Safety Critical Software and Systems Research @ General Electric

July 13, 2016

Imagination at work.
Software & Systems Challenges @ GE

• Critical Infrastructure
  – Aerospace, Power Generation, Transmission & Distribution, Oil & Gas, Healthcare, Transportation
  – Systems becoming more software intensive, distributed, autonomous & connected to the Industrial Internet
  – Focused on technology & process to develop and verify protection, control, optimization and security
  – Goal: measurable security, efficiency, and cycle time benefits at scale
Team, Approach and Status

Team Structure

- **GE Research**
  - Computer Science, Controls & Embedded Systems
  - Niskayuna, NY, Munich, Germany, Rio de Janeiro, Brazil
- **GE Aviation**
  - Chief Engineers Office (CEO)
  - Product Line Engineers and Management
  - Certification Team
  - Tools Organization
  - Grand Rapids, MI, Evendale, OH, Clearwater, FL
- **Academia**
- **Government Agencies**

Approach

- Based on DO-178C, DO-333 (FM) & DO-331 (MBD)
- Develop Technology & Tools
- Pilot with Product Teams
- Transition Process & Best Practices Through CEO
- Transition Technology Through Tools Organization

Key Technologies & Status

- Formal Modeling & Analysis
- Automated Test Case Generation
- Design Model Extraction from Legacy Software
- Compositional Verification of Software/Systems
- Formal Modeling and Analysis of Multi-Time Scale Systems
- Architectural and Functional Synthesis
- Composition and Verification of Autonomous Systems
ASSERT™ Tool Chain

Analysis of Semantic Specifications and Efficient generation of Requirements based Tests

Requirements Capture
- Built on SADL
- Structured English Semantic Model
- DOORS® interface

Requirements Analysis
- Built on ACL2s
- Theorem Proving
- Conflict detection
- Completeness

Auto Test Generation
- Multiple solvers
- Model Checking
- Test Cases
- Test Procedures

Test Procedure Translator
- Maps to Execution
- SCADE Suite®
- VectorCAST®
- GE Environment

© 2016, General Electric Company
Requirements Capture
1.3. Generator Can Connect
The generator is connect-capable when generator is off-grid and generator frequency equal to grid frequency.

\[ \text{off-grid AND generator frequency} = \text{grid frequency} \]

implies
\[ \text{connect-capable} = \text{true} \]
1.3. Generator Can Connect
The generator is connect-capable when generator is off-grid and generator frequency equal to grid frequency.

off-grid AND generator frequency == grid frequency
implies
connect-capable = true

• What is a “generator”?  
• What is a “grid”?  
• What does “off-grid” mean?  
• What is a “frequency”?  
• What is a “connect-capable”?
A **Semantic Model** Underpins a Requirement and Answers These Kinds of Questions

### 1.3. Generator Can Connect

The generator is connect-capable when generator is off-grid and generator frequency equal to grid frequency.

- off-grid **AND** generator frequency **==** grid frequency **implies**
  - connect-capable **=** true

- What is a “generator”?
- What is a “grid”?
- What does “off-grid” mean?
- What is a “frequency”?
- What is a “connect-capable”?
Semantics is about Meaning: Where Does Meaning Come From?
What Might This Mean?

<xml...>
    ....
    <name>
        <given-name>Andrew</given-name>
        <surname>Crapo</surname>
    </name>
    <email>crapo@research.ge.com</email>
    ....
</xml>
And This?

<xml...>
   ....
   <cart>
      <item>Tom Sawyer</item>
      <price>$19.95</price>
   </cart>
   <total>$25.98</total>
   ....
</xml>
Meaning Is Not Captured by Syntax

<xml...>
    ....
    <name>
        <given-name>Andrew</given-name>
        <surname>Crapo</surname>
    </name>
    <email>crapo@research.ge.com</email>
    ....
</xml>

<xml...>
    ....
    <cart>
        <item>Tom Sawyer</item>
        <price>$19.95</price>
    </cart>
    <total>$25.98</total>
    ....
</xml>
Hypothesis: Meaning Is an Emergent Characteristic of Structure

But not syntactic structure....

Some structure that you have in your mind
Hypothesis: Meaning Is an Emergent Characteristic of Structure

But not syntactic structure....

Some structure that you have in your mind

Can we create a meaning-containing structure in a computable artifact?
Hypothesis: Meaning Is an Emergent Characteristic of Structure

But not syntactic structure....
Some structure that you have in your mind

Can we create a meaning-containing structure in a computable artifact?

An ontology (aka semantic model) is a structuring of metadata (tags in XML) that gives it (them) meaning.
Definition of Concepts in a Semantic Model
(expresssed in the Semantic Application Design Language or SADL)

**Generator** is a class, described by **connect-capable** with values of type boolean, described by **status** with values of type **ConnectStatus**. **ConnectStatus** is a class, must be one of {On-Grid, Off-Grid}.

**Grid** is a class.

**frequency** is a property with values of type int. **frequency** describes {Generator or Grid}.

**EMS** (alias "Energy Management System") is a class. **managedBy** describes **Generator** with values of type **EMS**. **connectsTo** describes **EMS** with values of type **Grid**.
Semantic Application Design Language (SADL)

- Formal language
- Eclipse-based IDE
- Integrated version control & testing
- Semantic coloring
- Continuous error checking

- Templates
- Suggested completions
- Hyperlinking
- Multi-project integration
- Documentation and Context Help

Built on Xtext (See http://sadl.sourceforge.net)
Project Explorer

* EnergyManagement_v1
  * EnergyManagement_v1.1
    * OwlModels
      * BaseConcepts.sadl
        * EnergyManagement.sadl
      * RoomLighting
      * Sports_complete
      * Sports_recomplete

EnergyManagement.sadl

```

Generator is a class,
described by connect-enable with values of type boolean,
described by status with values of type ConnectStatus.
ConnectStatus is a class, must be one of {On-Grid, Off-Grid}.

grid is a class.

frequency is a property with values of type int.
frequency describes (Generator or Grid).

ems (alias "Energy Management System") is a class.
managedBy describes Generator with values of type EMS.
connectsTo describes EMS with values of type Grid.
```

SADL

Beginning clean process...
Clean/build process completed for 1 project in 281 ms
No Direct Write or XML model output generated for 'platform://resource/EnergyManagement_v2/EM_MGR.sreq'.
Model totals: 0 errors, 0 warning, 0 infos

© 2016, General Electric Company
The Previous Example in the SADL Requirement Language (SRL)

1.3. Generator Can Connect
The generator is connect-capable when generator is off-grid and generator frequency is equal to grid frequency.

import "http://sadl.org/EnergyManagement.sadl".

Context: Generator.

Requirement R1.3: EMS shall set connect-capable to true when status of Generator is Off-Grid and frequency of Generator == frequency of connectsTo of managedBy.
The Previous Example in the SADL Requirement Language (SRL)

1.3. Generator Can Connect
The generator is connect-capable when generator is off-grid and generator frequency is equal to grid frequency.

```plaintext
import "http://sadl.org/EnergyManagement.sadl".

Context: Generator managedBy EMS connectsTo Grid.

Requirement R1.3: EMS shall set connect-capable to true when status is Off-Grid and frequency of Generator == frequency of Grid.
```
The Previous Example in the SADL Requirement Language (SRL)

1.3. Generator Can Connect
The generator is connect-capable when generator is off-grid and generator frequency is equal to grid frequency.

```
import "http://sadl.org/EnergyManagement.sadl".

Context: Generator managedBy EMS connectsTo Grid.

Requirement R1.3: EMS shall set connect-capable to true when status is Off-Grid and frequency of Generator == frequency of Grid.
```
The Previous Example in the SADL Requirement Language (SRL)

1.3. Generator Can Connect
The generator is connect-capable when generator is off-grid and generator frequency is equal to grid frequency.

Model(s) defining concepts

Disambiguation (if needed)

import "http://sadl.org/EnergyManagement.sadl".

Context: Generator managedBy EMS connectsTo Grid.

Requirement R1.3: EMS shall set connect-capable to true when status is Off-Grid and frequency of Generator == frequency of Grid.

Parsimonious statement of Requirement

`Generator` is a class,
- described by `connect-able` with values of type boolean,
- described by `status` with values of type `ConnectStatus`.
  `ConnectStatus` is a class, must be one of `(On-Grid, Off-Grid)`.

`Grid` is a class,

`BMS` (alias "Energy Management System") is a class.

managedBy describes `Generator` with values of type `BMS`.
connectsTo describes `BMS` with values of type `Grid`.
Requirements Analysis
Automated Test Generation
Automated Test Generation: Overall Process

1. Requirement Capture Tool
2. Requirement Translator
3. Requirement Analysis
4. Analysis Report
5. Automated Test Case Generation
6. Common Test Case File (.stst)
7. Automated Test Procedure Generation
8. Common Test Procedure File (.xml)

Next slide

© 2016, General Electric Company
Automated Test Generation: Overall Process

Common Test Procedure format (.xml)

- VAIS Test Procedure translator (VACON)
- VectorCast Test Procedure translator (VCCON)
- Rhapsody Test Procedure translator (RHAPCON)
- SCADE Test Procedure translator (SCACON)
- HM Test Procedure translator (C codeCON)

- .vba scripts
- .tst scripts
- .rhp scripts
- .sss scripts
- C code
Automated Test Generation: Test Strategies

Test Strategies → Reasoner → Test Objectives → Interpreter → Human-Readable Test Cases

Requirements → Translator → Intermediate Model → Synthesis → Machine-readable Test Procedures

Logic Coverage Analysis

Total seven test strategies being supported
Automated Test Generation Demo
Summary of Test Results

- All Test Passes
- Achieve coverage specified by the requirements
- Able to alert for potential unreachable code

Unreachable code

```plaintext
) return Boolean is

   --> operation Is_Gen_Valid(System_State_Type, Grid_Status_Type, Boolean, Boolean)
   -->
   Gen_Valid : Boolean;

   begin
   --> operation Is_Gen_Valid(System_State_Type, Grid_Status_Type, Boolean, Boolean)
   -->
   if Grid_Status = Off_Grid then
       --> Logic Step: Off_Grid
       Gen_Valid := (Grid_Con Capable and Ref_Freq_Available) and (Frequency <= 60);
   -->
   elsif Grid_Status = On_Grid then
       --> Logic Step: On_Grid
       return the value of Gen_Valid when the generator is on the grid
       Gen_Valid := (Active_Plan_Exists and System_State.Is_Valid and Protection_Is_Configured and Ref_Freq_Available) and (Frequency <= 60);

   -->
```

© 2016, General Electric Company
Thank you, questions & contact information

Michael Durling
durling@ge.com

Andrew Crapo, PhD
crapo@ge.com

Meng Li, PhD
meng.li3@ge.com