Architecture-led Pedagogical Artifacts as a Unifying Theme

John D. McGregor
Clemson University
SEI

Roselane S. Silva
Federal University of Bahia
Goal

• To reduce the time needed to understand/build an architecture by packaging semantically linked elements in a bundle to facilitate understanding and reuse.
Polling Questions

1. Interest in instruction?
   - Industry
   - University

2. Interest in construction of systems?

3. Knowledge of AADL?

4. Other ADLs?

5. Experience in model driven development?
Pedagogical Product Line

- Our previous experience in packaging assets:
  - Arcade Game Maker Pedagogical Product Line
  - [http://www.sei.cmu.edu/productlines/ppl/](http://www.sei.cmu.edu/productlines/ppl/)

- The basis for courses and research studies
Objective

- Describe a set of development assets based on an architecture-led development method that uses the **AADL** ecosystem within an Architecture Centric Virtual Integration Process (ACVIP).
Context

Architecture-Centric Virtual Integration Process (ACVIP)

- correct architecture design decisions
- traceability
- Cohesive model
- Artifact dependencies
- Model driven development

specific implementations
specific architectures
understanding
development
Slice and Dice

- Assets can be shaped in many ways
- All requirements
- A requirement and its tests
- A feature, a set of requirements, and related tests
- A set of product line assets

http://www.telegraph.co.uk/foodanddrink/foodanddrinkpicturegalleries/6932740/Food-news-by-Carolyn-Hart.html
My Mental Model - Asset Characteristics Meta-model

Architecture significance of assets

• An architecture design decision impacts
  – Requirements,
  – the architecture, and
  – the verification activities.
Assets in Architecture-led Development

- The most familiar form of asset is all of the instances of a particular type of asset
- The set of functional requirements, for example
Assets in Architecture-led Development [2]

• An asset may group together heterogeneous elements that have a specific relationship.
• Such as the architecture elements that satisfy a requirement and the tests that verify it.
Important properties of an asset

• Cohesion within the asset
• Ability to compose them safely
• Striking the correct balance between being sufficiently large to be useful and being sufficiently small to satisfy a specific need
Problem & Context

• A Tier 2 automotive supplier is developing a set of cruise control products to satisfy the needs of a number of Original Equipment Manufacturers.

• Since one member of the family is for totally autonomous vehicles, the family of products will be treated as safety critical products.
Problem domain

• Consider a family of vehicles that span from a traditional vehicle to a fully autonomous. Focus on the cruise control subsystem.

<table>
<thead>
<tr>
<th></th>
<th>Cruise Control (CC)</th>
<th>Adaptive CC (ACC)</th>
<th>Collaborative ACC (CACC)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Accelerate</strong></td>
<td>As needed</td>
<td>As needed</td>
<td>According to plan</td>
</tr>
<tr>
<td><strong>Brake</strong></td>
<td>As commanded</td>
<td>To maintain gap</td>
<td></td>
</tr>
</tbody>
</table>

• Quality attributes:
  – safety, performance, reliability, ride comfort...
Control/Feedback Loop

• The basic architecture for the product line of cruise controls is control/feedback loop

• The main design decision is how to design the controller for the product line

• Alternatives include
  – Three separate designs
  – A single container with changeable state machines
  – A single controller with one state machine

• But to make that decision it is necessary to understand the assets we have
What do we have to work with?

- A standard architecture
- A standard modeling language and ecosystem
  - AADL
  - Error ontology
  - Analysis tools
- Reusable analysis artifacts
  - Industry literature on the domain
Finite set of product line scenarios

(cc)

(acc)

(cacc)
AADL

• An SAE standard architecture description language
• Structured as a core language and a set of annexes
  – Error annex
  – Behavior annex
  – AGREE annex
• Syntax in text, xml, and graphical
• Extensible property sets
  – Memory properties
  – Thread properties

An annex is a DSL with its own tools

Perhaps the most powerful modeling construct
An AADL Component Definition

-- radar
device radarSensor_rt extends sensor::generic_sensor
features
sensor_data_out: refined to out data port data_types::radar_info;
BA: requires bus access platform::generic_bus{SEI::PowerBudget=>5.0W;};

annex EMV2 {**
use types error_library;
error propagations
sensor_data_out : out propagation {InvalidValue};
flows
ef0 : error source sensor_data_out{InvalidValue};
end propagations;

properties
emv2::hazards =>
([failure => "InvalidValue";
description => "Invalid distance sent by the radar";
])
applies to sensor_data_out.invalidvalue;
**};
end radarSensor_rt;
Instantiated system
Open Source AADL Tool Environment (OSATE)
Domain specific languages enable these artifacts
Even before any formal development begins we have some ideas about the product requirements and the product architecture. These initial ideas start an iterative process.
An experienced engineer recognizes a fundamental principle and provides an essential requirement.

Vehicles are subject to gravity.

Control speed

Requirements

Architecture

Implementation
That recognition leads to a possible architecture style.

Vehicles are subject to gravity.
Feedback control loop

- Set point
- Disturbance
- Control
- Actuators
- Process
- Sensors
A possible failure is identified given the style being used. Use the error ontology to help identify it.

Vehicles are subject to gravity.

Control speed

Requirements

Control feedback loop

Architecture

Error ontology

Implementation

Missing data
Error Annex of AADL
A new requirement is defined to handle the error condition.

Vehicles are subject to gravity.

If missing data use previous value

Error ontology

Control speed

Requirements

Architecture

Implementation

Control feedback loop

Missing data
Add an error flow to the architecture

Vehicles are subject to gravity.

If missing data use previous value

Control speed

Error ontology

Control feedback loop

Requirements

Architecture

Implementation

Define error flow for missing data

Missing data
Hardware fails differently from software. Bad sensor hardware will result in what types of errors?

What artifacts are related to this propagation?
Standard hazard analyses for a reference architecture

Used as guidewords on various analyses
Asset Design Discussion #1

What would an asset need to include?

- actuators
- sensors
- disturbance
- set point

Control

Process

Hardware errors propagate into software through incorrect readings.
Any time there is an input from hardware there should also be a validator for that type of data.
Asset Design Discussion #1

The Input data type, requirements/constraints on the input, validators of the input, and the flows of correct and incorrect values form an asset that is reusable and aids understanding.

Our goal is to facilitate building the asset
Domain specific languages

- Reqspeq - SEI
- Verify - SEI
- Assure - SEI
- AGREE – Rockwell Collins
- Resolute – Rockwell Collins
An organization is operating the process.
That organization has goals for a product

stakeholder goals caccStakeholderGoals for caccIntegration::cacc_rt
use constants caccConstants
[

description "Stakeholder goals for the family of cruise controls"

  goal g1 : "Safety" [
    description "The system shall only change modes when it is safe to do so."
    rationale "This is a control system, whose failure affects lives."
    stakeholder caccProject.rs
    quality safety
  ]
]

organization caccProject
stakeholder rs [
  full name "Roselane Santana Silva"
  title "Researcher"
  role "System modeler"
]
Cross reference between goal and stakeholder identity

stakeholder goals caccStakeholderGoals for caccIntegration::cacc_rt
use constants caccConstants
[

description "Stakeholder goals for the family of cruise controls"

  goal g1 : "Safety" [
  description "The system shall only change modes when it is safe to do so."
  rationale "This is a control system, whose failure affects lives."
  stakeholder caccProject.rs
  quality safety
  ]

]
stakeholder goals caccStakeholderGoals for caccIntegration::cacc_rt
use constants caccConstants
[

description "Stakeholder goals for the family of cruise controls"

    goal g1 : "Safety" [  
    description "The system shall only change modes when it is safe to do so."  
    rationale "This is a control system, whose failure affects lives. "  
    stakeholder caccProject.rs  
    quality safety
    ]

Reqspeq
Verify
Assure

Requirements

Architecture

Implementation
Reqspeq-System Requirements

requirement safety_R1
[

description this "The system responds to an excessive speed"
rationale "Handles a high speed situation"
compute cruiseStatus
value predicate cruiseStatus == false
mitigates "inadvertent too high throttle setting"
see goal caccStakeholderGoals.g1
see requirement speed_R1
category cc acc cacc
quality safety
uncertainty
  volatility 1
  impact 4
]
]

Reqspeq

Verify
Assure

Error ontology

Requirements

Architecture

Implementation
Verify – Verification Plan

verification plan CACCPlan for caccReqs
description "This is the verification plan for the requirements in caccReqs"

claim speed_R1: "The vehicle does not exceed maximum speed" [ rationale "This plan achieves a certification level of verification"
activities
  speed_ActTest1: "Test of speed control":
  caccVerificationMethods.comp(MaximumSpeed, CurrentSpeed)
property values ()
[ [ weight 2
  timeout 5
  ] ]
Verify – Verification Method

verification methods caccVerificationMethods: "Activities for caccReqs"

\[
\text{method } \text{comp (double } a, \text{ double } b)\text{boolean: } \text{compare two values}[^{\text{java caccverify2.caccTestComp.comp}} \\
\text{description } "\text{A test of two values}"]
\]

Requirements \xrightarrow{\text{Verify}} Architecture \xrightarrow{\text{Implementation}}
package caccverify2;
import org.osate.aadl2.instance.ComponentInstance;
@SuppressWarnings("all")
public class caccTestComp {
  public boolean comp(final ComponentInstance ci, final double a, final double b) {
    return (a > b);
  }
}

Verify – Java Implementation

Verify

Reqspec
Verify
Assure

Error ontology

Requirements
Architecture
Implementation

AGREE
Resolute
Java
OSATE Plug-ins

ArchitectureRequirements Implementation

Verify

Assure Error ontology

AGREE
Resolute
Java
OSATE Plug-ins
Verify – Resolute

SystemWideReq1() \leq \textbf{"All threads have a period"} \forall (t: \text{thread}). \text{HasPeriod}(t)

\text{HasPeriod}(t: \text{thread}) \leq \textbf{"Thread "} t \text{ " has a period"} \forall \text{has\_property}(t, \text{Timing\_Properties::Period})
Verify - AGREE

system controller extends controller_interface

annex agree {**
  guarantee "targetSpeed>=0" : targetSpeed >= 0 ;
  guarantee "actualSpeed>=0" : actualSpeed >= 0 ;
  guarantee "actualSpeed<=targetSpeed" : actualSpeed <= targetSpeed ;
  guarantee "gapLimit>0" : gapLimit > 0 ;
  guarantee "gap>0" : gap > 0 ;
  guarantee "gap<=gapLimit" : gap<=gapLimit;
**};
end controller;

Error ontology

Requirements

Architecture

Implementation

AGREE
Resolute
Java
OSATE Plug-ins

Verify

Assure

Verify
Verify – OSATE Plug-ins

verification methods Plugins[
  method ElectricalPower: ""
  plugin PowerAnalysis
]

Requirements

Architecture

Implementation

Error ontology

AGREE
Resolute
Java

OSATE Plug-ins

Verify

Verify

Assure

Verifying requirements
Asset Design Discussion #2

- Assets should be built as the artifacts are created.
- What level of granularity will be most useful?
- Who creates the artifacts?
Asset Design Discussion #2

• Goals are referenced by requirements
• Requirements are referenced by verification plans
• Architecture elements are referenced by requirements

stakeholder goals caccStakeholderGoals for cacclntegration::cacc_rt

see goal caccStakeholderGoals.g1

verification plan CACCPlan for caccreqs
Asset Design Discussion #2

• Any component definition or implementation can be referenced.
• Any named requirement can be linked
assurance case caccAssuranceCase: "Assurance case for the family of cruise control systems-CACC"
  for caccIntegration::cacc_rt

  assurance plan caccAssurancePlan: "The assurance case for the CACC implementation"
    for caccIntegration::cacc_rt.devices [
      assure CACCPlan
      issues "this assurance plan is related to the assurance case for CACC"
    ]

assurance task firstTest: "assurance task for one element" [
  category cc acc cacc
  quality security safety
  Reqs spec
  Verify
]

Assure
Error ontology
Requirements
Architecture
Implementation
caccProject.org
rs
jdm

caccGoals.goals
use constants caccConstants
goal g1 : "Safety" [stakeholder rs]

caccreqs.reqspec
requirement speed_R1 : for cacc_rt
[see goal caccGoalsDoc.g1
category cc acc cacc ]
use constants caccConstants

caccCat.cat
selection categories[
ccSel accSel caccSel ]

caccGlobals.constants
constants caccConstants
val MaximumSpeed = 120.0 mph

Caccreqs.reqspec
requirement speed_R1 : for cacc_rt
[see goal caccGoalsDoc.g1
category cc acc cacc ]
use constants caccConstants

caccPlan.verify
verification plan CACCPlan for caccreqs
activities
speed_ActTest1 for cacc_rt.devices
category ccSel
caccActivities.method1()

caccVerificationActivities.methodregistry
method method1 ():"first method"
[manual test5]

caccAssurance.alisa
Assurance task cameraLight
filter caccSel only

AGREE
assertions
Resolute
Predicates

AADL Model

ALISA Model

EMV2{'**'
mitigates'
**}
systemTypes.cat
requirement categories
cc acc cacc

ccGlobals.constants
constants caccConstants
val MaximumSpeed = 120.0 mph

caccGoals.goals
use constants caccConstants
goal g1 : "Safety"
  [stakeholder rs]

use constants caccConstants
caccPlan.verify
verification plan CACCPlan for caccreqs
activities
  speed_ActTest1 for cacc_rt
  [see goal caccGoalsDoc.g1
category cc acc cacc ]
use constants caccConstants
use constants caccConstants

Caccreqs.reqspec
requirement speed_R1 : for cacc_rt
[see goal caccGoalsDoc.g1
category cc acc cacc ]
use constants caccConstants
use constants caccConstants

caccPlan.verify
verification plan CACCPlan for caccreqs
activities
  speed_ActTest1 for cacc_rt
  [see goal caccGoalsDoc.g1
category cc acc cacc ]
use constants caccConstants
use constants caccConstants

caccPlan.verify
verification plan CACCPlan for caccreqs
activities
  speed_ActTest1 for cacc_rt
  [see goal caccGoalsDoc.g1
category cc acc cacc ]
use constants caccConstants
use constants caccConstants

ALISA Model
Resolute
Predicates

AADL Model
AGREE
  guarantees
  assertions

EMV2 mitigates

cameraLight
Assurance task cameraLight
filter caccSel only

Reports.assure
**caccProject.org**

- rs
- jdm

**caccGoals.goals**

- use constants caccConstants
- goal g1: "Safety"
  - stakeholder: rs

**caccreqs.reqspec**

- requirement speed_R1: for cacc_rt
  - [see goal caccGoalsDoc.g1]
  - category: cc acc cacc

**caccCat.cat**

- requirement categories:
  - cc acc cacc

**caccGlobals.constants**

- constants caccConstants
  - val MaximumSpeed = 120.0 mph

**caccPlan.verify**

- verification plan CACCPlan for caccreqs

**caccVerificationActivities.methodregistry**

- method: method1(): "first method" [manual test5]

**caccAssurance.alisa**

- Assurance task cameraLight
  - filter: caccSel only

**Reports.assure**

- AGREE
  - guarantees
  - assertions

- Resolute
  - Predicates

**AADL Model**

**ALISA Model**
use constants caccConstants

goal g1 : "Safety" [stakeholder rs]

requirement speed_R1 [see goal caccGoalsDoc.g1 category cc acc cacc]

verification plan CACCPlan for caccreqs

activities speed_ActTest1 for cacc_rt

caccVerificationActivities.methodregistry

method method1():"first method"[manual test5]
- val MaximumSpeed = 120.0 mph

- caccGoals.goals
  - use constants caccConstants
  - goal g1 : "Safety" [stakeholder rs]

- caccPlan.verify
  - verification plan CACCPlan for caccreqs

- caccAssurance.alisa
  - Assurance task cameraLight for cacc_rt.devices
    - filter caccSel only

- AADL Model
  - AGREE guarantees assertions
  - Resolute Predicates

- ALISA Model
  - EMV2 mitigates
  - method method1(): "first method" [manual test]
requirement categories

selection categories

use constants caccConstants

val MaximumSpeed = 120.0 mph

use constants caccConstants

goal g1 : "Safety" [stakeholder rs]

use constants caccConstants

requirement speed_R1 for cacc_rt [see goal caccGoalsDoc.g1 category cc acc cacc]

use constants caccConstants

verification plan CACCPlan for caccreqs

verification activities speed_ActTest1 for cacc_rt [category ccSel]
caccVActivities.method1()
caccVerificationActivities.methodregistry method method1 ()":"first method"[manual test5]

Assurance task cameraLight for cacc_rt.devices

AMV2** mitigates

AGREE guarantees assertions

Resolute

Predicates
Verifying using Libraries/Inline Properties

• We write Resolute queries on the model
• The model checker traverses the model and evaluates predicates
• Result is true/false evaluation
• Computations are done as side effects
Inline queries

annex Resolute{**
    prove (SystemWideReq1())
    prove (SystemWideReq2())
    prove (SystemWideReq3())
    prove (Security_Features(this))
    prove (Security_Connections(this))
    prove (Security_Data(this))
**};
Resolute Library

- Queries defined in a library and then invoked in a verification activity.
- More portable than the inline queries.

```plaintext
package caccResoluteLibrary
public
  annex Resolute{**
    Security_Features(self : component) <= **"Calculating total attack surface " **
    let featureSet : {feature} = features(self);
    let sumSurfaceC:real = sum({surfaceAreaC(t) for (t:featureSet)});
    let sumAccessC:real = sum({surfaceAccessC(w) for (w:featureSet)});
    Req5(sumSurfaceC/sumAccessC)
  }
```
Lessons Learned about Assets

• Choose a single domain – there is some learning curve for those unfamiliar with the domain
• Choose clearly scoped design decisions
• Be open to unexpected suggestions, don’t force your prejudice on the decomposition
• Be aware of the levels within the assets
• Associate assets/artifacts at the same level
Limitations

• From a pedagogical perspective
  – Time is the biggest limitation. If this is a single unit within a larger course, or if it is an overview industrial course, there may not be time to let students discuss a design in class.
  – The instructor may not have time to build multiple views of a single design.

• From an industrial perspective
  – Scope
Reusable pieces in the architecture

- Abstract specifications for standard architecture styles
- Concrete spec/implementations of specific products that will be integrated into a system
- Property sets that cover a domain of interest
- Abstract implementations of standard nominal to error behavior
- Prototypes for generic structures
Reusable auxiliary assets

Inside the architecture model
• Resolute code/property sets
• AGREE contracts

Outside of the architecture model
• Requirements in reqspec
• Verification assets in verify
  – Resolute
  – Java
• Assurance case outline in assure
Summary

• The AADL ecosystem contains a number of DSLs that assist in the development of assets.
• The ecosystem supports the Architecture Centric Virtual Integration Process (ACVIP)
• ACVIP uses those assets to more rapidly build systems that have fewer defects and to find those defects that do exist earlier
Any questions?

AADL models available at:  https://github.com/rose2s/SATURN_2016
APPENDIX – AADL OVERVIEW
Overview of AADL

• Main software structures
  – System
  – Process
  – Thread
  – Subprogram

• Main hardware entities
  – Device
  – Processor
  – Memory
  – Bus
Snippets from a system definition

abstract cacc
end cacc;

abstract implementation cacc.devices
subcomponents
radar_sensor: abstract sensor::generic_sensor.impl;
radar_handler: abstract sensor::generic_sensor_handler.impl;
connections
radar_handler_conn: port radar_sensor.sensor_data_out->radar_handler.sensor_data_in;
radar_conn: port radar_handler.sensor_data_out->vehicle_controller.sensor1_data_in;
flows
-- sensor's info to logger
etef0 : end to end flow radar_sensor.sensor_source -> radar_handler_conn ->
radar_handler.sensor_logger_path -> logger_radar_conn-> logger.logger_sink;
etef1 : end to end flow camera_sensor.sensor_source -> camera_handler_conn ->
camera_handler.sensor_logger_path-> logger_camera_conn -> logger.logger_sink;
Annexes

• “annex” is used to enhance standard without revising the standard
• Behavior annex – used to describe nominal behavior; state-machine based
• Error annex – used to describe error behavior also state-machine based
Behavior annex

system implementation acc.impl

annex behavior_specification{**

states
  steady: initial state;
  accelerating: state;
  braking: state;

transitions
  steady-[actualSpeed = targetSpeed]->steady;
  steady-[actualSpeed < targetSpeed]->accelerating;
  steady-[actualSpeed > targetSpeed]->braking;
  accelerating-[actualSpeed = targetSpeed]->steady;
  braking-[actualSpeed = targetSpeed]->steady;
  steady-[gap < gapLimit]->braking;
  accelerating-[gap < gapLimit]->braking;
**

end acc.impl;
Hierarchies

• Generalization hierarchy
  System sensor
  System speedSensor extends sensor

• Composition hierarchy
  system vehicle
    features
      speedometer: out data port speedSensor
Separation of specification and implementation

System sensor
  feature
    output: out data port aadlreal

System sensor.impl
  subcomponents
    physical_reading: speed_cable
Support for variation

• Abstract

• Refined to

• Prototype
Properties

• Properties are defined, given specific values, and used in analyses
• Properties are defined in a property set and then instantiated in structures

properties
  latency => 1 ms .. 3 ms applies to sensor_source;
  SEI::PowerBudget => 5.0W;

Go back
Advanced semantics/analyses

1. Sampling connection: takes the latest value
   » Problem: data consistency (lost or read twice)!

2. Immediate: receiver thread is immediately awakened, and will read data when emitter finishes

3. Delayed: actual transmission is delayed to the next time frame
Return

Link back
APPENDIX – A SECURITY METRIC
Attack surface of a product

- https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet
Attack Surface Metric

• Damage-Effort Ratio (DER)
• An attacker will choose the target that can cause the most damage for the least effort
• The access rights determine how hard it is to access the elements that will be compromised
ChannelConnectors

Connector Type ChannelT = {
    Property channelAccessRights : int;
    Property channelProtocol : int;
}

\[
\sum_{m \in M} DER_m(m)
\]

Larger protocol values indicate larger chunks of data that can be passed making it easier to move programs
Port Type EntryExitPointT = {
    Property entryExitPointPrivileges : int;
    Property entryExitPointAccessRights : int;
}

Privileges
------------
Access rights \sum_{c \in C} DER_c(c)

Level of privileges determines the damage that can be done
Component Type DataItemT = {
    Property dataItemType : int;
    Property dataItemAccessRights : int;
}

\[ \sum_{i \in I} DER_i(i) \]

The less restrictive the data types are the easier it is for attackers to enter.
Complete Attack Surface

\[
\sum_{m \in M} DER_m(m) \sum_{c \in C} DER_c(c) \sum_{i \in I} DER_i(i)
\]

A transform is evaluated to determine its effect on the attack surface

Would using a feature group reduce the port vulnerability?

Would using a record to group data fields together make an attack easier?
Property set

property set securityProperties is
--access rights are scored based on the clearance required
--no clearance required => 1.0
--secret clearance required => 3.0
--top secret clearance required => 5.0

Channel_Protocol : aadlreal applies to (connection);
Channel_AccessRights: aadlreal applies to (connection);

entryExitPointPrivileges: aadlreal applies to (port);
entryExitPointAccessRights:aadlreal applies to (port);

dataItemType:aadlreal applies to (data);
dataItemAccessRights:aadlreal applies to (data);
-- From SEI Tech Report CMU/SEI-2015-TR-01

-- properties to support documenting and analyzing security
-- Added property that supports access mode of data
AccessProtection: list of record
  (  
    AccessMode: enumeration(r, w, rw, x);
    AccessGroup: enumeration (CC, ABS);
  ) applies to (all);
end securityProperties;
Results of Claim functions

- ✔ print_aadl(cacc_rt_devices_Instance : caccIntegration::cacc_rt.devices)
  - ✔ cacc_rt_devices_Instance : caccIntegration::cacc_rt.devices
- ✔ SystemWideReq1()
- ✔ SystemWideReq2()
- ❌ SystemWideReq3()
- ✔ Security_Features(cacc_rt_devices_Instance : caccIntegration::cacc_rt.devices)
  - ✔ Calculating total attack surface
    - ✔ DERC is 1.6666666666666667
- ✔ Security_Connections(cacc_rt_devices_Instance : caccIntegration::cacc_rt.devices)
- ✔ Security_Data(cacc_rt_devices_Instance : caccIntegration::cacc_rt.devices)
  - ✔ Calculating total attack surface
    - ✔ DERI is 0
Ports

features
sensed_speed: in data port
data_types::speed{securityProperties::entryExitPointPrivileges=>5.0;securityProperties::entryExitPointAccessRights=>3.0;};
sensed_speed_limit: in data port
data_types::speed{securityProperties::entryExitPointPrivileges=>5.0;securityProperties::entryExitPointAccessRights=>3.0;};
DER$_C$

Security_Features(self : component) <= "Calculating total attack surface"
let featureSet : {feature} = features(self);
let sumSurfaceC: real = sum({surfaceAreaC(t) for (t: featureSet)});
let sumAccessC: real = sum({surfaceAccessC(w) for (w: featureSet)});
Req5(sumSurfaceC/sumAccessC)

Req5(total: real) <= "DERC is " total**true

surfaceAreaC(self: feature): real = property(self, securityProperties::entryExitPointPrivileges, 0.0)
surfaceAccessC(self: feature): real = property(self, securityProperties::entryExitPointAccessRights, 0.0)
Channel

connections
bus_radar: bus access this_bus<->
radar_sensor.BA{securityProperties::Channel_Protocol=>9.0;securityProperties::Channel_AccessRights=>9.0;};
bus_camera: bus access this_bus<-
camera_sensor.BA{securityProperties::Channel_Protocol=>9.0;securityProperties::Channel_AccessRights=>9.0;};
DERM

Security_Connections(self : component) <= **"Calculating total attack surface " **
let connectionSet : {connection} = connections(self);
let sumSurfaceM:real = sum({surfaceAreaM(u) for (u:connectionSet)});
let sumAccessM:real = sum({surfaceAccessM(v) for (v:connectionSet)});
Req6(sumSurfaceM/sumAccessM)

Req6(total:real)<= **"DERM is " total**true

surfaceAreaM(self:connection):real=
property(self,securityProperties::Channel_Protocol,0.0)

surfaceAccessM(self:connection):real=
property(self,securityProperties::Channel_AccessRights,0.0)
item: data simple{securityProperties::dataItemType=>7.0; securityProperties::dataItemAccessRights=>5.0;};
DERI

Security_Data(self : component) <= **"Calculating total attack surface " **
let dataSet : {component} = subcomponents(self);
let sumSurfaceI:real = sum({surfaceAreaI(u) for (u:dataSet)});
let sumAccessI:real = sum({surfaceAccessI(v) for (v:dataSet)});
Req7(sumSurfaceI/(sumAccessI))

Req7(total:real)<=**"DERI is " total**true

surfaceAreaI(self: data): real =
property(self,securityProperties::dataItemType,0.0)
surfaceAccessI(self: data): real =
property(self,securityProperties::dataItemAccessRights,0.0)
4 security patterns
Sanitize Data at Entry/Exit Points

- this transformation requires the architect to insert a component between an entry/exit point and the environment
- Ports that previously served as entry/exit points should be moved to the sanitizer
- have their privileges reduced by an order of magnitude to reflect the sanitizing function
Favor
Restricted Channels

Limiting the type of data transmitted over a channel can reduce the attack surface of the system by lessening the advantage gained by exploiting that channel.

The protocol value should be lowered to reflect the more restrictive nature of the new protocol.
Move Data Items to the Interior

• Moving data items to the interior of a system shifts untrusted data items away from the system’s perimeter.

• Data items that cannot be moved to the interior of the system should be evaluated to determine if they are necessary and be eliminated if they are not.
property set securityProperties is
Channel_Protocol : aadlinteger applies to (connection);
Channel_AccessRights: aadlinteger applies to (connection);
entryExitPointPrivileges: aadlinteger applies to (port);
entryExitPointAccessRights: aadlinteger applies to (port);
dataItemType: aadlinteger applies to (data);
dataItemAccessRights: aadlinteger applies to (data);
end securityProperties;