Model-Based Assurance Cases

Richard Hawkins

Department of Computer Science,
The University of York
Assurance Cases

• Assurance – *Justified confidence in a property of interest*
  ▪ Often safety or security, but could also reliability, maintainability etc

• Assurance case – *Documents the argument and supporting evidence to demonstrate sufficient assurance is achieved*
  ▪ Both the evidence *and* argument must be present
  ▪ The argument *explains* how the evidence supports claims of assurance

• Assurance arguments are easier to understand when represented graphically
  ▪ We use Goal Structuring Notation (GSN) for this
Assurance Argument Patterns

Goal 1
{System X} is acceptably safe

Strategy 1
Argument over all safety related functions implemented by system

Context 1
The safety related functions of {system X} are {functions}

n = no. of safety related functions

Goal 2
{Function Y} is acceptably safe

Goal 3
Interactions between system functions are non-hazardous

Goal 4
All system functions are independent (no interactions)
Assurance Argument Patterns

Target system – car braking system

Goal 1
(System X) is acceptably safe

Strategy 1
Argument over all safety related functions implemented by system

Context 1
The safety related functions of (system X) are (functions)

Goal 2
(Function Y) is acceptably safe

Goal 3
Interactions between system functions are non-hazardous

Goal 4
All system functions are independent (no interactions)
Assurance Argument Patterns

Goal 1
(System X) is acceptably safe

Strategy 1
Argument over all safety related functions implemented by system

Context 1
The safety related functions of (system X) are (functions)

Goal 2
(Function Y) is acceptably safe

Goal 3
Interactions between system functions are non-hazardous

Goal 4
All system functions are independent (no interactions)

Goal 1
Car braking system is acceptably safe

Context 1
The safety related functions of the car braking system are the monitoring and processing functions.

Goal 2
Braking monitoring function is acceptably safe

Goal 3
Braking processing function is acceptably safe

Goal 4
System functions are independent (no interactions)
Assurance Argument Patterns

**Goal 1**
(SYSTEM X) is acceptably safe

**Context 1**
The safety-related functions of (SYSTEM X) are functions

**Goal 2**
Braking monitoring function is acceptably safe

**Goal 3**
Braking processing function is acceptably safe

**Goal 4**
System functions are independent (no interactions)

---

The safety-related functions of the car braking system are the monitoring and processing functions.
Information from Diverse Sources
Pattern Instantiation

- Argument patterns essentially define information requirements
  - to instantiate the assurance claims, provide evidence and make instantiation choices
- Information required from multiple sources
- Possible to manually obtain this to instantiate pattern
  - But, repetitive and mechanistic in nature and prone to human error
- Instead we can try to automate...
Benefits of automation

• Argument generated directly from, and therefore consistent with, the design and development models themselves.

• Snap-shot instantiations can be produced quickly and easily to reflect the current state of development.

• Consistent, reusable instantiation rules are established, ensuring consistent instantiations.
  ▪ particularly important where complicated relationships between multiple models are required.

• Human instantiation error is mitigated
  ▪ also becomes easier to check and verify the resulting output.
Benefits of automation 2

• Facilitate design assessment
  ▪ i.e. highlighting claims and evidence needed for assurance but not available in the system information models

• Automated support for change management

• Generated argument is still in a format that is amenable to human review.

• Based on the current models of the system, areas of argument requiring further development and support are highlighted
  ▪ This allows the human effort to be focused on analysing and addressing those areas where most value is added.
A Model-Based Approach
Key advantages of approach

• **Tool and notation independent**
  - approach does not rely on the use of any particular tools.
  - Tools only required to provide input models that conform to their own defined metamodels to be compatible.
  - Notation also unimportant so long as an XML representation of the model can be provided.

• **Weaving model**
  - Links reference information metamodels to the patterns
  - Captures dependencies between
    - role bindings in GSN patterns -> individual reference information metamodels
    - multiple reference information metamodels
  - Normally these dependencies are implicit.
  - Using weaving model makes this dependency information both explicit and formally defined.
Key advantages of approach 2

• Assessment and validation
  ▪ possible to partially automate analysis and validation of the assurance case
  ▪ Most automated analysis focuses on verification of the argument
    ○ Our approach allows for assessment w.r.t. external information models

• Use existing MBE tools
  ▪ quickly and easily add extra functionality and features
  ▪ **EOL** - use EOL for the main instantiation program
  ▪ **EVL** – allows us to automatically check and enforce sets of constraints
  ▪ **EGL** - use EGL to generate the output of the instantiation program. Flexibility to provide a number of options for the output format
Example model-based instantiation
GSN pattern inputs

• Consider GSN patterns as set of models that input to the instantiation program.
• Requires that the GSN patterns are documented in a machine readable format that conforms to a defined meta-model.
• No official GSN meta-model
  ▪ OMG standard meta-model for structured assurance cases (SACM)
  ▪ We have extended this for GSN
  ▪ Proposed for inclusion in GSN standard
Simple example system

- Cryptographic Controller
- GSN Pattern Models

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<Argumentation>
   <GSN_Claim id="G1" toBeInstantiated=true>
      <Claim_Type>Goal<Claim_Type>
      <Text>[DMILS System] security policy is enforced</text>
      <Role_Binding>DMILS System</Role_Binding>
   </GSN_Claim>
   <GSN_ContextAsReference id="C2" toBeInstantiated=true>
      <Text>The system security policy is [system security policy]</text>
      <Role_Binding>security policy</Role_Binding>
   </GSN_ContextAsReference>
   <AssertedRelationship id="R1">
      <Relationship_Type>GSN_SupportedBy<Relationship_Type>
      <Source>G1</Source>
      <Target>St1</Target>
   </AssertedRelationship>
   <GSN_Strategy id="St1" toBeInstantiated=true>
      <Text>Argument over the individual software component behaviour and the compositional behaviour of {DMILS Sys</text>
      <Role_Binding>DMILS System</Role_Binding>
   </GSN_Strategy>
   <GSN_Claim id="G2" toBeInstantiated=false>
      <Claim_Type>Goal<Claim_Type>
      <Text>Trusted software components behave according to the policy architecture</Text>
   </GSN_Claim>
   <GSN_Claim>
      <GSN_Claim id="St2" toBeInstantiated=false>
      <Text>Argument over each trusted software component</text>
   </GSN_Claim>
   <AssertedRelationship id="R2">
      <Relationship_Type>GSN_SupportedBy<Relationship_Type>
      <Source>St1</Source>
      <Target>G4</Target>
   </AssertedRelationship>
   <GSN_Claim id="G4" toBeInstantiated=true>
      <Claim_Type>Goal<Claim_Type>
      <Text>[software component] enforces its local policy</text>
      <Role_Binding>software component</Role_Binding>
   </GSN_Claim>
</Argumentation>
```
Generating Pattern Models
Example automated instantiation
Reference information models

• The set of models containing the information required for the instantiation of the assurance argument
• The models required are determined from the “role bindings” of instantiable elements in patterns
• Each model may have a different meta-model
  ▪ this must be explicitly defined such that it can be used in creating the weaving model.
  ▪ place no restrictions on tools and notations used to generate the models
Example AADL reference model

system CryptoController
features
  inframe: in data port Frame;
  outframe: out data port Frame;
end CryptoController;

system implementation CryptoController.Imp
  subcomponents
    red: node Splitter.Imp accesses channels;
    bypass: node Bypass.Imp accesses channels;
    crypto: node Crypto.Imp accesses channels;
    black: node Merger.Imp accesses channels;
    channels: network CryptoNet.Imp;
flows
  port inframe -> red.frame;
  port red.header -> bypass.inheader;
  port red.payload -> crypto.inpayload;
  port bypass.outheader -> black.header;
  port crypto.outpayload -> black.payload;
  port black.frame -> outframe;
end CryptoController.Imp;

node Splitter
features
  frame: in data port Frame;
  header: out data port Header;
  payload: out data port Payload;
end Splitter;
node implementation Splitter.Imp
flows
  port fst(frame) -> header;
  port snd(frame) -> payload;
end Splitter.Imp;
Reference information models 2

- Normally multiple models will be required
- Example: pattern for error behaviour of a component…

<table>
<thead>
<tr>
<th>Role Binding</th>
<th>Reference Information Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>AADL specification</td>
</tr>
<tr>
<td>Component Error Model</td>
<td>AADL error model specification</td>
</tr>
<tr>
<td>Error Effect</td>
<td>FMEA analysis results</td>
</tr>
<tr>
<td>Stage</td>
<td>FMEA process model</td>
</tr>
</tbody>
</table>

- Note interrelationships between these models
  - E.g. AADL error model and FMEA analysis
Informally defined models

- Some models informally defined
  - E.g. FMEA analysis process description
  - Still possible to generate simple XML representations sufficient to input to the instantiation program.
Example automated instantiation
Weaving Meta-model

• Model weaving is an approach to model transformation
  - handles fine-grained relationships between elements of distinct models, establishing links between them.
  - model weaving takes multiple models as input
  - the mappings between the models themselves are also considered as models

• Existing approaches and tools support the creation of weaving models
Example Weaving Model
Example automated instantiation

Diagram:
- Reference Information Metamodel
- Reference Information Model
- Instantiation program
- GSN Pattern
- GSN Metamodel
- Weaving Metamodel
- Instantiation Model
- GSN Argument
Instantiation program

- EOL program that runs on the Eclipse platform.
  - **Input:**
    - GSN argument pattern models
    - Reference information models
    - Weaving model
  - **The instantiation program:**
    1. Identifies elements requiring instantiation in the pattern model
    2. Identifies information from the reference models required to instantiate pattern by querying weaving model
    3. Obtains required information from the relevant information models
    4. Outputs instantiation information
Example automated instantiation
Instantiation Output

• Many different ways in which the instantiation program can output the instantiation information
  ▪ output is presented in a manner that is easily and clearly understood by a human
  ▪ Instantiation table
  ▪ GSN model
    o Compatible with existing GSN editors
    o We have imported this to graphical editor
Instantiation Table

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<table>
  <entry id="G1.0" object="[Crypto Controller System]" roleBinding="DMILS System"/>
  <entry id="G4.0.1" object="[Crypto]" roleBinding="software component"/>
  <entry id="G10.1" object="[All outgoing information shall be encrypted]" roleBinding="local policy"/>
  <entry id="St10.1" object="[Crypto Controller System AADL Design Model]" roleBinding="tier"/>
  <entry id="C20.1" object="[encrypt(inpayload, mykey) -&gt; outpayload]" roleBinding="software requirement"/>
  <entry id="G11.1.1" object="[encrypt(inpayload, mykey) -&gt; outpayload]" roleBinding="software requirement"/>
  <entry id="G4.0.2" object="[Bypass]" roleBinding="software component"/>
  <entry id="St10.2" object="[Crypto Controller System AADL Design Model]" roleBinding="tier"/>
  <entry id="G4.0.3" object="[Red]" roleBinding="software component"/>
  <entry id="G10.3" object="[All payload information shall be stripped from the header]" roleBinding="local policy"/>
  <entry id="St10.3" object="[Crypto Controller System AADL Design Model]" roleBinding="tier"/>
  <entry id="C20.3" object="[fst(frame) -&gt; header; snd(frame) -&gt; payload;]" roleBinding="software requirement"/>
  <entry id="G11.3.2" object="[fst(frame) -&gt; header; snd(frame) -&gt; payload;]" roleBinding="software requirement"/>
  <entry id="G4.0.4" object="[Black]" roleBinding="software component"/>
  <entry id="St10.4" object="[Crypto Controller System AADL Design Model]" roleBinding="tier"/>
  <entry id="St1.0" object="[Crypto Controller System]" roleBinding="DMILS System"/>
</table>
```

<table>
<thead>
<tr>
<th>{DMILS System}</th>
<th>{software component}</th>
<th>local policy</th>
<th>{tier}</th>
<th>{software requirement}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crypto Controller System</td>
<td>Crypto</td>
<td>All outgoing information shall be encrypted</td>
<td>Crypto Controller System AADL Design Model</td>
<td>encrypt(inpayload, mykey) -&gt; outpayload</td>
</tr>
<tr>
<td></td>
<td>Bypass</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Red</td>
<td>All payload information shall be stripped from the header</td>
<td>Crypto Controller System AADL Design Model</td>
<td>fst(frame) -&gt; header; snd(frame) -&gt; payload;</td>
</tr>
<tr>
<td></td>
<td>Black</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
<xml version="1.0" encoding="UTF-8" standalone="no"?>
<Argumentation>
  <GSN_Claim id="G1.0" toBeInstantiated="true">
    <Claim_Type>Goal</Claim_Type>
    <Text>[[DMILS System} security, policy is enforced]</Text>
    <Role_Binding>[[Crypto Controller System]]</Role_Binding>
  </GSN_Claim>
  <GSN_Claim id="G2" toBeInstantiated="false">
    <Claim_Type>Goal</Claim_Type>
    <Text>[[Trusted software components, behave according to the, policy architecture]]</Text>
    <Role_Binding> </Role_Binding>
  </GSN_Claim>
  <GSN_Claim id="G3" toBeInstantiated="false">
    <Claim_Type>Goal</Claim_Type>
    <Text>[[The composition of the, DMILS system guarantees, the system security policy, is met]]</Text>
    <Role_Binding> </Role_Binding>
  </GSN_Claim>
  <GSN_SupportedBy id="RG4.0.1">
    <Source>M1</Source>
    <Target>G4</Target>
  </GSN_SupportedBy>
  <GSN_SupportedBy id="RG10.1">
    <Source>G4</Source>
    <Target>G10</Target>
  </GSN_SupportedBy>
  <GSN_Claim id="G10.1" toBeInstantiated="true">
    <Claim_Type>Goal</Claim_Type>
    <Text>G10</Text>
    <Role_Binding>[[All outgoing information shall be encrypted]]</Role_Binding>
  </GSN_Claim>
</Argumentation>
Future Directions

• This approach opens up lots of model-based engineering solutions to the assurance case domain
  ▪ Model validation
  ▪ Model simulation
  ▪ Change management
  ▪ Real feedback to design process
  ▪ Use of formal constraints

• Assurance cases as engineering rather than an art!