Current Trends in Exception Handling

Dewayne E. Perry, *Member*, *IEEE*, Alexander Romanovsky, and Anand Tripathi, *Member*, *IEEE*

1 The Articles

The second part of this special issue on Current Trends in Exception Handling includes four papers which primarily deal with exception handling in human-centered systems such as workflow, requirements specification, and new interactive programming models such as spreadsheets. These research contributions demonstrate that exceptions are not restricted to programming languages, but occur in many, if not most, real-world situations. These papers also reflect that exceptions can be deviations from normal conditions and may not necessarily imply errors. This is similar to Goodenough's observations in his classic paper in the 1970s [1].

These papers lead us to observe that anything that has an algorithmic flow, whether it be workflow or a design process or a program, has a pervasive exception handling need. Programming may be the ultimate in an algorithm, so many of the problems encountered there have analogies in other areas. Moreover, the computation model presented by programming languages tends to be relatively more simple in regard to handling of exceptions in contrast to dealing with such problems in large-scale systems such as enterprise-wide workflow. In those environments, it is not simply exception handling language constructs that are needed, but a methodology on how to use exception handling.

The programming model of spreadsheet systems raises many unique issues related to exception handling. This is a widely used model of programming by end users through the use of many commercial products. In the paper "Exception Handling in the Spreadsheet Paradigm," Margaret Burnett, Anurag Agrawal, and Pieter van Zee discuss these issues and present an approach for handling exceptions in this programming paradigm. Many spreadsheet programs can be quite large and complex and, therefore, both reliability as well maintainability of such programs becomes an issue when exception handling is introduced. The authors present their experience with and analysis of the error value models for spreadsheet programs.

Achieving a high level of fault tolerance is one of the main concerns in developing modern workflow systems due to many factors: Distributed environment, long duration of activities, and complexity of the software involved are among them. The paper "Exception Handling in Workflow Management Systems" by Clause Hagen and Gustavo Alonso describes an advanced fault tolerance mechanism for incorporating both transactions and exception handling into such systems. The approach is unique for workflow systems as it treats workflow support as a programming environment and relies on general research on developing fault-tolerant software. The authors use fundamental research on linguistic issues of exception handling and propose simple ways of applying these concepts to transactional workflow management. The modeling language incorporates special features for error detection and handling which are conceptually similar to exception handling features found in programming languages. Another important way in which this approach is new is how it combines transaction atomicity and exception handling. A validation technique is developed to make it possible to assess the correctness of workflow specification in situations when exceptions are raised and handled.

In the paper "Handling of Irregularities in Human Centered Systems: A Unified Framework for Data Processes," Takahiro Murata and Alex Borgida address exception handling problems in human-centered systems. In such systems, exception handling is required for dealing with errors, as well as deviations, in data as well as processes, from their normal constraints. The paper focuses on exception handling in enterprise workflow systems. Generally, in enterprise systems, process models are used for describing the dynamic nature of activities of humans and semi-automated system entities. Most often, such models do not capture many unanticipated deviations. Sometimes such deviations have to be corrected and other times they are to be tolerated. This paper presents a unified model for handling errors and deviations, which are treated as exception conditions resulting from violations of some specified constraints. When permitting deviations to persist, it relies on runtime checks for assessing their consequences.

Axel van Lamsweerde and Emmanuel Letier address the issues of "Handling Obstacles in Goal-Oriented Requirements Engineering." The requirements elicitation process often results in goals, requirements, and assumptions about the desired system that are too idealized and that do not take into account the various kinds of problems that can occur. Not anticipating exceptional behaviors results in unrealistic, unachievable, or incomplete requirements specifications. This, in turn, leads to systems that are not robust enough and which may fail at critical times, perhaps with critical consequences. The authors present formal techniques for

- *D.E. Perry is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712. E-mail: perry@ece.utexas.edu.*
- *A. Romanovsky is with the Department of Computing Science, University of Newcastle upon Tyne, Newcastle upon Tyne NE1 7RU, UK. E-mail: alexander.romanovsky@newcastle.ac.uk.*
- *A. Tripathi is with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455. E-mail: tripathi@cs.umn.edu.*

For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 112189.

reasoning about obstacles: For generating obstacles from goal formulations and for generating resolutions once the obstacles have been identified. A key principle in this paper is that exceptions should be considered when engineering the requirements while there is still a great deal of freedom to resolve them in satisfactory ways.

References

[1] J.B. Goodenough, "Exception Handling: Issues and a Proposed Notation," *Comm. ACM,* vol. 18, no. 12, pp. 683-693, 1975.

Dewayne E. Perry is currently the Motorola Regents Chair of Software Engineering at the University of Texas at Austin (UT Austin). The first half of his computing career was spent as a professional programmer, with the latter part combining both research (as a visiting faculty member in computer science at Carnegie-Mellon University) and consulting in software architecture and design. The last 16 years were spent doing software engineering research at Bell Laboratories in Murray Hill, New Jersey. His appointment at UT Austin began in January 2000.

His research interests (in the context of software system evolution) are empirical studies, formal models of the software processes, process and product support environments, software architecture, and the practical use of formal specifications and techniques. He is particularly interested in the role architecture plays in the coordination of multisite software development, as well as its role in capitalizing on company software assets in the context of product lines.

His educational interests at UT include building a great software engineering program, both at the graduate and undergraduate levels, creating a software engineering research center, and focusing on the empirical aspects of software engineering to create a mature and rigorous empirical software engineering discipline. He is a co-editor-in-Chief of Wiley's *Software Process: Improvement and Practice*; a former associate editor of the *IEEE Transactions on Software Engineering*; a member of the ACM SIGSOFT and the IEEE Computer Society; and has served as organizing chair, program chair, and program committee member on various software engineering conferences.

Alexander Romanovsky received the MSc degree in applied mathematics from Moscow State University in 1976 and the PhD degree in computer science from St. Petersburg State Technical University in 1988. He was with St. Petersburg State Technical University from 1984 until 1996, doing research and teaching. In 1991, he worked as a visiting researcher at ABB Ltd. Computer Architecture Lab Research Center, Switzerland. In 1993, he was a visiting researcher at Istituto di Elaborazione della Informazione, CNR, Pisa, Italy. In 1993-1994, he was a postdoctoral fellow in the Department of Computing Science, University of Newcastle upon Tyne, United Kingdom. From 1992-1998, he was involved in the Predictably Dependable Computing Systems (PDCS) ESPRIT Basic Research Action and the Design for Validation (DeVa) ESPRIT Basic Project. From 1998-2000, he worked on the Diversity in Safety Critical Software (DISCS) EPSRC/UK Project. He is currently a senior research associate with the Department of Computing Science, University of Newcastle upon Tyne, working on the Dependable Systems of Systems (DSoS) EC IST RTD Project. His research interests include software fault tolerance, software diversity, concurrent programming, concurrent object-oriented and object-based languages, real time systems, exception handling, operating systems, software engineering, and distributed systems. He has coauthored more than 120 scientific papers, book chapters, reports, and a patent in these and related areas.

Anand Tripathi obtained his BTech degree in electrical engineering from the Indian Institute of Technology, Bombay, 1972. He received the MS and PhD degrees in electrical engineering from the University of Texas at Austin in 1978 and 1980, respectively. From 1981 through 1984, he was a senior principal research scientist at the Honeywell Computer Science Center in Minneapolis. There he led research and development projects in distributed operating systems with a focus on fault tolerance and distributed object management. He also was involved in the development of TDC-2000, a distributed process control system developed by Honeywell. He joined the University of Minnesota in 1984. While at the University of Minnesota from 1985 to 1989, he led the design and development of a middleware system called Nexus for distributed object-based computing. From 1995 to 1997, he also served as the program director for the Computer Systems Software program at the US National Science Foundation. Currently, he is an associate professor in the Departmentof Computer Science and Engineering at the University of Minnesota. He has published more than 60 research papers in various journals and conferences in the areas of distributed systems, object-oriented systems and languages, and fault-tolerant computing. His current research focus is on system level mechanisms for building robust and secure distributed applications using mobile agents. He is a member of the IEEE, the editorial board of *IEEE Concurrency*, and the editor for the education column of *IEEE Concurrency*. He is also currently serving as an IEEE distinguished visitor.