Fault-Tolerant Systems

Dominik Tornow
Outline

- Dependability
- Fault Tolerance
- Examples
Dependability
Basic Concepts

- Services, Systems and Components
Basic Concepts

- Services, Systems and Components
Dependability

- Dependability

  *Dependability* is defined as the trustworthiness of a system such that reliance can be placed on the service it delivers.
Attributes

The attributes of dependability allow to express the expected properties of a system.
Dependability – Attributes

- Availability
  
  *Availability* denotes the readiness for usage

- Reliability
  
  *Reliability* denotes the continuity of service

- Safety
  
  *Safety* denotes the avoidance of catastrophic consequences

- Security
  
  *Security* denotes the prevention of unauthorized access to services
Impairments

The *impairments* of dependability cause un-dependability, so reliance cannot be placed on the service.
Dependability – Impairments

- **Faults**
  Divergence between the real system and the specified system

- **Errors**
  Divergence between the real system state and the specified system state

- **Failures**
  Divergence between the real system behavior and the specified system behavior
 Dependability – Impairments

☐ Failure Semantics
Dependability – Means

- Means

The *means* of dependability are methods and/or techniques enabling the ability to deliver reliable services.
Dependability – Means

- **Fault Removal**
  
  *Fault Removal* aims to reduce the number and/or seriousness of faults

- **Fault Prevention**
  
  *Fault Prevention* aims to prevent fault introduction or occurrence

- **Fault Forecasting**
  
  *Fault Forecasting* aims to estimate the number and/or seriousness of faults

- **Fault Tolerance**
  
  *Fault Tolerance* aims to provide a service complying with the specification in spite of faults
Fault Tolerance
Fault Tolerance

- Fault Tolerance
  - Fault Diagnosis
  - Fault Passivation
  - Error Recovery
  - Error Compensation
  - Error Processing
  - Fault Treatment
Fault Tolerance – Fault Treatment

- Fault Treatment
  
  *Fault Treatment* prevents faults from being activated

- Fault Diagnosis
  
  *Fault Diagnosis* determines the cause(s) of error(s) in terms of location and nature

- Fault Passivation
  
  *Fault Passivation* makes a fault passive i.e. prevents faults from being activated
Fault Tolerance – Error Processing

☐ Error Processing
Fault Tolerance – Error Processing – Error Recovery

- Error Recovery
  
  *Error Recovery* substitutes an erroneous state by an error-free state

- Forward Recovery
  
  *Forward Recovery* transforms an erroneous state into a new error-free state

- Backward Recovery
  
  *Backward Recovery* transforms an erroneous state into a state already occupied prior to error occurrence
Error Compensation

*Error Compensation* delivers an error-free service in spite of an erroneous state due to redundant system components.

- **Hierarchical Masking**
  *Hierarchical Masking* provides redundant system components whereas the service consumer is responsible to handle an error.

- **Group Masking**
  *Group Masking* provides redundant system components whereas a group manager is responsible to handle an error.
Fault Tolerance – Error Compensation – Masking

- Hierarchical Masking

```
Service Consumer

Service Provider
State

... 

Service Provider
State
```
Group Masking

- Service Consumer
- Server Group Manager
- Service Provider
- State
Examples
VAX
Ariane 5

Inertial Reference System II
vertical velocity
horizontal velocity

Inertial Reference System I
vertical velocity
horizontal velocity
declare

v_veloc_sensor: float; -- 64 bit Gleitkomma-Darstellung
h_veloc_sensor: float;

v_veloc_bias: integer; -- 16 bit Signed Integer
h_veloc_bias: integer;

begin

sensor_get (v_veloc_sensor);
sensor_get (h_veloc_sensor);

v_veloc_bias := integer (v_veloc_sensor);
h_veloc_bias := integer (h_veloc_sensor);

exception

when others => use_irs1();

end;
Literature

- *Dependability, Basic Concepts and Terminologie*, J.C. Laprie
- *Understanding Fault-Tolerant Distributed Systems*, F. Christian
- *Komponentenbasierte Softwareentwicklung*, Andresen, 2003
Thank You

Questions