

A Component Model and Software Architecture for CPS

Abhishek Dubey, <u>Gabor Karsai</u>, Nagabhushan Mahadevan ISIS/Vanderbilt University

NASA Cooperative Agreement NNX08AY49A

VANDERBILT UNIVERSITY

Outline

- Software components for real-time systems
- ARINC-653 features
- CCM features
- The ARINC Component Model
 - Components and interactions
 - Modeling and generation
 - Application: Software Health Management
 - Implementation
 - An Example
- Lessons Learned / Summary

Notional Design Flow for High-Confidence Software Systems







• Need:

• A Component Model suitable for hard real-time systems that codifies all component interactions and allows specification of timing requirements

Real-time CORBA?

- QoS and scheduling attributes on CCM
- MARTE UML Profile?
 - Specifications for timing properties in UML models

AUTOSAR?

• Component execution model? (Only recently added).

ARINC-653/APEX: Partitioning Kernel API



- Partitions:
 - Spatial and Temporal separation of activities Fault isolation!
 - Partition memory size and temporal duration are fixed
- Within a partition (shared address space)
 - Multiple processes (static); periodic/aperiodic, with opt deadline
 - Primitives for process interactions: buffers and blackboards, semaphores and events
 - Health monitor (to restart processes)
- Across partitions (isolated address spaces)
 - Fixed allotment of CPU time
 - Message-based interactions via channels connecting sampling and queuing ports
- Multiple processors ('modules') few details standardized



CORBA Component Model

Components

- Generalized 'objects' with state
- Synchronous (call/return) interactions via provided/required interfaces
- Asynchronous (publish/subscribe) interactions via publish/subscribe interfaces

Component homes

Lifecycle and resource management for components





- Provide a CCM-like layer on top of ARINC-653 abstractions
- Notional model:



- Terminology:
 - Synchronous: call/return
 - Asynchronous: publish-return/trigger-process
 - Periodic: time-triggered
 - Aperiodic: event-triggered

ACM: The ARINC Component Model



- Each 'input interface' has its own process
 - Process must obtain read-write/lock on component
- Asynchronous publisher (subscriber) interface:
 - Listener (publisher) process
 - Pushes (receives) one event (a struct), with a validity flag
 - Can be event-triggered or time-triggered (i.e. 4 variations)
- Synchronous provided (required) interface:
 - Handles incoming synchronous RMI calls
 - Forwards outgoing synchronous RMI calls
- Other interfaces:
 - State: to observe component state variables
 - Resource: to monitor resource usage
 - Trigger: to monitor execution timing



ACM: The ARINC Component Model



A component assembly



Components interact via asynchronous/event-triggered and synchronous/call-driven connections.

Example: The *Trigger* component is released periodically and it <u>publishes</u> an event upon each activation. The *GPS* component <u>subscribes</u> to this event and is triggered sporadically to obtain GPS data from the receiver, and when ready it publishes its own output event. The *Display* component is triggered sporadically via this event and it uses a <u>required</u> interface to retrieve the position data from the *GPS* component.

ACM: The ARINC Component Model



Mapping the CCM concepts to APEX in ACM

ACM: APEX Component Model				APEX	APEX Concept Used	
Component method		Periodic Sporadic		Periodic process	Process start, stop Semaphores	
				Aperiodic process		
Invocation	Synchronous Call-Return	Periodic Target	Co-located	Ν/Α		
			Non-co-located	N/A		
		Sporadic Target	Co-located	Caller method signals callee to release then waits for callee until completion.	Event, Blackboard	
			Non-co-located	Caller method sends RMI (via CM) to release callee then waits for RMI to complete.	TCP/IP, Semaphore, Event	
	Asynchronous Publish-Subscribe	Periodic Target	Co-located	Callee is periodically triggered and polls	Blackboard	
			Non-co-located	'event buffer' – validity flag indicates whether data is stale or fresh	Sampling port, Channel	
		Sporadic Target	Co-located	Callee is released when event is available	Blackboard, Semaphore, Event	
			Non-co-located	Caller notifies via TCP/IP, callee is released upon receipt	Queuing port, Semaphore, Event	

Observe:

- All component interactions are realized via the framework
- Process (method) execution time has deadline, which is monitored

Modeling Language

- Modeling elements:
 - Data types: primitive, structs, vectors
 - Interfaces: methods with arguments
 - Components:
 - Publish/Subscribe ports (with data type)
 - Provided/Required interfaces (with i/f type)

Name GPSAssemble

Component

Sensor

GPSDataSource GPS NavDisplay Sensor GPSAssembly X

gps 1

Component

- Health Manager
- Assemblies
- Deployment
 - Modules, Partitions
 - Component \rightarrow Partition



Aspect SystemView V Base: N/A

Component

dat and ups

NavDispla

· GME Browser

🕀 🎯 Assemblies

200

G C GPSAssembly

C. GPS C. NavDisplay

C. Sensor



VANDERBILT UNIVERSIT



VANDERBILT UNIVERSITY

Modeling

- Needs for analysis: component internals + assembly
 - Component internal data- and control flows





Project on Model-based Software Health Management

- How to build 'software health management functions' into systems that monitor, diagnose, and mitigate software defects at run-time?
- Concept
 - Use model-based fault diagnostics techniques for monitoring and diagnosis
 - Use model-based software development techniques to design, analyze, and generate the code for the software health management function



Modeling Language: Monitoring

- Monitoring on component interfaces
 - Subscriber port \rightarrow 'Subscriber process' and Publisher port \rightarrow 'Publisher process'
 - Monitor: pre-conditions and post-conditions
 - On subscriber: Data validity ('age' of data)
 - Deadline (hard / soft)
 - Provided interface → 'Provider methods' and Required interface → 'Required methods'
 - Monitor: pre-conditions and post-conditions
 - Deadline (hard / soft)
 - Can be specified on a per-component basis
- Monitoring language:
 - Simple, named expressions over input (output) parameters, component state, delta(var), and rate(var,dt). The expression yields a Boolean condition.

T Name: N	VavDispla	ay.	Com	ponent	A
	-				
	data	in			
	0010				
bject Inspe	ctor				
data in					
Attributes.	Destaura			1	
DeadlineTu	Fieleler		ropentes		
PreConditio	pe n	nanu			
I recordido					
Period in Se	econds	-1			
PostConditi	on	cond1	= (DELT/	A[data_in	.time) > 0)
Deadline in	Seconds	8			
Deadline in Data Validit	Seconds yPeriod in	8			
Deadline in Data Validit Queue Size	Seconds yPeriod in	8 4 1			
Deadline in Data Validit Queue Size	Seconds yPeriod in	8 4 1			
Deadline in Data Validit Queue Size	Seconds yPeriod in	8 4 1			
Deadline in Data Validit Queue Size Queue Size	Seconds yPeriod in e	8 4 1			

Modeling Language: Component Health Manager



Reactive State Machine

- Event trigger:
 - Predefined condition (e.g. deadline violation, data validity validation)
 - User-defined condition (e.g. pre-condition violation)
- Reaction: mitigation *action* (start, reset, refuse, ignore, etc.)
- State: current state of the machine
- (Event X State) \rightarrow Action

SystemView	T Name: chm	Health Manger	Aspect	Sys	Aggregate	Inheritance Meta	
ErrorCond Response	e1 e2	et / REFUSE	al			bot Folder 2Partition 3Partition Assemblies GPS Assembly Components GPS NavDisplay Chim Chim Chim Components Compo	
State	Object Inspector		-	×		-C data_in	
ß	e1 Attributes Preferences	s Properties	for Kir	rd		gps_data_source grs_data_source GPSDataS GrsDataS getGPS grs_data_source grs_	ource Data
StateRef	ErrorTypes Condition (Specify if API Method Name (Specify	APPLICATION_ERRO PLIC: cond1 if not)R			Sensor Interfaces Types	-uraid

A Prototype Implementation

- ARINC-653 Emulator
 - Emulates APEX services using Linux API-s
 - ▶ Partition \rightarrow Process, Process \rightarrow Thread
 - Module manager: schedules partition set
 - Partition level scheduler: schedules threads within partition

- CORBA foundation
 MICO CCM ORB
 - No modifications
- CLHM: Component-level Health Manager







ACM:

A Prototype Implementation

Platform:

- ARINC-653 Emulator on Linux
- MICO (open source CORBA)
- Module manager, infrastructure
- Code generator
 - Produces 'glue code' for the component framework
 - Compiles monitoring expressions
 - Builds code for CHM

Designer supplies functional code





ACM: Model-based Development



• Graphical models are used the generate 'infrastructure' code

<pre>interface GPSDataSource { void getGPSData(out GPSData gpsData); }; component GPS { consumes SensorOutput data_in; provides GPSDataSource gps_data_src; publishes SensorOutput data_out; }; component NavDisplay { consumes SensorOutput data_in; uses GPSDataSource gps_data_src; }; IDL specification</pre>	Ivoid GPS_impl::APEX_getGPSData() Framework provided GPS_impl::APEX_GPS_impl->readlock(); GPS_impl::APEX_GPS_impl->getGPSData_impl(*GPS_impl::APEX_GPS_impl->m_getGPSData_gpsDataPtr); GPS_impl::APEX_GPS_impl->unlock(); Intercepting an ORB void GPS_impl::getGPSData_gpsData(GPSAssembly::GPSData&gpsData&gpsData) call and releasing regetGPSData_gpsData(GPSAssembly::GPSData&gpsData) call and releasing m_getGPSData_gpsDataPtr = &gpsData process PROCESS_ID_TYPE GPS_GET_DATA_PROCESS_ID; process PROCESS_ID_TYPE GPS_GET_DATA_PROCESS_NAME ="GPS_GET_DATA"; GET_PROCESS_ID(GPS_GET_DATA_PROCESS_NAME, &GPS_GET_DATA_PROCESS_ID, &RETURN_CODE; dormant state GET_PROCESS_ID(GPS_GET_DATA_PROCESS_ID, &RETURN_CODE); START(GPS_GET_DATA_PROCESS_ID,&RETURN_CODE); APEX_HELPER_WAIT_EVENT (GPS_GET_DATA_PROCESS_NAME, INFINITE_TIME_VALUE,&RETURN_CODE); APEX_HELPER_WAIT_EVENT (GPS_GET_DATA_PROCESS_NAME, INFINITE_TIME_VALUE,&RETURN_CODE);
<pre>void GPS_impl::getGPSData (GPSAssembly::GPSData&gpsData) { MICOMT::AutoLock l(m_mutex); gpsData.x = m_xPos; gpsData.z = m_zPos; } gpsData.z = m_zPos; } </pre>	Image: Provided p



ANDERBILT UNIVERSI

Fault	Detected at	Fault source	Mitigation
Hard deadline violation	GPS Trigger interface	GPS Component	Stop and restart
Stale data (missing update)	NAVDisplay Subscribe port	GPS Component	Use previous value
Missing sensor event	GPS Subscribe port	Sensor Component	Use previous value
Rate of change is too high	NAVDisplay required interface	GPS Component	Use previous value



Lessons Learned / Summary

- Two worlds: The highly dynamic CCM and the strictly static ARINC do not mesh well
- Allocating a thread to every method is possibly a waste of resources
- For analyzability a deeper modeling of component structure and behavior is needed
- ACM: Steps towards a hard real-time component model
 - CCM: provides the essential component abstraction
 - ARINC: provides the API / platform
- Model-based configuration and code generation helps
- ACM is an experiment work in progress