## 6.CGT Concurrency – Directed Termination [CGT]

### 6.CGT.0 Terminology

### 6.CGT.1 Description of Application Vulnerability

This discussion is associated with the effects of unsuccessful or late termination of a thread. For a discussion of premature termination, see CGS Concurrency – Premature Termination.

When a thread is working cooperatively with other threads and is directed to terminate, there are a number of error situations that may occur that can lead to compromise of the system. The termination directing thread may request that one or more other threads abort or terminate, but the terminated thread(s) may not be in a state such that the termination can occur, may ignore the direction, or may take longer to abort or terminate then the application can tolerate. In any case, on most systems, the thread will not terminate until it is next scheduled for execution.

Unexpectedly delayed termination or the consumption of resources by the termination itself may cause a failure to meet deadlines, which, in turn, may lead to other failures.

### 6.CGT.2 Cross References

Hoare C.A.R., "Communicating Sequential Processes", Prentice Hall, 1985

Holzmann G., "The SPIN Model Checker: Principles and Reference Manual"., Addison Wesley Professional. 2003

Larsen, Peterson, Wang, "Model Checking for Real-Time Systems"., Proceedings of the 10th International Conference on Fundamentals of Computation Theory, 1995

The Ravenscar Tasking Profile, specified in ISO/IEC 8652:1995 Ada with TC 1:2001 and AM 1:2007

CWE 364 Signal Handler Race Condition

### 6.CGT.3 Mechanism of Failure

The abort of a thread may not happen if a thread is in an abort-deferred region and does not leave that region (for whatever reason) after the abort directive is given. Similarly, if abort is implemented as an event sent to a thread and it is permitted to ignore such events, then the abort will not be obeyed.

The termination of a thread may not happen if the thread ignores the directive to terminate, or if the finalization of the thread to be terminated does not complete.

If the termination directing thread continues on the false assumption that termination has completed, then any sort of failure may occur.

## 6.CGT.4 Applicable Language Characteristics

Languages that permit concurrency within the language, or support libraries and operating systems (such as POSIX-compliant operating systems or Windows) that provide hooks for concurrency control.

## 6.CGT.5 Avoiding the Vulnerability or Mitigating its Effects

Software developers can avoid the vulnerability or mitigate its ill effects in the following ways:

* Use mechanisms of the language or system to determine that aborted threads or threads directed to terminate have successfully terminated. Such mechanisms may include direct communication, runtime-level checks, explicit dependency relationships, or progress counters in shared communication code to verify progress.
* Provide mechanisms to detect and/or recover from failed termination.

### 6.CGT.6 Implications for Standardization

In future standardization activities, the following items should be considered:

* Provide a mechanism (either a language mechanism or a service call) to signal either another thread or an entity that can be queried by other threads when a thread terminates.