Wild and Crazy Idea Session

14th International Conference on
Architectural Support for Programming Languages and Operating Systems

Abstract:

Software, computer interfaces, and operating systems potentially fail. Currently various methods are used to handle failures. Software may return a numeric or short string error code; raise an exception; write to a log; or generate a signal that may potentially be caught. Some standards allow error records generated from multiple threads to be stored in a queue, so that each such error record can be retrieved, and handled as deemed appropriate. However an error is reported, the information typically only indicates that a problem exists. Such reports rarely identify what has failed; why failure has occurred; and how observed problems in software can be corrected.

A radical change in thinking is needed in both programming languages and operating systems, so as to ensure that:

1. Problems encountered by software are recorded for subsequent resolution.
2. Errors are uniquely tied to the specific location in the code that generated them.
3. Cascading errors can be handled without inducing needless program termination.
4. Documentation regarding the nature of reported errors can be augmented and shared over time, by arbitrary users seeking to troubleshoot errors, as well as by the software provider, without necessitating recompilation of any code.

It is suggested that the operating environment be extended to support a single distributed system log to which all software must report errors. Each log entry would include (in addition to the conventional information stored in such logs, such as references within the log to earlier contributing causes of concern) a unique software, version and problem specific URL where further information on the observed problem could be found. This URL would serve as the meeting room where software developers could explain the problem observed, with instructions regarding how to correct it, and where end users could post their own discoveries and workarounds regarding unsolved problems in software.

Such a mechanism could incrementally be fitted into legacy software and could be readily used in new software. If universally implemented it would greatly simplify the current challenges involved in root-cause analysis; encourage cooperation between independent parties seeking to diagnose and correct the same observed problem; and permit potential clarification of poorly documented errors such as “Unable to open file”. From a software developer’s perspective, this approach would be of enormous value in dynamically documenting (as web page hits) where released software was actually failing in the field and through parameterized URLs why.

In summary, URLs need to be better exploited in providing Architectural Support for Programming Languages and Operating Systems.

Ian Davis
University of Waterloo, 200 University Avenue West, Waterloo, Ontario, N2L 3G1, Canada

17 February 2009