
Abstract
ET: Enhanced tool (solution strategy)
The effectiveness of <code validation> in supporting the design of <software> has been demonstrated. An enhanced tool / method is described for the design of <safe software> based on <representing faulty states as a tree of sub-states that contribute to the fault appearing in parent nodes of the tree>. Examples are provided confirming the effectiveness of its support for <software development> in design.

Why ET?
The authors state that their “technique is an offshoot of an engineering technique used for the safety analysis of electromechanical devices,” which they have applied to software rather than hardware faults.

Question - [Method/means of development]
<model-type / solution strategy>
What is the most effective way of evaluating the safety of software? It appears that previous approaches did not place a premium on evaluation of what we today call “safety,” often because they glossed over environmental conditions and perhaps because they weren’t formal enough.

<artifact-type>
The artifact being produced is a software implementation. The question is the best way to ensure that the implementation never ends up in an unsafe state.

Results - [Procedure / technique]
<model-type / solution strategy>
The authors propose to use a tree-like structure to support the analysis of faulty states within software. Each node represents a state, with the root node being an unsafe, faulty state. Child nodes, combined through OR / AND logic, depict how sub-states together conspire to generate each faulty state. The authors highlight the fact that environmental conditions can also be represented in nodes of such trees.

Validation – [Experience]
<model-type / solution strategy>
The authors applied their analysis method to build a model of the FIREWHEEL software system, used to control a real satellite. In doing so, they successfully uncovered a handful of unsafe states that had not been caught by previous validations.