useful information, because slice except initialization processes is not gotten even if elements of the array are selected as a condition of slicing. However, TVIS can show useful information for programmers to grasp behavior of a program, because it shows the situation of each variable and loop by the diagram.

Breakpoint is one of the most frequently used methods for debugging [5]. Breakpoint has the problem which programmers need experience of programming to decide a location of breakpoint. Nevertheless, it is a useful method of finding the cause of a bug, and also the method which automatically generates breakpoint has reported [6]. When programmers found an abnormal process, it is difficult to understand correctly the situation of execution with only information of a point. TVIS becomes easy to grasp behavior of a program, when programmers found the abnormal process, because TVIS shows them to information of variables or loops which concern it. In a large program, efficiency of TVIS may lower in comparison with breakpoint, because information of the diagram is liable to increase.

An applicable range of programs which TVIS can visualize is small in comparison with the tool [7] which can visualize a large program by a diagram with a high abstraction level or tool [8] which can visualize multithreaded programs. TVIS, which generates a diagram of low abstraction level, may make gigantic a diagram because data to visualize even a program of moderate size may become huge [9]. Therefore, the variety of programs which TVIS can visualize may have a limit. Accordingly, TVIS need reduce spaces to show information by improving the expression format of the diagram. An applicable range of programs which TVIS can visualize is enlarged by focusing object to visualize on only an important object.

6. Conclusion

This paper has proposed the visualizing method of data transitions to support debugging for Java programs in order to improve efficiency of debugging by supporting to find the cause of a bug. We have implemented TVIS which supports debugging and visualizes data transitions, in order to show efficiency of the proposed method. We visualize programs which include a bug by using TVIS, and show that TVIS can support to grasp behavior of executing programs and to find the cause of a bug.

Visualization of the proposed method can give the information which is not gotten by using conventional methods for analyzing data transitions. The data transitions diagram can show visually abnormal behavior: abnormal data renewals and so on. Also, it is easy to understanding states of other variables when abnormal behavior is found.

Therefore, the proposed method can support to find the cause of a bug by visualizing data transitions, and improve efficiency of debugging for Java programs. The method reduces time to grasp behavior in executing a program for programmers by showing data transitions to them. In particular, it is useful for unskilled programmers.

The future issues are as follows.

(1) Improving the format for expressing values of variables.
(2) Introducing the localization in visualization.

References

1. Roger S. Pressman, Software Engineering A Practitioner's Approach 5thEdition, McGraw-Hill Science (2001).
2. M. Sridharan, S. J. Fink, R. Bodik, Thin slicing, In Proc. the 2007 ACM SIGPLAN Conference on PLDI (2007), pp.112-122.
3. Mark Weiser, Programmers Use Slices When Debugging, Communications of the ACM Vol.25 (1982), pp. 446-452.
4. H Agrawal, JR Horgan, Dynamic Program Slicing, SIGPLAN Notices, Vol.25, No.6 (1990), pp.246-256.
5. G. C. Murphy et al, How Are Java Software Developers Using the Eclipse IDE?, IEEE Software, Vol.23, No.4 (2006), pp.76-83.
6. Cheng Zhang et al, Automated Breakpoint Generation for Debugging, JOURNAL OF SOFTWARE, Vol.8, No.3 (2013), pp.603-616.
7. Steven P. Reiss, Guy Eddon, From the Concrete to the Abstract: Visual Representations of Program Execution, DMS 2005 (2005), pp.315-320.
8. Jan Lönnberg et al, Java replay for dependence-based debugging, PADTAD '11 (2011), pp.15-25.
9. W. De Pauw et al, Execution patterns in object-oriented visualization, In Proc. 4th COOTS (1998), pp. 219-234.