Tool-Supported Dependability Evaluation of Redundant Architectures in Computer-Based Control Systems

István Majzik, Péter Domokos, Melinda Magyar
Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

FORMS / FORMAT 2007
Motivation

- **Architectural choices** have profound influence on system dependability (reliability, availability)
  - Degree and type of redundancy (fault tolerance)
- **Standards** require a thorough evaluation of possible failures and protection mechanisms
  - Quantitative evaluation: Computation of system level measures using component-level reliability parameters
- **Model-based dependability evaluation**
  - (Formal) dependability model is constructed
    - Component failure and repair (recovery) behaviour is modelled
  - Allows „what-if” kind of analysis in early design phases
  - Optimization of architectural choices (decisions)
Dependability modelling approach*

• Formalisms for dependability models
  – Combinatorial models (e.g. fault trees)
  – Stochastic state space models (CTMC, GSPN) allow to capture dependencies between components

• Design models shall be supported
  – Construction of dependability models automatically
  – Assembling the state space of the model taking into account failure states and repair processes
  – Integrating the expert knowledge in a tool

• UML: formalism of the design model, GSPN: formalism of the dependability model

Dependability model construction

• Dependability model represents
  – Fault occurrences in components
  – Error propagations between components
  – Repair (maintenance) mechanisms

• Component types are assigned GSPN subnets that represent these processes
  – Hardware, software, stateful, stateless components are distinguished
  – Component types and related local dependability parameters can be identified in the design model (UML stereotypes and tagged values)
Dependability model construction

- Dependability model represents:
  - Fault occurrences in components
  - Error propagations between components
  - Repair (maintenance) mechanisms

- Component types are assigned GSPN subnets that represent these processes:
  - Hardware, software, stateful, stateless components are distinguished
  - Component types and related local dependability parameters can be identified in the design model (UML stereotypes and tagged values)

Failure subnet of a stateful hardware

Generic error propagation subnet between components

[Diagram of GSPN subnets and error propagation]

Extensions of the UML design model

- Identify component types and local parameters (profile)

**Diagram: Class Diagram**

- **Composite Component**
  - **StatefulHardware**
    - FaultOccurrenceRate
    - ErrorLatencyTime
    - PermanentFaultRatio
    - RepairDelay
  - **StatelessHardware**
    - FaultOccurrenceRate
    - PermanentFaultRatio
    - RepairDelay
  - **StatefulSoftware**
    - FaultOccurrenceRate
    - ErrorLatencyTime
    - RepairDelay
  - **StatelessSoftware**
    - FaultOccurrenceRate

**Stereotype: Stateful Software**
Tagged values:
- Fault occurrence rate
- Error latency time
- Repair delay
Modelling redundancy

• Identification of roles
  – Redundancy manager
  – Variant
  – Adjudicator

• „Logic” of redundancy (error propagation)
  – Fault tree
  – Specific GSPN subnet
Tool support for dependability modelling

• **Integration** of subnets assigned to components
  – Subnets are constructed by dependability experts

• **Re-use** of dependability subnets from a library
  – Assigned to common redundancy management (TMR, NMR, NVP, etc.)
  – Assigned to architectural design patterns (optionally handled as aspect models)

• **Refinement** of dependability subnets
  – Early phases of design: Generic subnets
  – Design refinements: Refined subnets
    They can be transformed from behavioural models:
    E.g. statechart of the redundancy manager → Fault tree
Tool architecture

**System model**
UML architecture diagrams:
- classes and objects
- associations / relations, deployment (sw on hw)
- packages (subsystems)

Extensions:
- Component type from dependability viewpoint (stateless, hardware etc.)
- Numerical attributes (local failure rate, repair rate, propagation probability etc.)
Tool architecture

System model
XML Metadata Interchange

Identifying component types and relations

UML model

UML → IM

IM model

IM → GSPN

GSPN model

UML → IM

IM → GSPN

IM → GSPN
Tool architecture

Dependability model

Elements:
- components: local failure / repair characteristics
- subsystems: measures

Relations:
- component uses component: error / repair propagation
- subsystem is composed of (redundant) components: (non-trivial) error propagation
- system is composed of subsystems: error propagation
Tool architecture

- UML model
- UML → IM
- IM model
- IM → GSPN
- Assembling subnets
- GSPN model

Dependability model
Tool architecture

**Analysis model**

System-level GSPN: composed of modular subnets

- **Component subnets:**
  - “local” failure/repair subnets
  - Relations subnets:
    - simple error propagation (uses relation)
    - non-trivial error propagation (composed-of relation, FT)
    - repair propagation

**Diagram**

1. UML model
2. UML → IM
3. IM model
4. IM → GSPN
5. GSPN model

IM → GSPN

Tool architecture

UML model

UML → IM

IM model

IM → GSPN

GSPN model

Analysis model

```
int gtT5_930() {
    return (mark("Pl_475_H") == 1);
}

net() {
    place("Pl_873_H");
    init("Pl_873_H", 1);
    rateval("T0_930", 1000.0);
    guard("T5_930", gtT5_930);
    oarc("ExpRep_610", "Pl_610_H");
    harc("Impl_610", "P4_610");
    iarc("T5_649", "A_bad_649");
}
```

Model solution (SPNP)

System-level measures
Tool architecture

- UML model
- IM model
- GSPN model
- IM editor
- UML → IM
- IM → GSPN

FTS library
- Redundancy structures
- Design patterns (re-use)
- Manual integration
- Reference to analysis submodels (GSPN subnet library)

Patterns are mapped to corresponding GSPN subnets

UML model library (design patterns)
GSPN subnet library (analysis subnets)
Tool architecture

Constructing analysis subnets:
- From fault trees
- Directly in GSPN (PNML) editor

• UML model → IM
• IM → GSPN
• UML model library (design patterns)
• Fault tree editor
• Fault tree → GSPN
• PNML editor
• GSPN subnet library (analysis subnets)
Tool architecture

- **IM editor**
  - UML model → IM
  - IM model → GSPN

- **UML model library** (design patterns)
  - UML model → IM

- **GSPN subnet library** (analysis subnets)
  - IM model → GSPN

- **PNML editor**
  - Fault tree editor
  - Fault tree → GSPN

Library can be extended:
- Identification of component types
- Support of new design patterns
- Behavioral information available
Aspect-oriented modelling approach

Core architecture without redundancy

- UML model
- UML → IM
- IM model
- UML → GSPN
- GSPN model
- GSPN subnet library (analysis subnets)

UML model library (design patterns)

- Weaving layer

Model weaver

- Fault tree editor
- Fault tree → GSPN
- PNML editor

Dependability related design decisions are separated

Core architecture without redundancy
Aspect-oriented modelling approach

Where and how to apply redundancy: Instantiating components from design patterns and specifying links to the core UML model
Aspect-oriented modelling approach

- Core architecture without redundancy
- System level integrated models
- UML model
  - UML → IM
  - IM model
    - UML → GSPN
      - GSPN model
- UML model library (design patterns)
  - Weaving layer
    - GSPN subnet library (analysis subnets)
      - PNML editor
        - Fault tree editor
          - Fault tree → GSPN
            - Dependability related design decisions are separated
Summary

• Method to construct GSPN dependability models
  – External GSPN solver → System level availability

• Adaptability to different input models
  – UML, AADL (in progress)
  – IM is the core mathematical formalism

• Extensibility: Subnet library for components
  – Specialisation, design refinement

• Aspect-oriented modelling of redundancy
  – Separation of design decisions related to fault tolerance
  – Weaving of design and analysis models