Idealised Fault Tolerant Architectural Element

Rogério de Lemos
University of Kent, UK

- Motivation - architectural fault tolerance;
- iFTE & propagation of exceptions;
- Case study - mining control system;
- Conclusions & future work;
Motivation

Architectures are about structures:

◆ unstructured approaches can reduce system dependability by introducing more faults;
◆ a good architecture should promote error confinement;

Architectural fault tolerance:

◆ avoid the failure of systems
  ◆ error detection and handling;
  ◆ fault handling;
◆ components need to collaborate for handling certain failure scenarios;
An architectural solution based on exception handling:

- **idealised fault tolerant component** enables fault tolerance to be built into the system [Anderson & Lee 81]:
  - separation between normal and abnormal behaviour;
  - provided and required services;
  - local, interface and failure exceptions;
Exception handlers provides mechanisms for:

- handling exceptional conditions so that the exception can be masked;
  - backward recovery - roll back to a previous state;
  - forward recovery - perform actions to correct the state by other means;
- signalling exceptions;

Handlers are provided for anticipated exceptions:

- default handlers are provided for unanticipated exceptions;
Idealised Fault Tolerant C2 Component (iC2C)

- **NormalActivity**: iC2C_top
  - upper_detector
  - COTS
  - lower_detector

- **AbnormalActivity**: iC2C_internal
  - AbnormalActivity
  - AbnormalActivity

- **Error Detector (1)**: upper_detector
  - Error Detector (n)

- **Error Diagnosis**: abnormal_top

- **Error Handler (1)**: abnormal_internal
  - Error Handler (n)

- **Error Handler (n)**: abnormal_bottom
**Idealised fault tolerant architectural element (iFTE):**

- **fault-tolerant software component:**
  - preventing the propagation of internal errors by constraining its exceptional behaviour;

- **fault-tolerant software connector:**
  - coordinating exceptional behaviour among components;
  - resolving potential mismatches;
  - preventing the propagation of errors by handling them as exceptions;
Architectural solution/pattern:

- peer-to-peer style;
- request/reply interaction;
Normal behaviour:

- internal services with no exceptions;
- internal services with exceptions:
  - masked by internal handlers;
  - masked by external handlers;
- requests external services with exceptions:
  - masked by internal handlers;
  - masked by external handlers;
- requests external services with no exceptions;

Exceptional behaviour:

- internal services with exceptions:
  - not masked by internal handlers;
  - not masked by external handlers;
- requests external services with exceptions:
  - not masked by internal handlers;
  - not masked by external handlers;
normal behaviour when requesting external services with no exceptions;
 contexts for handling exceptions:
  ◆ component, roles and connectors;
  ◆ exceptions meaningful for components and connectors;
  ◆ translation on the types of exceptions;

Propagation of exceptions:
  ◆ from components to connectors;
  ◆ from connectors to components;
Propagation of exceptions:

- from connectors to connectors;
**Embedded System: Mining Control System**

Mining environment

1- Control system
2- Pump
3- Exhaustor
4- Water sensor (low level)
5- Water sensor (high level)
6- Methan sensor
Embedded System: Mining Control System
Exception propagation when AirExtractor fails exception is propagated to OperatorInterface:

- the whole system shuts down;
Conclusions

Fault tolerance at the architectural level:

◆ error detection and handling:
  ◆ application dependent;
  ◆ idealised Fault Tolerant Architectural Elements (iFTE);
    ◆ architectural solution/pattern based on exception handling;

◆ fault handling:
  ◆ not application dependent;
  ◆ reconfiguration support by CA action;
Future Work

- model the iFTE with AADL - Error Model;
- iFTE is application dependent and requires additional assurances:
  - model iFTE with B and CSP for analysing the propagation of exceptions;
  - identification of iFTE properties that can be applied to architectures;
  - identification of iFTE test cases;
  - automatic generation of Provided and Required components;