Model-Based Engineering with AADL: An Overview

Peter Feiler

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Outline

AADL: The Language

What's New in AADL V2

Modeling with AADL

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

AADL: Components and Connections

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Application Components

System: hierarchical organization of components

Process: protected address space

Thread group: organization of threads in processes

Thread: a schedulable unit of concurrent execution

Data: potentially sharable data & datqa typing

Subprogram: callable unit of sequential code

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Execution Platform Components

Processor – provides thread scheduling and execution services

Virtual processor – hierarchical schedulers & partitions

Memory – provides storage for data and source code

Bus – provides physical connectivity between hardware components

Virtual bus – virtual channels & protocols

Device – interface to external environment, physical components

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

System Type

System

```
system GPS
features
  speed_data: in data port metric_speed
     {SEI::BaseType => UInt16;};
  geo_db: requires data access real_time_geoDB;
  s_control_data: out data port state_control;
flows
  speed_control: flow path
       speed_data -> s_control_data;
properties SEI::redundancy => Dual;
end GPS;
```

{type}
extends
features
flows
properties

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

System Implementation

```
system implementation GPS.secure
subcomponents
  decoder: system PGP_decoder.basic;
  encoder: system PGP_encoder.basic;
  receiver: system GPS_receiver.basic;
connections
  c1: data port speed_data -> decoder.in;
  c2: data port decoder.out -> receiver.in;
  c3: data port receiver.out -> encoder.in;
  c4: data port encoder.out -> s_control_data;
flows
 speed_control: flow path speed_data -> c1 -> decoder.fs1
            -> c2 -> receiver.fs1 -> c3 -> decoder.fs1
            -> c4 -> s_control_data;
modes none;
properties arch::redundancy_scheme => Primary_Backup;
end GPS;
```

{implementation}
- extends
- refines type
- subcomponents
- calls
- connections
- flows
- modes
- properties

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Some Standard Properties

Thread

Dispatch_Protocol => Periodic;
Period => 100 ms;
Compute_Deadline => value (Period);
Compute_Execution_Time => 10 ms .. 20 ms;
Compute_Entrypoint => “speed_control”;
Source_Text => “waypoint.java”;
Source_Code_Size => 12 KB;

Code to be executed on dispatch

File containing the application code

Processor

Thread_Swap_Execution_Time => 5 us.. 10 us;
Clock_Jitter => 5 ps;

Bus

Allowed_Message_Size => 1 KB;
Propagation_Delay => 1ps .. 2ps;
Bus_Properties::Protocols => CSMA;

Protocols is a user defined property

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Latency Impact of Partitions

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Flow Sources, Paths, Sinks

```
device brake_pedal
features
   brake_status: out data port bool_type;
flows
   Flow1: flow source brake_status;
end brake_pedal;

system cruise_control
features
   brake_status: in data port;
   throttle_setting: out data port;
flows
   brake_flow_1: flow path brake_status -> throttle_setting;
end cruise_control;

device throttle_actuator
Features
    throttle_setting: in data port float_type;
flows
    Flow1: flow sink throttle_setting;
end throttle_actuator;
```


© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Architecture Execution Semantics Defined – Components to SoS

- Nominal & recovery
- Fault handling
- Resource locking
- Mode switching
- Initialization
- Finalization

Thread Example Diagram

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Component Interactions & Modes

Completely defined interfaces & interactions

- Port-based flows
  - State data, events, messages
  - Flow specifications & connections
  - End-to-end flows
- Synchronous call/return
- Shared access

Modal & dynamically configurable systems

- Modeling of operational modes
- Modeling of fault tolerant configurations
- Modeling of different levels of service

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

A Partitioned Run-Time Architecture

Strong Partitioning

- Timing Protection
- OS Call Restrictions
- Memory Protection

Interoperability/Portability

- Tailored Runtime Executive
- Standard RTOS API
- Application Components

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Domain Data Modeling

Domain types & base types

Data value range and units of measurement

Data transfer assumptions

- Guaranteed delivery
- Ordered delivery

Data stream characteristics

- Computational error
- Miss rates
- Freshness

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

AADL and Safety-Criticality

Fault management

- Architecture patterns in AADL
  - Redundancy, health monitoring, …
- Fault tolerant configurations & modes

Dependability

- Error Model Annex to AADL
- Specification of fault occurrence and fault propagation information
- Use for hazard and fault effect modeling
- Reliability & fault tree analysis

Behavior validation

- Behavior Annex to AADL
- Model checking
- Source code validation

```
package errormodels
public
  annex error_model {**
    -- simple error model
error model Basic
features
    Failed : error event;

    Error_Free: initial error state;
    Permanent_Failure: error state;

    Visible_Failure: in out error propagation;
end Basic;

error model implementation Basic.Nominal
transitions
     Error_Free -[Failed, in Visible_Failure]-> Permanent_Failure;
     Permanent_Failure -[out Visible_Failure]-> Permanent_Failure;
properties
     Occurrence => poisson 10E-4 applies to Failed;
     Occurrence => poisson 10E-6 applies to Visible_Failure;
end Basic.Nominal;
```

Consistency checking of safety-criticality levels

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Architecture Redundancy Pattern

External and internal mode control

Pattern variants with different latency & fault propagation characteristics

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Leverage Connectivity in AADL Models

Fault propagation at the application logic level, at the hardware level, and between the two levels.

- Provides compositional model specification approach
- Architecture defines propagation paths for software and hardware

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Large-Scale Development

Component evolution

- Component templates & refinement
- System families
- Component variants
- Components as extensions of other components
- Model configuration by property values

Large models & team development

- Components organized into AADL packages
- Public & private package sections
- Independently developed packages
- Version management of AADL packages
- Model integration

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

AADL Language Extensions

Model annotation through properties and sublanguages

New properties defined through property sets

Standard compliant sublanguage syntax in annex subclauses

Project-specific language extensions

Language extensions as approved SAE AADL standard annexes

Examples

- Error Model
- Interaction behavior
- System partitions (e.g., ARINC 653)

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

COTRE Annex Extension

```
THREAD t
  FEATURES
    sem1 : DATA ACCESS semaphore;
    sem2 : DATA ACCESS semaphore;
END t;

THREAD IMPLEMENTATION t.t1
  PROPERTIES
    Period => 13.96ms;
    cotre::Priority => 1;
    cotre::Phase => 0.0ms;
    Dispatch_Protocol => Periodic;
```

```
ANNEX cotre.behavior {**
   STATES
     s0, s1, s2, s3, s4, s5, s6, s7, s8 : STATE;
     s0 : INITIAL STATE;
   TRANSITIONS
     s0 -[ ]-> s1 { PERIODIC_WAIT };
     s1 -[ ]-> s2 { COMPUTATION(1.9ms, 1.9ms) };
     s2 -[ sem1.wait ! (-1.0ms) ]-> s3;
     s3 -[ ]-> s4 { COMPUTATION(0.1ms, 0.1ms) };
     s4 -[ sem2.wait ! (-1.0ms) ]-> s5;
     s5 -[ ]-> s6 { COMPUTATION(2.5ms, 2.5ms) };
     s6 -[ sem2.release ! ]-> s7;
     s7 -[ ]-> s8 { COMPUTATION(1.5ms, 1.5ms) };
     s8 -[ sem1.release !]-> s0;
  **};
END t.t1;
```

COTRE thread properties

COTRE behavioral annex

Courtesy of

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Outline

AADL: The Language

What's New in AADL V2

Modeling with AADL

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

What's New in AADL2

Packages and classifier visibility

New classifiers

Classifier matching and substitution

Explicit parameterization of classifiers (prototypes)

Abstract features

Connection improvements

Feature groups

Arrays of components

Property improvements

Threads and communication timing

Subprograms and subprogram groups

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Packages and Classifier Visibility

Component implementation in public and private section of packages

- Public: represents component variant: only properties
- Private: represents realization

With statement

- Specifies the set of packages whose classifiers can be referenced

Renames statement

- Represents a local alias for long qualified classifier references to component types
- Renames provides short names for fully qualified package names

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Abstract Components

Example: A conceptual architecture

```
abstract car
  end car;
abstract implementation car.generic
      subcomponents
        power_train: abstract PowerTrain;
        exhaust_sys: abstract ExhaustSys;
    end car.generic;

abstract PowerTrain
  end PowerTrain;

  system carRT extends car
  end car;
system implementation carRT.impl extends car.generi
      subcomponents
        power_train: refined to process PowerTrain;
        exhaust_sys: refined to process ExhaustSys;
```

Concrete category extends abstract type/implementation

Concrete category supplied by refinement

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

New Classifiers

Subprogram group classifier

- Represents subprogram library
- Provides subprogram features
- Provides and requires access to subprogram group

Virtual processor classifier

- Represents hierarchical scheduler, virtual machine, processor partition, rate group task
- Virtual processor as subcomponent
- Virtual processor is bound to virtual processors and processors

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

New Classifiers - 2

Virtual bus classifier

- Represents logical/virtual channel, protocols and protocol stacks
- `Required_Virtual_Bus_Class` and `Provided_Virtual_Bus_Class` property associated with buses, processors and virtual buses
- `Required/Provided_Connection_Quality_Of_Service` property
  - Example value: guaranteed_delivery
- Virtual bus subcomponents
- No requires virtual bus access and provide virtual bus access

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Refinement Substitution Rules

`Classifier_Match`

1. Goto type
2. Select an implementation

`Type_Extension`

1. Goto type
2. Select an extension
3. Select an implementation (optional)

`Signature_Match`

1. Goto type
2. Select a type with a superset of features and flow specifications
3. Select an implementation (optional)

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Prototypes – Consistent Refinement

Example: Type of data on a port

```
system GpsGeneric
  prototypes
    dt: data;
  features
    pos_1: out data dt;
    pos_2: out data dt;
end GpsGeneric;
```

```
system Gps
  extends GpsGeneric(dt=>PosData)
end Gps;
```

Compare to refinement

```
system GpsBasic
  features
    pos_1: out data;
    pos_2: out data;
end GpsBasic;
```

```
system GpsRef extends GpsBasic
  features
    pos_1: refined to out data PosData;
    pos_2: refined to out data OtherData;
end GpsRef;
```

No enforcement of consistency possible

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Relaxation of Port Connection Rules

Connection syntax

- Only **port** keyword

Connections between data port, event data ports, event ports

- event port -> event port
- data port -> data port, event data port, event port
- event data port -> event data port, data port, event port

Connections between ports and data components

- data, data access -> data port, event data port, event port
- data port, event data port -> data, data access

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Relaxation of Port Connection Rules

Connection syntax

- Only **port** keyword

Connections between data port, event data ports, event ports

- event port -> event port
- data port -> data port, event data port, event port
- event data port -> event data port, data port, event port

Connections between ports and data components

- data, data access -> data port, event data port, event port
- data port, event data port -> data, data access

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Feature Group Improvements

For any feature

- Not just ports

Inverse of in feature declaration

- No need to define both a feature group type and an inverse feature group type

Matching of independently declared feature group type

- Ability to define *equivalence*

Prototypes and refinement

- Classifier and feature prototypes
- Substitution rules

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Arrays in AADL

Arrays of subcomponents

Multiple array dimensions

- Of same subcomponent
- Of subcomponent and ancestor component

Single array dimension for features

- Example applications: routers, voters

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Arrays in AADL – Example

Fan-in, fan-out at process port

Thread connectivity across processes

T1, t2, t3, t4 : thread controller [4];

Connection_Pattern => (one_to_one);

t1 -> t3 arrays ; t1 -> t4 arrays; t2 -> t3 arrays; t2 -> t4 arrays with pairwise connected elements

Feature group in process

- t1 -> t3 array; t2 -> t4 array

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Connection patterns – one dimension

Identity aka. One_To_One

next

Cyclic_Next

previous

Neighbor = (next, previous)

Cyclic Neighbor = (CyclicNext, CyclicPrevious)

next, one_to_one

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Connection patterns – two dimensions

S[3,3]; D[3,3];
Port S.p1 -> D.p2;

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Connection patterns – two dimensions self

M[3,3]; port M.p1 -> M.p2;

(identity , next)

(next, identity)

(next, next)

(next, next),
(next, previous)

(identity, previous) ,
(identity, next)

(identity, identity)

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Mode Improvements

Modes in component types

- Modes common to all implementations
  - No additional modes in implementation
- Mode-specific property values on component type, flow spec, and features

Named mode transitions

Improved specification of mode transition activities

- Based on Rolland TLA specification

Inherited modes

- **Requires modes** to indicate which modes on an enclosing component the component is sensitive to
- Mapping of mode names

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Flow Improvements

End-to-end flows composed of end-to-end flows

- Ete12 : **end to end flow** ete1 -> conn12 -> ete2
- Ete13 : **end to end flow** ete1 -> conn13 -> ete3

Flow through data components

Did you know
Multiple end-to-end flows from feature group connections and connections involving arrays

End-to-end: T1.fs1 -> c1 (write) -> d -> c2 (read) -> T2.fs2

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

System Modeling

Layered architectures

- Use of system implementation declarations
- `Implemented_As` property

Asynchronous systems

- Synchronization domains
- `Reference_Time` property

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

What's New in AADL2

Packages and classifier visibility

New classifiers

Classifier matching and substitution

Explicit parameterization of classifiers (prototypes)

Abstract features

Connection improvements

Feature groups

Arrays of components

Property improvements

Threads and communication timing

Subprograms and subprogram groups

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Outline

AADL: The Language

What's New in AADL V2

Modeling with AADL

- **What is your system?**
- Modeling multiple tiers
- Finding the right abstraction
- Filling in the software architecture
- Team support & Variants
- Modeling a reference architecture

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Towards Architecture Centric Engineering

Build on architecture tradeoff analysis (e.g., SEI ATAM)

- Provides focused evaluation method
- MBE/AADL provides quantitative analysis & starter models to build on

Facilitate pattern-based technical architecture root cause analysis

- Identify systemic risks in technology migration and refresh
- AADL provides semantic framework to reason about technical problem areas and potential mitigation strategies

Scalability through architecture extraction

- Leverage existing design data bases
- Challenge: abstract away from design details
- Focus on “what” instead of “how”

Support system and software assurance

- Provides structured approach to safety/dependability assurance
- MBE/AADL provides evidence based on validated models

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Software & System Engineering

AADL – SysML coordination

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Modeling an Embedded System Architecture

Elements of an embedded system architecture

- Software Architecture PLUS
- Hardware Architecture PLUS
- Physical system/environment PLUS
- Logical interface between software and physical system PLUS
- Physical interface between hardware and physical system PLUS
- Deployment of software on hardware

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Outline

What is your system?

Modeling multiple Tiers

Finding the right abstraction

Filling in the software architecture

Team support & Variants

Modeling a reference architecture

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Multiple Tiers of a System

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Abstraction in System Architecture

Bus to represent resource such as hydraulics

Bus to represent resource connections such as hydraulic lines

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Embedded Software Architecture

Computing Platform

Application Software

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Outline

What is your system?

Modeling multiple Tiers

Finding the right abstraction

Filling in the software architecture

Team support & Variants

Modeling a reference architecture

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Hardware as an Abstraction

Memory
- Individual memory modules
- Processor with memory

High-speed Ethernet switch
- As bus with latency properties
- As collection of hardware components with processing queues

Multi-core processors
- As single AADL processor
- Individual cores as processors

Digital camera
- As AADL device
- *Implemented_as* system with hardware and software

Challenge: abstract away from design details
Focus on “what” instead of “how”

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Intended Data Flow in Task Architecture

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Modeling the What

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Types Vs. Properties

Example: Power supply

Type enforcement by AADL semantics

Multiple properties mapped into types

```
bus LAN
  features
    Power: requires bus access PowerSupply.V9_0 {
      SEI::PowerBudget => access 0.1 W;
      };
  properties
    SEI::NetWeight => 2.5 kg;
end LAN;

bus PowerSupply
  properties
    SEI::PowerCapacity => 10.0 W;
    SEI::NetWeight => 25.0 kg;
end PowerSupply;

bus implementation PowerSupply.V9_0
end PowerSupply.V9_0;

bus implementation PowerSupply.V4_5
end PowerSupply.V4_5;
```


© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Flow Use Scenario through Subsystem Architecture

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Outline

What is your system?

Modeling multiple Tiers

Finding the right abstraction

Filling in the software architecture

Team support & Variants

Modeling a reference architecture

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Systems Context Diagrams

Defines all of the entities that are within the scope of an application

Shows the data flows that are included in the scope of a application
.

Focuses on relationships with external entities and identifies the information that is exchanged between these external entities and the system under review.

Context Diagram at the project level is the root of the Functional Decomposition at the conceptual level of detail. (Level 0)

Subsequent decompositions (1 to n) identify components and the logical flow of data between the components

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

A Typical Context Diagram

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Cruise Control
Functional Decomposition-Level 0

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Feature Groups

Feature Groups are collections of individual ports and port groups such that

- a feature group can be connected to individually
- a component feature group can be connected as a single unit

Bundling of connections reduces graphical clutter

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Compute Desired Speed
Functional Decomposition-Level 2

Functional specification in domain notation

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Outline

What is your system?

Modeling multiple Tiers

Finding the right abstraction

Filling in the software architecture

Team support & Variants

Modeling a reference architecture

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Use of AADL Packages

AADL Packages as name spaces

- Qualify references by package name: BaseTypes::uint16
- Nested package names: edu::cmu::sei::MySystem::App1

Component libraries

- Component types and implementations
- Hardware & application libraries
- Subsystem details in separate packages

Data dictionary

- Data types
- Domain information on data types

Interaction specifications

- Port group type specifications

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Model Repository

Packages as versioned units
Eclipse projects as Configuration Items

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Modeling of System Configurations

```
package SystemConfigurations
public
  system mysystem
    properties
      SEI::WeightLimit => 11.0 kg;
  end mysystem;

  --        a system with the application as a parts list of subsystems with resource budgets
  --        and a platform with hardware parts with resource capacities
  system implementation mysystem.parts
    subcomponents
      Platform: system HardwarePlatform::ComputingPlatform.ThreeProcessorParts;
      ApplicationSystem: system AppSystems::EmbeddedApp.SubSystemParts;
  end mysystem.parts;

  --        the same system of parts with subsystems allocated to processors
  --        This may be an initial allocation or an allocation specified as part of requirements
  --        Given such an allocation we can analyze whether the                                    ssor
  system implementation mysystem.allocatedparts
    extends mysystem.parts
    properties
      Actual_Processor_Binding =>  reference platform.missionprocessor1 applies to ApplicationSystem.DM;
      Actual_Processor_Binding =>  reference platform.missionprocessor1 applies to ApplicationSystem.PCM;
      Actual_Processor_Binding =>  reference platform.missionprocessor2 applies to ApplicationSystem.FM;
      Actual_Processor_Binding =>  reference platform.missionprocessor2 applies to ApplicationSystem.FD;
      Actual_Processor_Binding =>  reference platform.missionprocessor3 applies to ApplicationSystem.WAM;
      Actual_Memory_Binding =>  reference platform.missionprocessor2.membank1 applies to ApplicationSystem.FD;
      Actual_Memory_Binding =>  reference platform.missionprocessor2.membank1 applies to ApplicationSystem.FM;
      Actual_Memory_Binding =>  reference platform.missionprocessor1.membank1 applies to ApplicationSystem.PCM;
      Actual_Memory_Binding =>  reference platform.missionprocessor1.membank1 applies to ApplicationSystem.DM;
  end mysystem.allocatedparts;
```

Use of extends to specify configurations

Use of contained property association to keep deployment information in one place

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Variants in System Families or Product Lines

AADL V2 supports multiple data sets

Multiple interface variants

- AADL component types with extends

Multiple realizations

- Multiple AADL component implementations per type

Variation in component structure and communication

- Parameterized component implementations (AADL V2 Prototype)
- Dynamic variation through mode-specific subcomponents and connections

Source code variations

- Different source files as Source_Text property
- Conditional compilation flags as properties or property constants

Seed & calibration values

- As property values on data components

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Outline

What is your system?

Modeling multiple Tiers

Finding the right abstraction

Filling in the software architecture

Team support & Variants

Modeling a reference architecture

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

The Mission Data System

A reference architecture

- To be instantiated for different applications

An embedded systems architecture

- Consists of physical system, computing hardware, application software

A control systems architecture

- Feedback loops in application architecture
- Feedback loops in data management system

A multi-layered architecture

- From low-level control loops to goal-oriented planning and plan execution

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

A Reference Architecture

Generic Architecture Pattern With Connection Topology

Instantiation of Application Architecture Computing Platform, and Physical System

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Mission and Control Processing

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

A Layered Application System

Excerpt from the Textual Specification:

process implementation MDSControlSystem.basic
  subcomponents
    GoalPlanner: **thread group** ControlSoftware::GoalPlanner;
    GoalExecutive: **thread group** ControlSoftware::GoalExecutive;
    GoalMonitor: **thread group** ControlSoftware::XGoalMonitor;
    StateEstimation: **thread group** ControlSoftware::estimator;
    StateControl: **thread group** ControlSoftware::controller;
    OperatorConsole: **thread group**
ControlSoftware::OperatorConsole;

Textual & Graphical Representations

Focus on Information Flow

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Mission Data System (MDS) Architecture*

* M. Bennett, R. Knight, R. Rasmussen, M. Ingham, "State-Based Models for Planning and Execution, 2006-08-11.

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐

Separation of Concerns: Data Management System

Role of state history and state variables

- Information flow through application architecture
  - Access to multiple data stream values
- Historical log of data stream for post mortem analysis
  - Storage management through compression
- Distributed processing between space and ground system
  - Proxies & telemetry

Managed data history variables

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Flow-Oriented Model of Temperature Control

© 2009 Carnegie Mellon University

资料整理
火龙果软件
荐

Reference Architecture Instantiation

```
package ControlSoftware::Camera
public
  thread group estimator
    extends ControlSoftware::estimator
    features
      StateEstimates: refined to port group MDSData::Camera::StateVariables;
      measurements: refined to port group MDSData::Camera::RequiredMeasurements;
      PreviousCommands: refined to port group MDSData::Camera::RequestedCommands;
    flows
      StateFlow: flow path measurements -> StateEstimates;
  end estimator;

  thread group implementation estimator.camera
    subcomponents
      TemperatureEstimator: thread TemperatureEstimator;
      TemperatureSensorHealthEstimator: thread TemperatureSensorHealthEstimator;
      HeaterSwitchEstimator: thread HeaterSwitchEstimator;
    connections
```

Instantiation of reference architecture through refinement of AADL model

Deployment on different computing hardware platforms

© 2009 Carnegie Mellon University

资料整理 火龙果软件 荐