Architecture-driven, Multi-concern and Seamless Assurance and Certification of Cyber-Physical Systems

Architecture-Driven Assurance

First EAB Workshop
Trento, September 11, 2017

Stefano Puri
WP3 Leader
Architecture-Driven Assurance (AiDA): an assurance that exploits and is linked to the system architecture in order to provide more structured evidences and arguments to show that the system is free of vulnerabilities.

Main specific objectives:
1. Determine the needs and constraints based on which the approach will be developed.
2. Provide a conceptual framework that meets the information needs for architecture-driven assurance.
3. Specify the design of AMASS building blocks for applying the architecture-driven approach.
4. To implement the AMASS tools above.
5. To provide methodological guidance for the application of the approach and the maintenance of its results.
AiDA work areas

• System Architecture Modelling for Assurance
  – Exploit the system architecture in the assurance case
  – System architecture languages
  – Architecture trade-off and comparison
• Architectural Patterns for Assurance
  – Interaction between assurance and architectural patterns
  – Architectural patterns from standards
• Contract-based assurance
  – Assurance patterns for contract-based design
  – Enrich evidence produced by contract-based design
• V&V-based assurance
  – Enrich V&V techniques

<<Contract>>
Formalize properties of system and components in terms of assumptions and guarantees properties
The system architecture is used for model-driven engineering, supporting contract-based, pattern-based design, V&V.

How to link assurance models and system has received some attention in the literature, but not much in our opinion.
Architecture Driven Assurance Implementation

AMASS Platform Basic Building Blocks

- Access Manager
- Data Manager
- System Component Specification
- Assurance Case Specification
- Evidence Management
- Compliance Management
- Common Assurance & Certification Metamodel (CACM)

Specify system architecture at different levels
Browse, edit, update and drop component specifications at different abstraction levels
Specify component behavioral model
Specify component fault model
Allocate requirements
Monitor status of system specification
Reconfigure component
Assign contract to component
Refine contracts
Browse component contract status
Assurance Engineer
System Component Specification

- Polarsys Papyrus and CHESS projects as basic building blocks
  - Papyrus allows UML/SysML and DSL modelling facilities, integration with external models/tools (requirements import/export from different sources, import from Rhapsody...)
  - CHESS is methodology and toolset for model-based design, validation and implementation of high-critical software application (result of Artemis JU CHESS and CONCERTO projects)
    - Customizes UML/SysML/MARTE and Papyrus tool
    - Provides support for contract-based design
      - Originally created in the context of SafeCer project
      - Extended in AMASS to support AiDA meta-model
"Humanoid robots family is growing"
Aldebaran

"A collaboration platform for avionics systems"
DGA

"Design, architects, and complex communication systems: painting the bigger picture"
Ericsson

"Designing freedom of flight"
Airbus

"...partners have already built industrial tools based on Papyrus to support their domains and their customers."
OpenCert: powered by CHESS

- CHESS has been extended to allow modelling traceability between contracts/formal properties, claims and evidences
- Demo...
Features for Architecture-Driven Assurance
Functionalities that are focused on the modelling of the system architecture to support the system assurance.
Contract-based design for Assurance

- Contract-Based Design for Assurance: functionalities that support the contract-based design of the system architecture, which provides additional arguments and evidence for the system assurance.

Components reuse that is supported by checks on the contracts.

Generation of assurance arguments from the contract specification and validation.

Defining system boundaries is a difficult task…
Architectural Patterns for Assurance

- Architectural Patterns for Assurance: functionalities that are focused on architectural patterns to support the system assurance
  - Management of a library of architectural patterns
  - Application of architectural patterns by using associated contracts

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>define in which context the pattern is used. For example, define if the pattern is recommended for a specific safety-critical domain.</td>
</tr>
<tr>
<td>PatternAssumptions</td>
<td>the contract assumptions related to the design pattern.</td>
</tr>
<tr>
<td>PatternGuarantees</td>
<td>the contract guarantees related to the design pattern.</td>
</tr>
</tbody>
</table>
Functionalities that are focused on advanced analysis to enrich the evidence of the assurance case.
## V&V ex2: Contract-based verification and analysis

### Modeling and Analysis

<table>
<thead>
<tr>
<th>Architecture decomposition &amp; Contracts</th>
<th>V &amp; V</th>
<th>Safety Assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Automatic contract refinement verification</td>
<td>Fault extension</td>
<td>• Automatic fault extension</td>
</tr>
<tr>
<td>• Automatic compositional verification</td>
<td>Fault trees computation</td>
<td>• Automatic hierarchical fault tree generation</td>
</tr>
<tr>
<td>• Automatic monolithic verification</td>
<td></td>
<td>• Over-approximation</td>
</tr>
</tbody>
</table>

### Behavioral Implementation (Leaf components & System)

<table>
<thead>
<tr>
<th>M</th>
<th>OCRA</th>
<th>nuXmv</th>
<th>xSAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault extension</td>
<td>M \models \varphi</td>
<td>M \rightsquigarrow M_F</td>
<td>\delta(F) : M_F \not\models \varphi</td>
</tr>
</tbody>
</table>

1st EAB Workshop, Trento, September 11, 2017
Main Achievements

- Way forward has been identified with respect to the current state of the art and state of the practice
- Initial definition of the system component metamodel for assurance, with links between architectural and assurance case entities
- Basic building blocks for system architecture specification have been provided
- Initial definition of the conceptual level for architecture-driving assurance, as set of functionalities to be provided on top of the AMASS platform
AMASS Reference Tool Architecture, AiDA Scope: implementation status

Functionalities identified. Implementation in progress

Functionalities identified. Implementation provided and validated (core prototype)

1st EAB Workshop, Trento, September 11, 2017
Summary and future work

• Definition of techniques and functional architecture supporting AiDA
  – Assurance case fragments definition
• Definition of guidelines
• Support for the AMASS second prototype
Thank you for your attention!