF(NDAY.GT.(INT(DY(3))+NDIY-365)) GO TO 69
IM=2

ID=NDAY-31
GO TO 67
69 DO 70 I=3,12

IF(NDAY.GT.(INT(DY(I+1))+NDIY-365)) GO TO 70
IM=I
ID=NDAY-INT(DY(I))-NDIY+365
GO TO 67

CONTINUE
```

New code:

```
IYEAR=IYR

NDAY=JD-JDOR

IF(FJD >= 0.5) NDAY=NDAY+1
```

```
DO

IF(NDAY < 1462) EXIT

NDAY=NDAY-1461

IYEAR=IYEAR+4

ENDDO


DO

NDIY=365

IF(MOD(IYEAR,4) == 0) NDIY=366

IF(NDAY <= NDIY) EXIT

IYEAR=IYEAR+1

NDAY=NDAY-NDIY

ENDDO


If_Block1:&

IF(NDAY <= INT(DY(2)) ) THEN

IM=1

ID=NDAY

ELSEIF( NDAY .EQ. 60 .AND. NDIY .NE. 365) THEN

IM=2

ID=29

ELSEIF( NDAY <= INT( DY(3) ) + NDIY -365)THEN

IM=2

ID=NDAY-31

ELSE


DO I=3,12
```

```
IF( NDAY <= INT(DY(I+1))+NDIY-365)THEN

IM=I

ID=NDAY-INT(DY(I))-NDIY +365

EXIT

ENDIF

ENDDO



ENDIF &

If_Block1
```

1. Kalnay, E., Sela, J., Campana, K., Basu, B. K., Schwarzkopf, M., Long, P., Caplan, M. and Alpert, J., Documentation of the Research Version of the NMC Medium Range Forecast Model, 1998.
2. Sinha, U. N., Sarasamma, V. R., Rajalakshmy, S., Subramanium, K. R., Bhardwaj, P. V. R., Chandrashekhar, C. S., Venkatesh, T. N., Sunder, R., Basu, B. K., Gadgil, S. and Raju, A., *Curr. Sci.*, 1994, **67**, 178–184.
3. Nanjundiah, R. S. and Raju, A., Experiments in Parallel Implementation of the NCMRWF Model, Intromet '97, 2–5 December 1997, IIT New Delhi, 1997.
4. Flosolver Team, Status Report on Development of meteorological software at Flosolver. NAL report NAL PDFS 9816, Sept 1998.
5. Gadgil, S. and Sajani, S., *Clim. Dyn.*, 1998, in press.

ACKNOWLEDGEMENTS. We thank Prof. R. Narasimha for his suggestions.