Standardization of GSN Patterns in OMG sysA PTF

Yutaka Matsuno, University of Tokyo
Kenji Taguchi, AIST
Hiroki Takamura, DEOS R&D Center

Why should GSN patterns be standardized?

- Reuse of existing safety/assurance/ dependability cases
  - mandatory in order to improve productivity
- Tool support

What are needed to standardize GSN Patterns?

- What should be considered?
  - Patterns Catalogue
    - Catalogue for each system domain
    - Catalogue for each abstraction level
  - Pattern Description
  - New constructs to describe GSN patterns
    - Already specified in the new draft GSN standard (ver. 1.0)

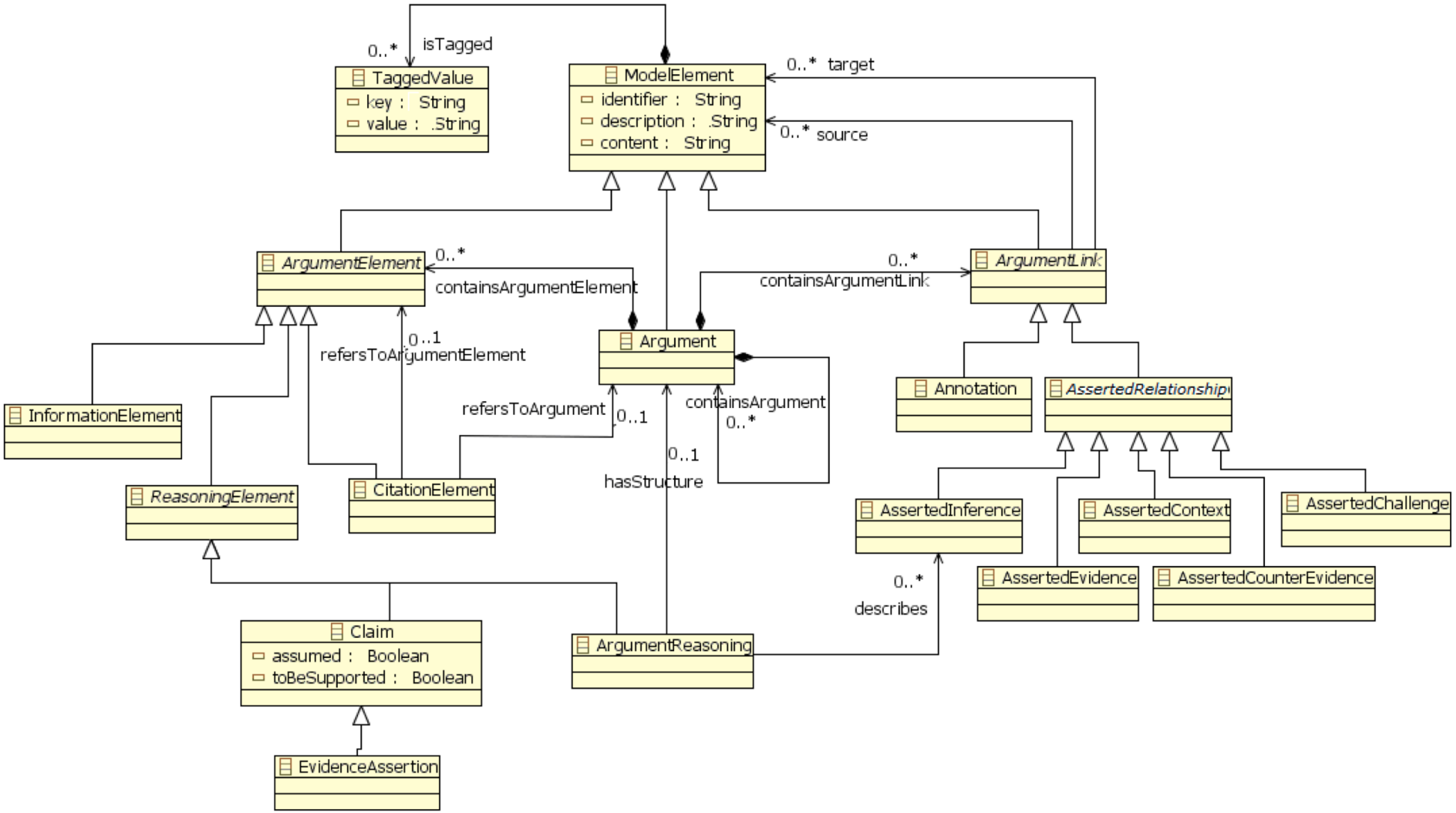
Pattern Catalogue from T. Kelly ([1])

- ALARP (As Low As Reasonably Practicable) Argument
- Hazard Directed Integrity Level Argument
- Control System Architecture Breakdown
- Diverse Argument
- Safety Margin
- Fault Tree Evidence
- Safety Principle 6 (Defence in Depth) Compliance Pattern

What should be standardized and how it should be standardized

- What should be targeted?
  - The ARM (or the upcoming SACM) should be extended to include new constructs to describe GSN patterns.
- How it should be designed?
  - Design Principle
    - Better to add it as a plug-in. Better not to alter any core elements and the design principle in the ARM (or the upcoming SACM).
- State of the art of GSN patterns
  - mature

ARM (ver. 1, March 2010)



Example Revision 1 : Placeholder Expression

New extension:

A place holder expression introduces a variable in expressions appeared in several constructs (Goals, etc) in GSN.

Possible revision:

Argument Element class.

Potential Problem:

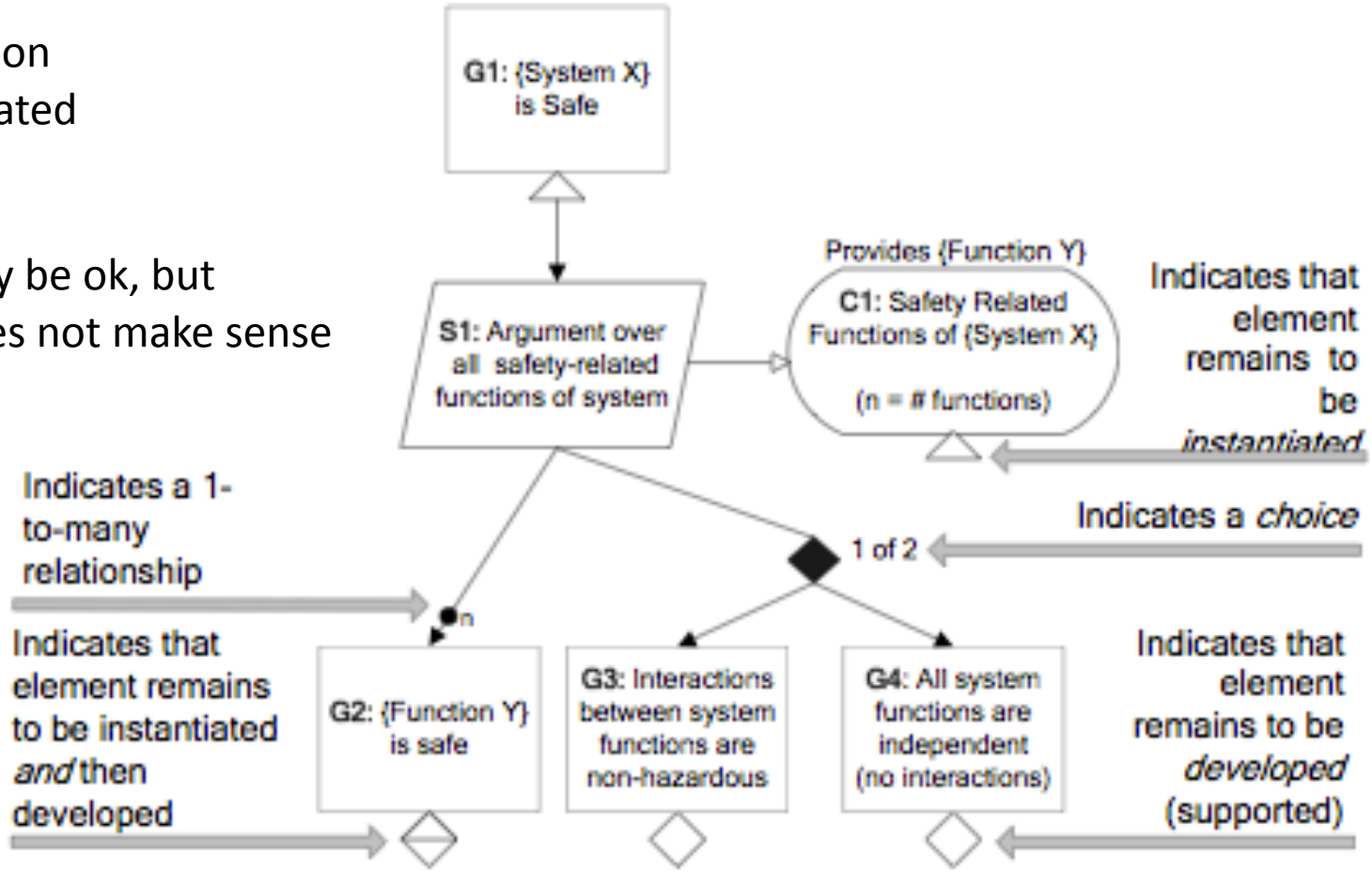
If a place holder expression is introduced in the *Claim*, it may not have any truth value even it has an attribute *assumed: Boolean*. So we have to have differences between free expressions and closed expressions just like in formal languages, e.g., first-order predicate logic.

An Example of Placeholder [4]

Placeholders need to be instantiated

We need a restriction for X to be instantiated

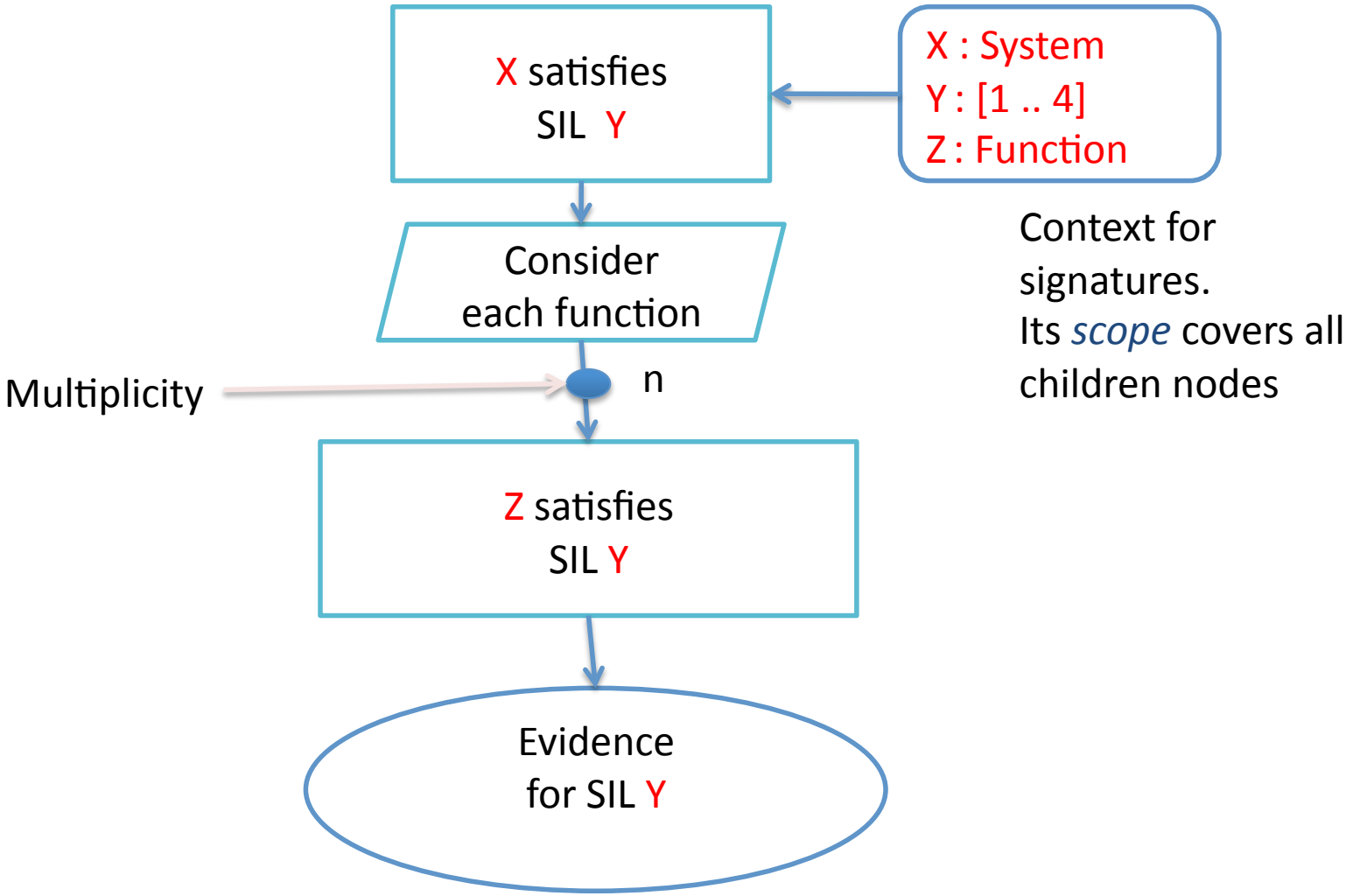
e.g
X = C/S logic -> may be ok, but
X = Dr. Who -> does not make sense



Introducing Signatures for placeholders

- Approach to the problem: give placeholders to types
  - Type checking ensures certain correctness
  - Signatures (placeholders → types) should be added to patterns
- Example of types
  - Data Types
    - int
    - string
    - …
  - Object Types
    - System
    - Function
    - …
  - User Defined Types

Example of Signatures in Pattern



What to be standardized for signatures

- Defining Types
  - Data Types : int, string, float, …
  - Object Types : System, Function, Failure, …
  - User defined Types
    - We need to consider the mechanism

Example Revision 2 : Multiplicity

New extension:

Multiplicity attached to *solved by* link (in GSN).

Possible revision:

ArgumentLink class.

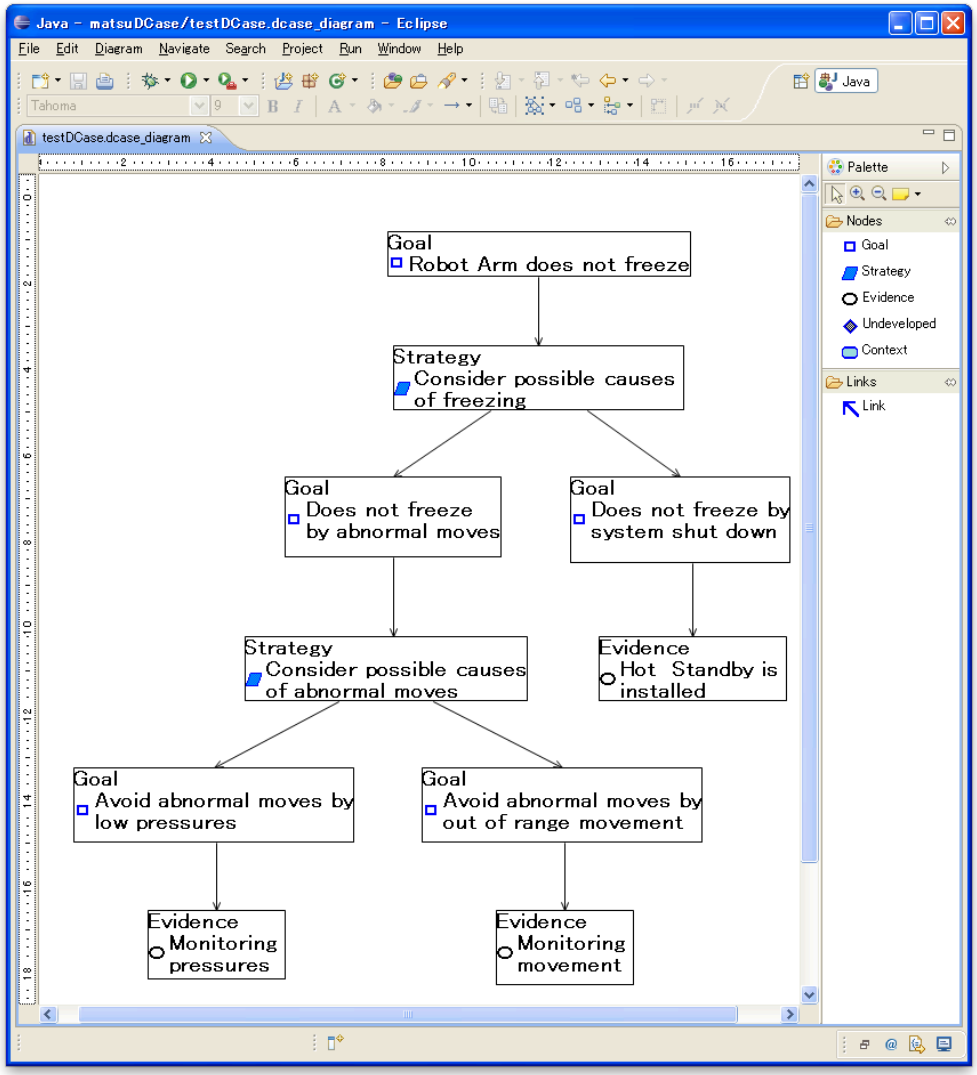
Potential solution:

This could be expressed as a new attribute *multiplicity* in *ArgumentLink* class.

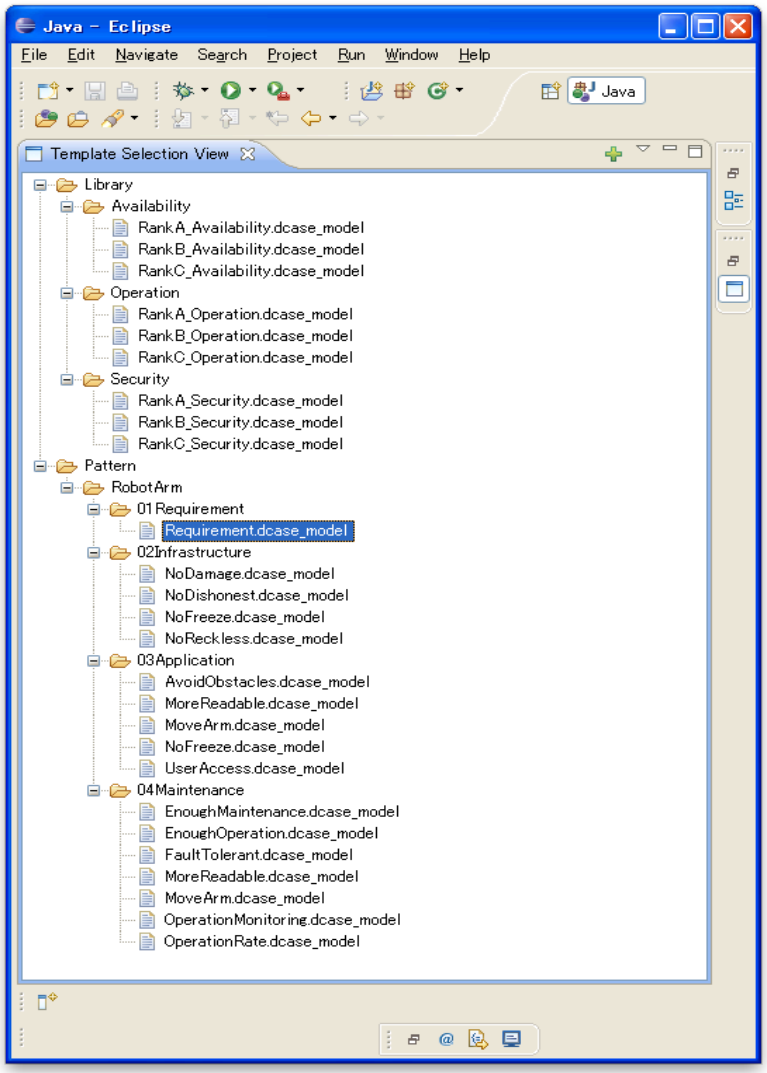
Current Our Activity

- Collecting dependability case [3] patterns
  - Based on several Japanese non-functional requirements guidelines
  - Patterns specific to movable robotics arm
    - An example of domain specific patterns
- Implementing a prototype dependability case editor called ``D-Case Editor”
  - pattern selection function from library
    - a fast abstract is to appear in IEEE HASE 2010 [5]
  - basic signatures function has been designed

A Snap Shot of D-Case Editor



An Example of Pattern for Movable Robotics Arm



Pattern Selection View

Roadmap

- RFI (Dec 2010)
- ?

Reference

[1] T. Kelly: Arguing Safety – A Systematic Approach to Managing Safety Cases, PhD thesis, U. York, 1998.

[2] T. Kelly, et. al.,: Draft GSN Standard Version 1.0, 2010.

[3] G. Despotou: Managing the Evolution of Dependability Cases for Systems of Systems, PhD thesis, U. York, 2007.

[4] R.Alexander, et. al.,: Safety Cases for Advanced Control Software: Safety Case Patterns, Tech. Report, U. York, 2007

[5] Y.Matsuno, et. al.,: A Dependability Case Editor with Pattern Library, to appear in HASE 2010 as a fast abstract