Towards a Metamodel for Dependability Cases

George Despotou
Dimitrios Kolovos
Richard Paige
Fiona Polack
<u>Tim Kelly</u>

E-mail: tim.kelly@cs.york.ac.uk

Department of Computer Science
University of York, UK

Overview

- Dependability
- Dependability cases
  - Assurance (safety cases)
  - Broader concerns (dependability assurance)
- The role of argumentation
  - Goal Structuring Notation (GSN)
- Challenges
  - Proposed methodologies
- Definition of framework
  - Technical approach
  - Design approach
- Advantages

Dependability

- Range of definitions
  - Concerned with undesirable consequences of behaviour
  - Consists of a number of attributes
    - Safety, security, maintainability, availability etc.
- For many, attributes resemble ‘non-functional’ requirements
  - Cannot effectively separate functionality
  - Satisfaction of a dependability attribute can result in addition of further functionality which affects other attributes
    - E.g. fault recovery for safety
- Attributes of Dependability
  - Heterogeneous
  - Ranked differently and subjectively by the stakeholders
  - Interrelated
    - In conflict or in harmony

The Purpose of a Safety Case

Principal Objective:

- **safety case presents the argument that a system will be acceptably safe in a given context**

- **‘system’ could be ...**
  - **physical (e.g. aero-engines, reactor protection systems)**
  - **procedural (e.g. railway operations, off-shore)**

In practice:

- **often series of safety cases produced — stages of development and/or operation**
- **safety cases are large, complex, technical and *political* documents**

Some Safety Case Definitions

- *"A safety case is a comprehensive and structured set of safety documentation which is aimed to ensure that the safety of a specific vessel or equipment can be demonstrated by reference to:*
  - *safety arrangements and organisation*
  - *safety analyses*
  - *compliance with the standards and best practice*
  - *acceptance tests*
  - *audits*
  - *inspections*
  - *feedback*
  - *provision made for safe use including emergency arrangements"*

(JSP 430)

- *"A Safety Case is a structured argument, supported by a body of evidence, that provides a compelling, comprehensible and valid case that a system is safe for a given application in a given environment."*

(DS 00-56 Issue 3)

Arguments about Dependability Attributes

- Several standards require assurance that the acceptability levels of an attribute have been met
- Examples:
  - MoD Defence Standard 00-56 requires a <u>Safety</u> case
  - MoD Defence Standard 00-40 requires a <u>Reliability and Maintainability</u> (R&M) case
  - Common criteria of information technology security requires a description of how a security level is met
    - A few examples of security 'cases' now exist
- Above examples tackle only a single attribute
  - Each attribute requires specific domain knowledge, expertise, methods

Dependability Cases

- The purpose of the Dependability Case is to communicate an argument that a system is acceptably dependable is a given context
  - Based on concepts established for Safety Cases
- A convincing ‘case’ requires two elements:
  - *Supporting Evidence*
    - *Argument without evidence is unfounded*
  - *High Level Argument*
    - *Evidence without argument is unexplained*
- Writing a case in a purely textual form is ineffective
  - Logical inferences
  - Clarity and ease of reading
- Goal Structuring Notation introduced (10+ yrs ago) as a means to represent (safety) arguments

The Goal Structuring Notation

Purpose of a Goal Structure

To show how **goals** are broken down into sub-goals,

and eventually supported by evidence (**solutions**)

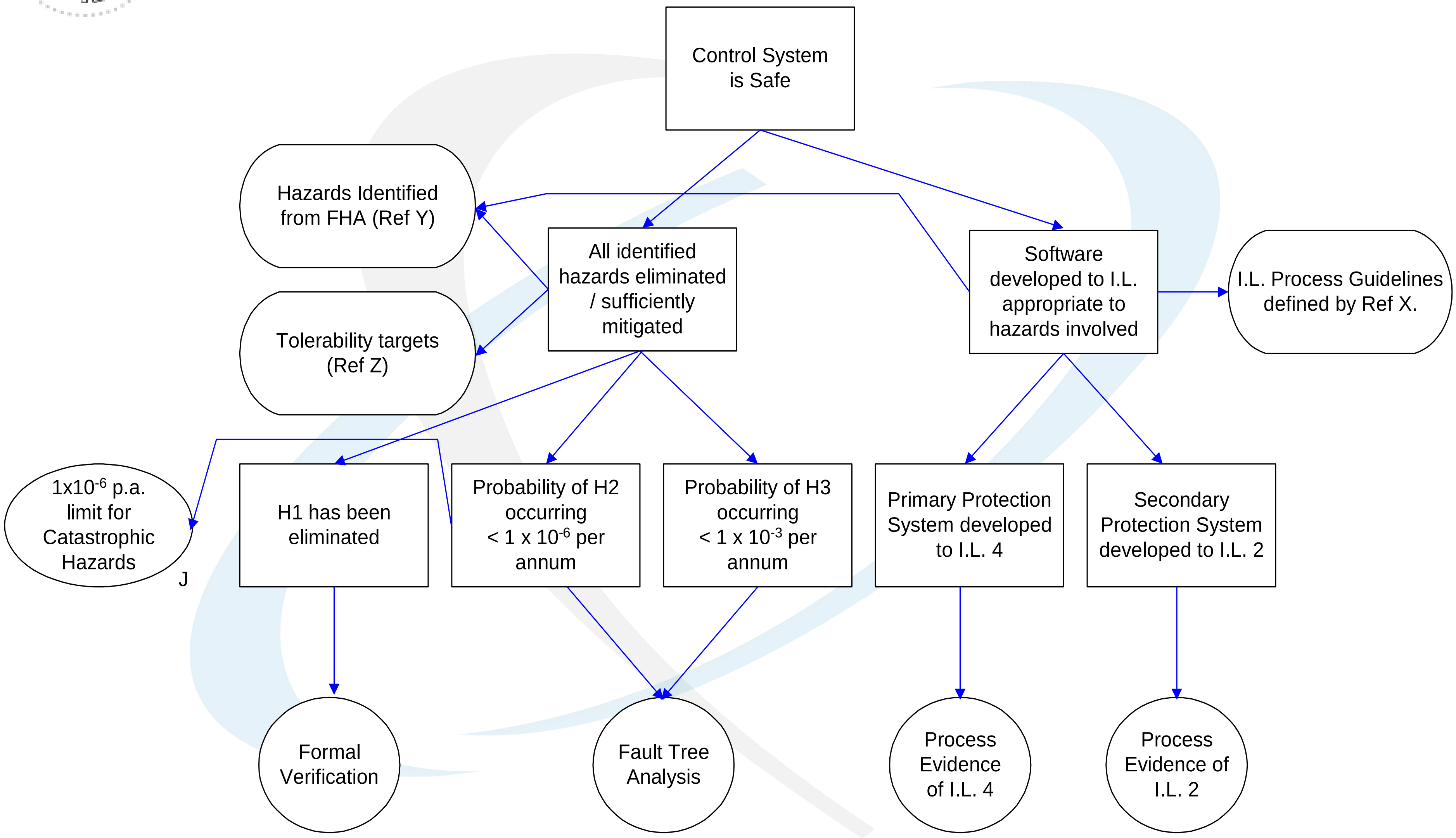
whilst making clear the **strategies** adopted,

A/J

the rationale for the approach (**assumptions, justifications**)

and the **context** in which goals are stated

A Simple Goal Structure



Argumentation vs. Requirements

- GSN is an <u>argumentation</u> technique
  - Goals are TRUE/FALSE propositional statements
    - Assertions that you are prepared to make to a evaluator (e.g. regulatory authority)
    - E.g. “This task is performed within 5s”
  - … rather than should / must ‘demands’
    - E.g. “This task should be performed within 5s”
- Typical use of GSN for evolving arguments:
  - Arguments built top-down during system / project evolution
    - What *will* we have to claim is true?
  - Arguments then ‘checked’ bottom-up as evidence becomes available
- In this way, evolving GSN argument serves to define objectives during system evolution

Problems and Challenges (1)

- Heterogeneity of attributes
  - Different methods, expertise, implementation strategies and consequently arguments
    - <u>Safety</u>: Hazard mitigation
    - <u>Reliability</u>: Redundancy, component resilience
    - <u>Security</u>: Protection from threats
- Construction of a case
  - Safety cases evolve in parallel with the system
  - A case about dependability should evolve in parallel with the system
    - Interaction of argument and design (teams)
      - ⇨ Efficiency of design
      - ⇨ Realism of requirements
  - All attributes should be acceptable in context of each other

Problems and Challenges (2)

- Attributes can be interrelated
  - Attributes can be in conflict or in harmony
  - Non-orthogonal
    - Previous research showed that cannot we effectively represent dependability in a single metric
    - Qualitative considerations necessary
  - Various types and magnitude of association depending on design
  - Results in trade-offs
- Trading attributes
  - Selection of the least worst design w.r.t. attributes
  - Justification of trade-off
    - Subjectivity – attribute importance
    - Rationale – impact of trade-off

Facilitation of Trade-offs

- **Trade-off Method**
  - **The method facilitates the resolution of conflicts**
    - **Processes information about attributes & architectural options**
    - **Provides the grounds for arguments and elicits rationale for design/architectural decisions.**
- **Ultimately creates an argument of preference among candidate decisions**
  - **Identification of prevailing decision**
    - **Acceptable by all stakeholders**
  - **Qualitative reasoning**
  - **Admissible decisions**
  - **Adoption of flexible requirements**

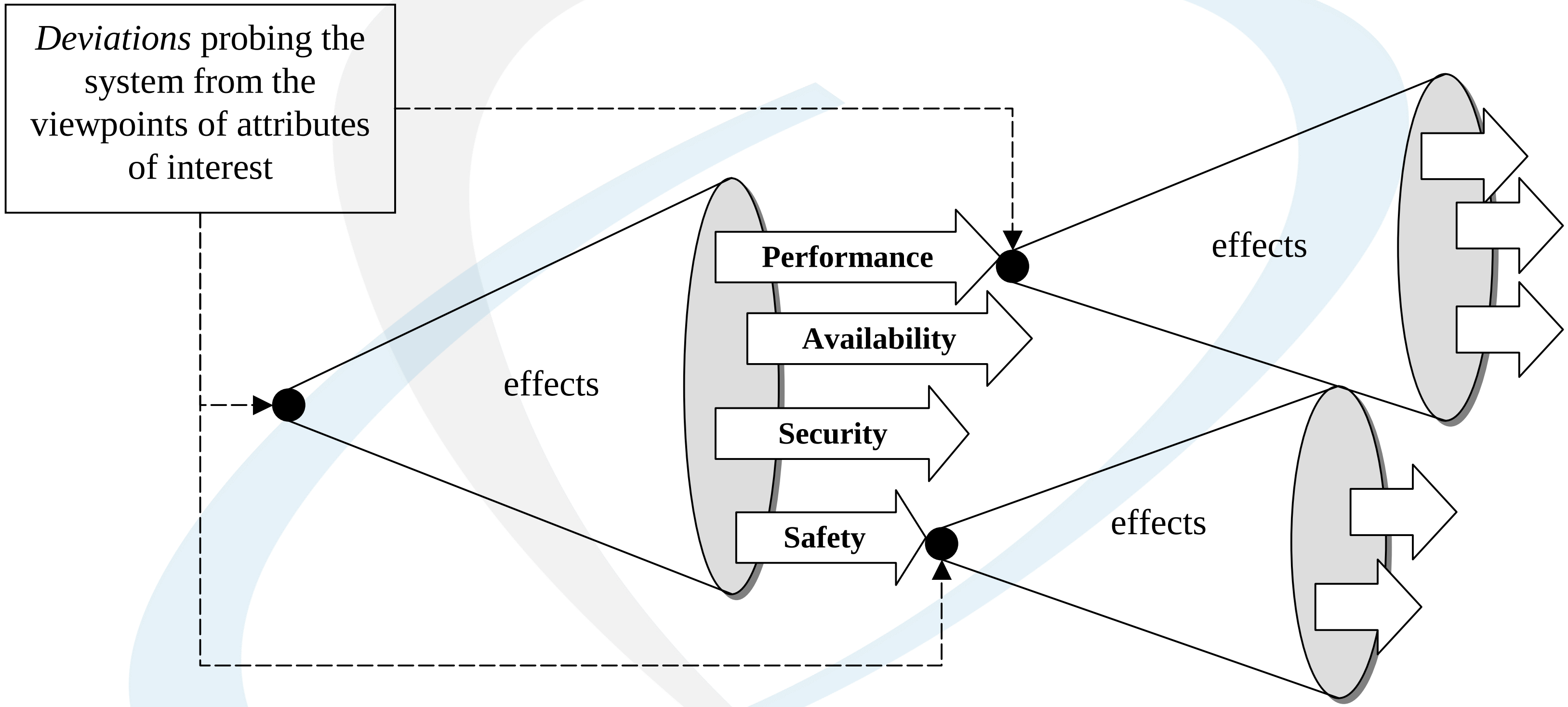
Flexible Requirements

- Avoid commitment to premature requirements
  - Over-specified systems
  - Wanting to define a solution ‘space’
  - Set of {Goal, Target, Limit, T&L Just’n, Achievement Claim, Optimality Claim} important to provide compelling case
    - Helps customer to appreciate acceptability of total case, and appreciate viewpoints
- Two aspects to goal satisfaction
  - (Level of) Achievement of Goal
  - (Level of) Assurance of Achievement of Goal
- Facilitation of trade-offs
  - Inevitable (especially in complex systems)
  - Admissible requirements with respect to operation
    - Corresponding trade-offs at different levels of the design
    - Justification of design decisions

Dependability Requirements Analysis

- Dependability deviation analysis (DDA)
  - Inspired from principles of (safety) deviation based analyses
- Impact of typical (dependability attribute) issues on system operation
  - Identification of primary concerns to envisioned CONOPS
- Identification of failure conditions
  - Definition of 'optimised' suitable deviations
    - Guideword + system element type
    - Cover the spectrum of identified attributes
  - Examination of interrelationships between failure conditions
- Induction of required system element behaviour
  - w.r.t to dependability attributes (prompted by the deviations)
  - Definition of a 'dependability profile'

DDA (Overview)

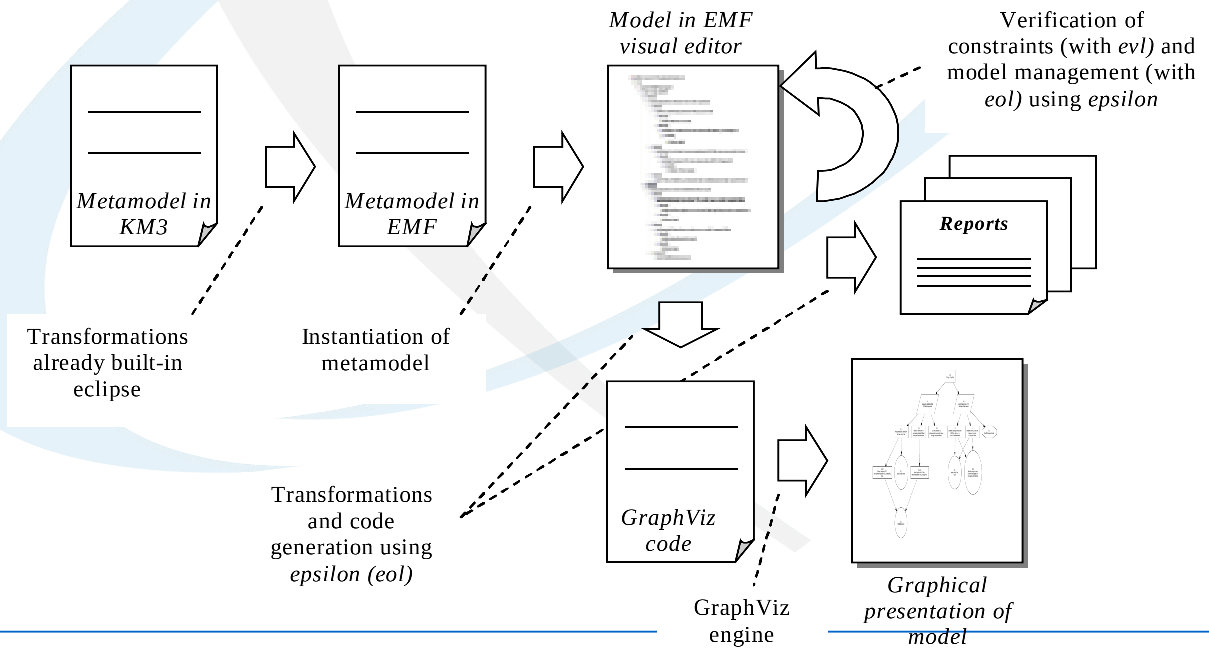


Factor Analysis and Decision Alternatives

- Capturing design rationale and brainstorming
- Part of argument – system evolution
- Main concept: factors
  - (Design) decisions collections of factors
  - -ve & +ve contributions to achieving a goal
  - Identification of ‘sensitivity points’
- Incremental collection of evidence about impact of decisions on goals
  - Input for trade-off method

Rigorous Definition of Framework

- Domain Specific Language
- Technical approach
  - Definition of metamodel in KM3
  - Transformation to ECore metamodel
  - Use of the Eclipse EMF editor for instantiation of metamodel
  - Model management using EPSILON
    - product of research at York sponsored by the EU ModelWare, ModelPlex projects
  - Transformation to Graphviz (graph visualization tool from AT&T)



Kernel MetaMetaModel (KM3)

- **Used mainly for its ease of use and clarity of the metamodel**
- **Example…**

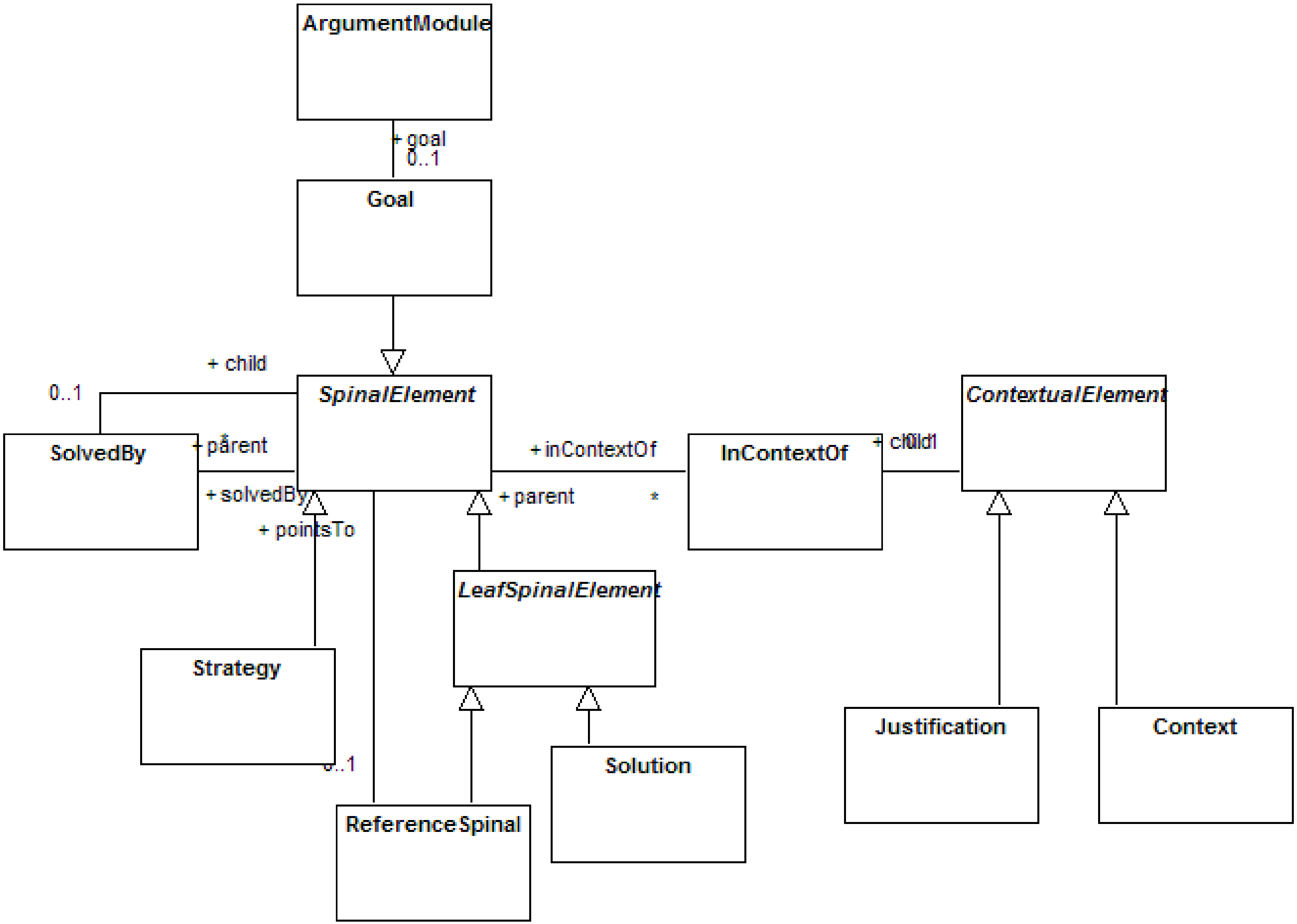
```
package GSN  {

    abstract class SpinalElement extends ModelElement {
        reference solvedBy [*] container : SolvedBy oppositeOf parent;
        reference inContextOf [*] container : InContextOf oppositeOf parent;
    }

    class SolvedBy {
        reference parent : SpinalElement oppositeOf solvedBy;
        reference child container : SpinalElement;
        attribute cardinality : String;
        attribute optional : Boolean;
    }

    datatype String;
    datatype Boolean;
}
```

UML Example of (Basic) GSN metamodel



Model Management using EPSILON

- Epsilon is a platform of integrated languages for Model Management *
  - Provides tailored languages for: transformation, validation, comparison, merging, code generation
  - Can manage models of diverse metamodels and modelling technologies
- For this case, the Epsilon Object Language (EOL) and the Epsilon Validation Language (EVL) were used.
- Exemplar validation constraint (with an error message) expressed in EVL:

```
context Goal {

    constraint HasUniqueDescription :
        Goal.allInstances.forAll
            (g|g.description = self.description implies g = self)

        fail : 'Goal ' + self.description + ' has not unique ID'
    }
}
```

* www.eclipse.org/gmt/epsilon

Advantages in Defining the Metamodel

- Clear view of the domain
  - Metamodel is explicit and separated from the tool(s) that support it
  - Can be the starting point for a discussion on a generally-accepted dependability metamodel
- Interoperability
  - Metamodel implemented using ECore, an implementation of the OMG MOF 2.0 standard
  - MOF models are serialized in a uniform XML-based format (XMI), and can be exchanged between tools from different vendors
- Model Management
  - Dependability models can be managed (transformed, validated, analyzed etc.) with various Model Engineering tools such as QVT, Epsilon, AMMA, MOFScript

Summary

- Concept of Safety Cases well established
- Dependability Cases extend this concept for multiple attributes
- Core argumentation can be provided by GSN (The Goal Structuring Notation)
  - Already well established for safety arguments
- Extensions needed to capture flexible objectives, tradeoffs, analysis of design alternatives
- GSN + Extensions captured within KM3 metamodel
  - Basis for discussion of Dependability Case extensions
- Metamodel provides basis for tool support and data exchange formats