Exceptional Handling in the Software Lifecycle

Rogério de Lemos, Computing Laboratory – University of Kent at Canterbury, UK

Motivation

A possible cause for the accident of Ariane 5 in its maiden flight has been associated with the mishandling of an abnormal situation (known as an exception) that had occurred in the embedded software of the launcher. That is, after an exception behaviour was detected, the inappropriate corrective action taken by the software made the vehicle to veer off its flight path leading to the structural destruction of the launcher.

• Exception handling has been associated with design:
  • protect the application software from faults: application, design, implementation and support;
  • consequences:
    • context of error detection and recovery is lost;
    • correlation between exceptions and handlers is lost.

• Proposed approach:
  • specify exceptions, and their respective handlers, in the context where faults occur.

Requirements Phase

• The normal behaviour of a use case is defined by:
  • primary scenarios:
    • capture the basic flow, the minimum number of actions for the specified functionality to be implemented;
  • alternative scenarios:
    • capture the actions that extend the basic flow, mainly to add functionality.

• The abnormal behaviour of a use case is defined by:
  • exceptional scenarios:
    • capture the deviation from normal behaviour when errors are manifested;
    • consist of error detection and recovery;
  • failure scenarios:
    • capture the unacceptable and irreversible system behaviour.

• The three stage approach for modelling and analyse the interdependencies between actors and use cases:
  • describe use cases in the usual manner, but including abnormal behaviour;
  • refine the previous use case description, including:
    • identification of system variables;
    • table of exceptions:
      • causal relation between the abnormal behaviours of actors and use cases;
      • collaboration diagrams for the exception behaviour;
      • formalise use case description for validation;
    • eg. model checking.

Automaton of the primary and exceptional scenarios of DrainSump

Architectural Design

• The exceptional behaviour associated with the actors and use cases should be mapped into the architectural elements of a software architecture.
  • components alone do not provide the appropriate encapsulation to deal with the collaborative behaviour of exception handling:
    • hence the need for a co-operative connector.

For further information
Contact Rogério de Lemos (r.delemos@ukc.ac.uk) or visit website http://www.cs.ukc.ac.uk/people/staff/rdl/