Monoversion techniques for increasing the fault tolerance of software systems for monitoring the parameters of technological processes

V V Losev1, I V Kovalev1,2, M V Saramud1,2, N A Testoyedov1,4 and Ya A Tynchenko1

1 Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russia
2 Siberian Federal University, 79, Svobodny pr., Krasnoyarsk, 660041, Russia
3 Krasnoyarsk State Agrarian University, 90, Mira pr., Krasnoyarsk, 660049, Russia
4 JSC Academician M.F. Reshetnev «ISS» Zheleznyogorsk, Krasnoyarsk region, Russia

E-mail: basilos@mail.ru

Abstract. The article discusses the current monoversion techniques in solving the problem of increasing the fault tolerance of software. The subject of research is the system of environmental monitoring of the technological process. Model examples demonstrate the algorithmization of the programming approaches used. As the first technique, a programming technique using Checkpoint is presented. The second technique is the programming technique using Checksums. Both techniques are designed to increase the fault tolerance of the developed specialized software systems.

1. Introduction
We will consider one of the techniques aimed at increasing the fault tolerance of software, the checkpoint technique. It is a widely used technique to reduce program time overhead in the presence of failures. The essence of the technique is as follows. A checkpoint periodically captures the state of a running program. Thus, if a failure occurs, the program will resume from the most recent checkpoint, rather than from the beginning. In checkpoint schemes, the problem is divided into N intervals [1]. A checkpoint is added at the end of each interval, either by the programmer [2], or the compiler [3,4].

We will consider an example of model design of an environmental monitoring system using elements of the checkpoint methodology. The initial model uses inputs: $t, c, v$, where $t$ is the temperature of the flue gases in the places of sampling, $^\circ C$; $c$ is the quantity of substances and the content of suspended particles, kg/m$^2$; $v$ is the gas flow velocity, m/s. The technological object is a smoke removal system.

As components of the implementation of the checkpoint technique, consider the use of a scheduler, DRA (data re-expression algorithms) [5], comparators for detecting errors, and a data processing algorithm.

2. Checkpoint programming
At the entrance to the program, the scheduler sends data from the process parameters of the exhaust gases and suspended particles in three parameters to the filtration system and in three parameters after
the filtration system on the DRA for re-expression. Moreover, each parameter has three-point character of measurement, taking into account the peculiarities of the sampling location (figure 1). The border of the zero interval in the program is the place where the primary data from the sensors are received. It should be noted that the primary data are distributed in two streams with indices 1 - before the filtration system, 2 - after the filtration system.

Figure 1. The example of programming implementation by Checkpoint taking into account the measurements multi-point of monitoring parameters and the spatial distribution of sensors.
DRA launches its re-expression algorithms in the input receiving the following re-expressed inputs. At this stage, the averaged data of the technological process parameters are formed.

The scheduler collects the re-expressed input and initiates an error detection mechanism. For this, a comparator is used, which compares the values of the corresponding parameters of the technological process before and after filtration. The comparison condition is "Parameter values before filtering" > "Parameter values after filtering"

When the condition of the comparator is fulfilled, all data goes to the next stage of processing. The scheduler collects the re-expressed input and buffers it to prepare for further processing.

The buffer is built on the FIFO (First In First Out) principle and is required for sequential data processing by the algorithm.

As the algorithm there is provided the calculation of the modular ratio of the difference of the parameter measured and given to the given. The resulting expression is a formula term that expresses a generalized quality criterion characterizing the state of the technological process [5].

- **Step 1.** The scheduler collects the output and triggers the error detection mechanism again. For this, a signature comparator is used, which compares the obtained terms of the formula [5] with a given threshold value (for example, 0.01). Thus, exceeding the threshold value will signal either data corruption or an error. In any case, the scheduler transfers control to an earlier checkpoint. In case of repeated detection of an error, an earlier one. This stage is the boundary of the second interval of the checkpoint application scheme.

- **Step 2.** The output values of the algorithm are then summed up. And they form a generalized quality criterion that characterizes the state of the technological process. In this case, the number of terms is a variable parameter.

In general, for this method, both advantages and disadvantages can be noted:

*Advantages:*

- High performance with relatively simple error detection elements.
- The method can be aimed both at detecting errors and increasing the reliability of the data.

*Disadvantages:*

- Additional mechanisms for fixing the program state at checkpoints are required.
- Advanced scheduling mechanisms are required to ensure control and synchronization of the program.

**3. Checksums programming**

We will consider one of the techniques aimed at increasing the fault tolerance of software, the checksums technique.

Method, first proposed and described by the authors in [6, 7, 8] and characterizes not only detection, but also fault tolerance. The method is targeted at high performance applications. The approach uses code conversion rules to provide fault detection and, in most cases, fix faults [5].

We will consider a similar example of model design.

As components of the implementation of the checkpoint technique, consider the use of a scheduler, DRA (data re-expression algorithms), comparators for detecting errors, and a data processing algorithm.

At the entrance to the program, the scheduler sends data from the process parameters of the exhaust gases and suspended particles in three parameters on the corresponding the DRA for re-expression. In this model example, only one measurement plane (before filtering) is considered. Moreover, each parameter has three-point character of measurement, taking into account the peculiarities of the sampling location (figure 2).
The received primary data from the sensors of the technological process form a checksum using the binary coding algorithm (1).

\[ \text{chk}(t) = t \cdot 2^9 + t \cdot 2^8 + t \cdot 2^7 + t \cdot 2^6 + t \cdot 2^5 + t \cdot 2^4 + t \cdot 2^3 + t \cdot 2^2 + t \cdot 2^1 + t \cdot 2^0 \]  

(1)

Figure 2. The example of programming implementation by Checksums taking into account the measurements multi-point of monitoring parameters and the spatial distribution of sensors.
Pre-data - real numbers are converted to boolean data. Thus, nonzero data forms a logical one, data equal to zero logical zero. The converted data enters the checksum algorithm in an orderly manner. Thus, the data from each sensor 1 (0) has its place in the algorithm. This allows identification of the sensor and the incoming data in the checksum. The received checksum \( \text{chk} (0) \) is a reference for subsequent comparison with it.

DRA launches its re-expression algorithms in the input receiving the following re-expressed inputs. At this stage, the averaged data of the technological process parameters are formed. DRA block outputs are also converted to boolean numbers. After that, the checksum is formed by the binary coding algorithm.

The scheduler uses the received checksums of the two data sets and initiates an error detection mechanism. For this, a comparator is used, which compares the results of the obtained checksums. If the condition is met, i.e. checksums are equal, then the output data of the DRA blocks are sent for further processing. If the condition is not met, i.e. the checksums are not equal, then the scheduler determines the error by additional calculations:

\[
\text{chk}(0) = 511 = (11111111)_2; \\
\text{chk}(1) = 479 = (111011111)_2;
\]

\[ R_{1,2,1}(t) - \text{error}. \]

The scheduler then handles this exception. One option is to transfer control to a checkpoint. This happens provided that this mechanism is used in the program.

When the condition of the comparator is fulfilled, all data goes to the next stage of processing. The scheduler collects the re-expressed input and buffers it to prepare for further processing.

The buffer is built on the FIFO (First In First Out) principle and is required for sequential data processing by the algorithm.

As the algorithm there is provided the calculation of the modular ratio of the difference of the parameter measured and given to the given. The resulting expression is a formula term that expresses a generalized quality criterion characterizing the state of the technological process [5].

The scheduler collects the output and triggers the error detection mechanism again.

For this, a comparator is used, which compares the results of the received checksums (\( \text{chk} (1) \) and \( \text{chk} (2) \)). If the condition is met, i.e. the checksums are equal, then the output data of the algorithms goes for further processing. If the condition is not met, i.e. the checksums are not equal, then by additional calculations the scheduler determines the error:

\[
\text{chk}(1) = 511 = (111111111)_2; \\
\text{chk}(2) = 503 = (1111101111)_2;
\]

\[ f_{1,3,3}(v) - \text{error}. \]

The scheduler then handles this exception. One option is to transfer control to a checkpoint. This happens provided that this mechanism is used in the program.

The output values of the algorithm are then summed up. And they form a generalized quality criterion that characterizes the state of the technological process.

4. Conclusion

In general, for this method, both advantages and disadvantages can be noted:

**Advantages:**

- The method is applicable for detecting errors and addressing their source.
- High performance with relatively simple algorithms and error detection tools.

**Disadvantages:**
• Parametric information of the technological process is not analyzed, only monitoring of the availability of data from sources.
• Advanced mechanisms of the scheduler are required to ensure control and synchronization of the program.
• Additional mechanisms are required for fixing the program state at checkpoints (when using the checkpoint mechanism).

This research allows us to conclude that the classical approaches to improving software fault tolerance are not exhaustive. Moreover, there is a possibility of combining various software and algorithmic solutions. For example, the checksum programming method using binary coding allows to determine a sensor failure by the presence / absence of a signal (data). Thus, the resulting binary code can be unambiguously interpreted as a failure event. All this makes it possible to develop new approaches to the development of such a class of systems as systems for environmental monitoring and monitoring of technological processes in general.

Acknowledgments
The reported study was funded by Russian Foundation for Basic Research, Government of Krasnoyarsk Territory, Krasnoyarsk Regional Fund of Science to the research project: «Multiversion method for improving information reliability of environmental monitoring of thermal power plants» (№ 18-48-240007).

References
[1] Goloubeva O, Rebaudengo M, Reorda M S and Violante M 2006 Software-Implemented Hardware Fault Tolerance (Springer Science+Business Media LLC)
[2] Chandy K M and Ramamoorthy C V 1972 Rollback and recovery strategies for computer programs IEEE Transactions on Computers 21(6) 546-56
[3] Fuchs W K and Li C-C J 1990 CATCH - compiler-assisted techniques for checkpointing Proceedings of the 20-th International Symposium on Fault-Tolerant Computing (FTCS-20) 74-81
[4] Long J, Fuchs W K and Abraham J A 1992 Compiler-Assisted Static Checkpoint insertion 20-th International Symposium on Fault-Tolerant Computing (FTCS-22) 58-65
[5] Losev V V et al. 2020 The research of multiversion methods for improving the fault tolerance of software systems for monitoring the parameters of technological processes IOP Conf. Ser.: Earth Environ. Sci. 548(3) 032042
[6] Rebaudengo M, Sonza Reorda M and Violante M A 2003 New Software-based technique for low-cost Fault-Tolerant application IEEE Annual Reliability and Maintainability Symposium 25-8
[7] Rebaudengo M, Sonza Reorda M and Violante M 2004 A new approach to software-implemented fault tolerance JETTA: The Journal of Electronic Testing: Theory and Applications 20 433-7
[8] Cheynet P, Nicolescu B, Velazco R, Rebaudengo M, Sonza Reorda M and Violante M 2000 Experimentally evaluating an automatic approach for generating safety-critical software with respect to transient errors IEEE Transactions on Nuclear Science 47(6) 2231-6