Adaptive Middleware for Real-Time Software

Louise Avila

CIS 700-02

November 2, 2005

Topic

- “An Adaptive Middleware for Context-Sensitive Communications for Real-Time Applications in Ubiquitous Computing Environments.” *Real-Time Systems Journal.* January 2004.
- Stephen S. Yau and Fariaz Karim
- Reconfigurable Context-Sensitive Middleware Research Project, Arizona State University

Ubiquitous Computing

- Computing experience is everywhere but enabling technologies are invisible
- Makes the user the center of computing
- Dynamically adapt to user’s needs and actions

Mobile Ad Hoc Networks (MANET)

- Collection of connected autonomous mobile nodes such as wearable, handheld and other mobile devices
- Free to move arbitrarily
- Bandwidth and energy constraints
- Dynamic network topologies
  - □ No dedicated network connectivity devices
  - □ Nodes form short-range wireless networks

Their Goal

- Make MANET context-sensitive
- Use data about environment and available resources
- Adapt behavior and interactions
- Schedule and execute time critical tasks
- Context sensitive interactions between applications

Context-Sensitive Services

- Detects, establishes and terminates communication channels
  - New devices enter the environment
  - Existing devices move away
- Efficient
- Address heterogeneity of devices
- One potential solution: middleware

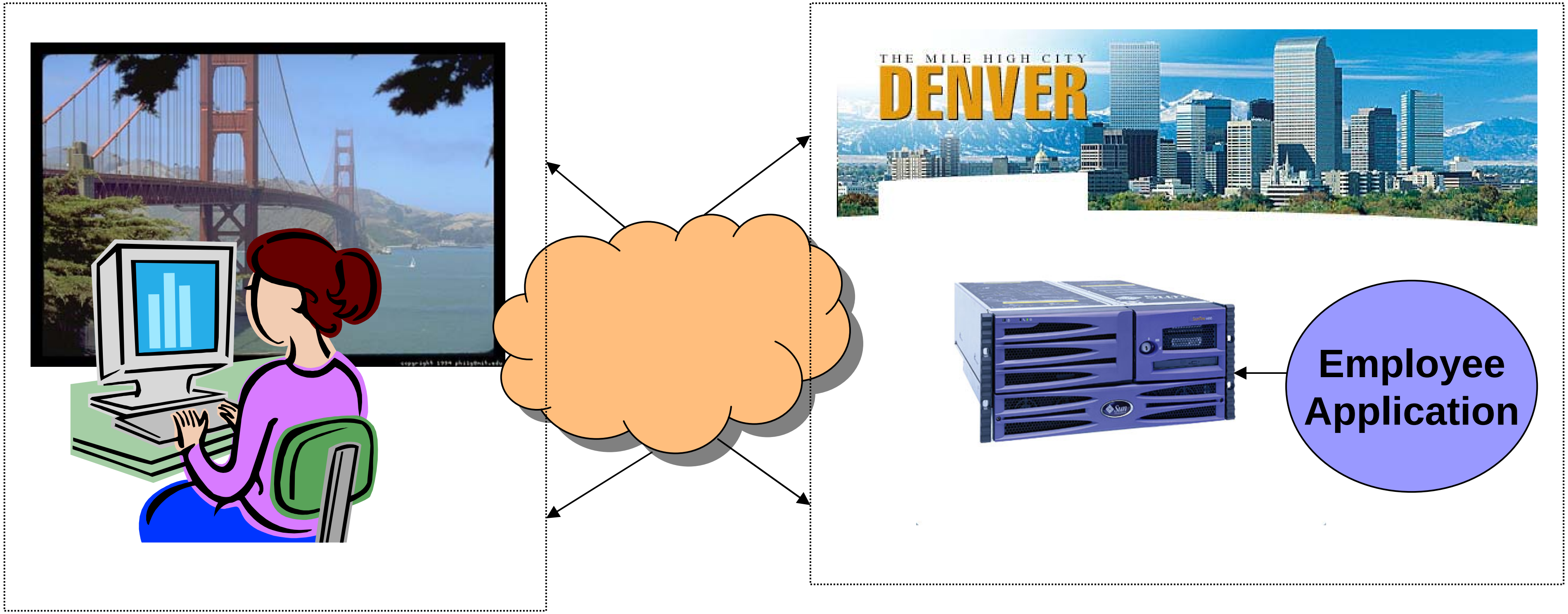
Middleware: Definition

- Software is distributed and developed using different languages, operating systems and hardware platforms
- Middleware "glues together" or mediates between two separate programs or software packages

CORBA

- Common Object Request Broker Architecture
- Creating and managing distributed objects in a network
- Industry standard developed by the Object Management Group
- http://www.omg.org/

CORBA Example



HR Rep works in San Francisco

HR Application runs on a server in Denver

CORBA Example

- Client programs don’t need to know:
  - Location of server program
  - Implementation of server
- Platform Independent
- Language Independent

Interface Definition Language

- Employee server class:

```
public class Employee {
    public String getEmployeeId(String name) {
        return eid;
    }
}
```

- Define interface for Employee class:

```
interface IEmployee {
    String getEmployeeId(in String name);
}
```

CORBA Example

- Compile interface with IDL compiler
- Client Stub
  - Proxy for the server that runs on the client
  - Converts method calls into messages
  - Client acts as though invoking on local object instance
- Server Skeleton
  - Converts messages back to method calls

CORBA Architecture

- Object Request Broker (ORB)
- Locates and activates object
- Delivers request
- Returns response
- Other services
  - Naming, Lifecycle, etc.

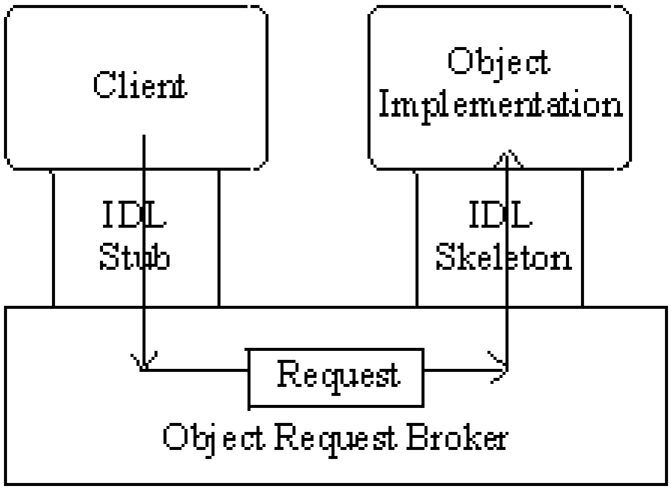


Figure 1: A request passing from client to object implementation

Copyright © 2000 Object Management Group

Middleware Benefits

- Reduce effort required to develop software
- Provide runtime services for applications
- Forces a separation between interface and implementation
- ORB approach
  - Isolate transport protocols from applications

Middleware: Limitations

- Existing middleware for enterprise and mobile networks:
  - Industry standards: CORBA, COM, EJB
  - Specialized “laboratory” versions: TAO
- Assume stable network
- Use client-server interaction semantics
- Do not use different contexts
- Laboratory versions have unique architectures – problem of interoperability

Challenges

- Systematic way to represent specific contexts and context awareness
- Timely context data collection, analysis and propagation
  - Transparent
  - Device and application-specific

Challenges

- Associating context with real-time actions
- Support for spontaneous and ad hoc context-sensitive communication

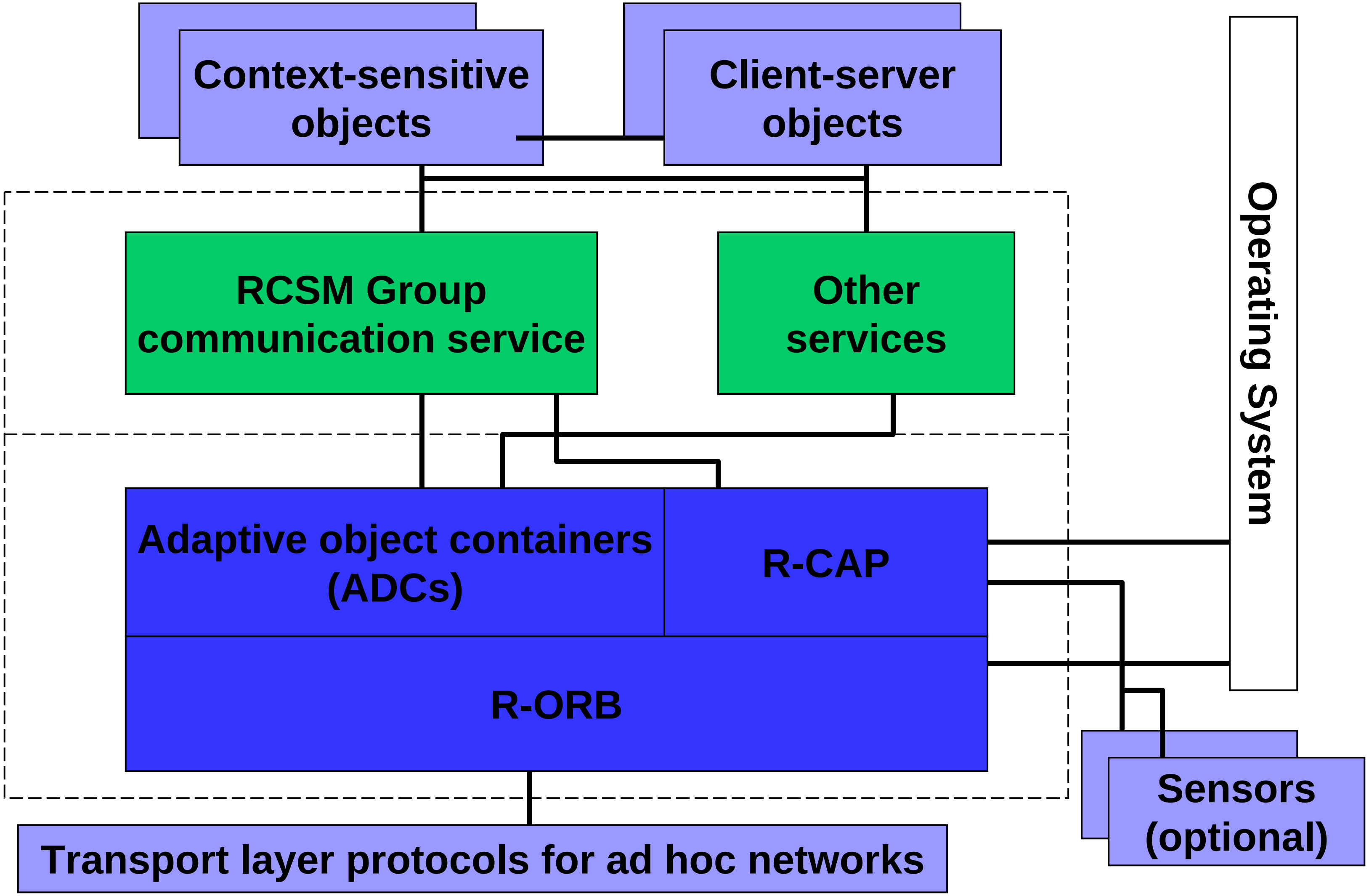
Reconfigurable Context-Sensitive Middleware (RCSM)

- Compliant with CORBA/OMA
  - User- level application software as application objects
- Object Request Broker (R-ORB)
  - Enables application objects implemented in different languages to communicate in a distributed, heterogeneous environment
  - Provides context- sensitive communication
- R-CAP performs low-level context monitoring and acquisition

RCSM Features

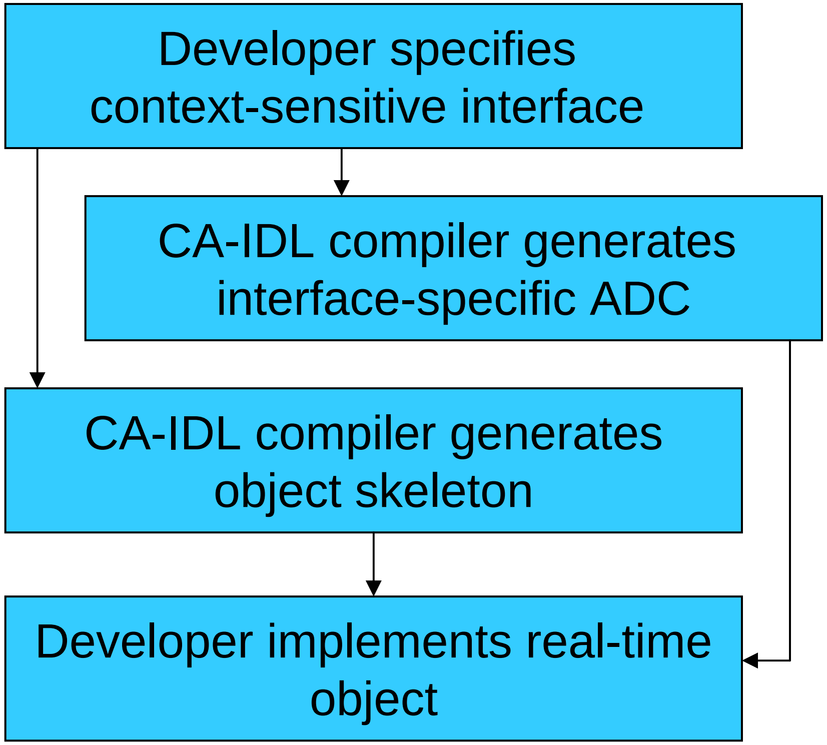
- Context-aware interface definition language (CA-IDL)
  - □ Based on IDL
  - □ Separates interfaces from implementations
- Adaptive Object Containers (ADC)
  - □ Interface specific context analyzer components.
  - □ Communicate at runtime with other components to acquire context data
  - □ Communicates with the object implementation to invoke different methods when suitable contexts are detected.

RCSM Component Hierarchy


Context-sensitive objects
Client-server objects
Operating System
RCSM Group communication service
Other services
Adaptive object containers (ADCs)
R-CAP
R-ORB
Sensors (optional)
Transport layer protocols for ad hoc networks

Development and Runtime Support

Development Support



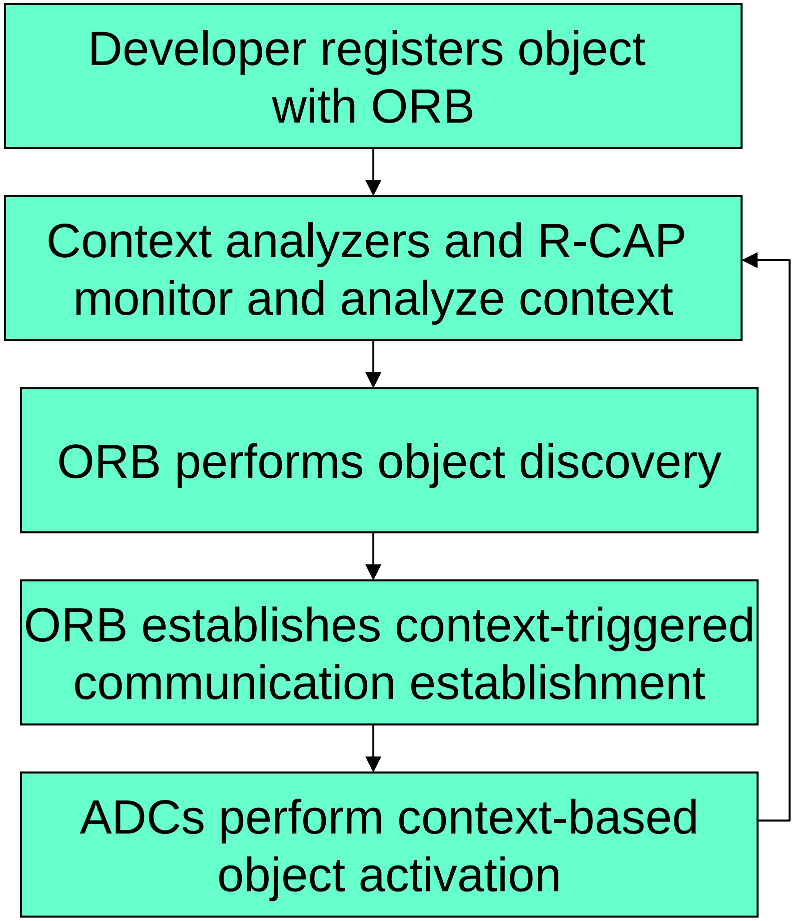
Developer specifies context-sensitive interface

CA-IDL compiler generates interface-specific ADC

CA-IDL compiler generates object skeleton

Developer implements real-time object

Runtime Services



Developer registers object with ORB

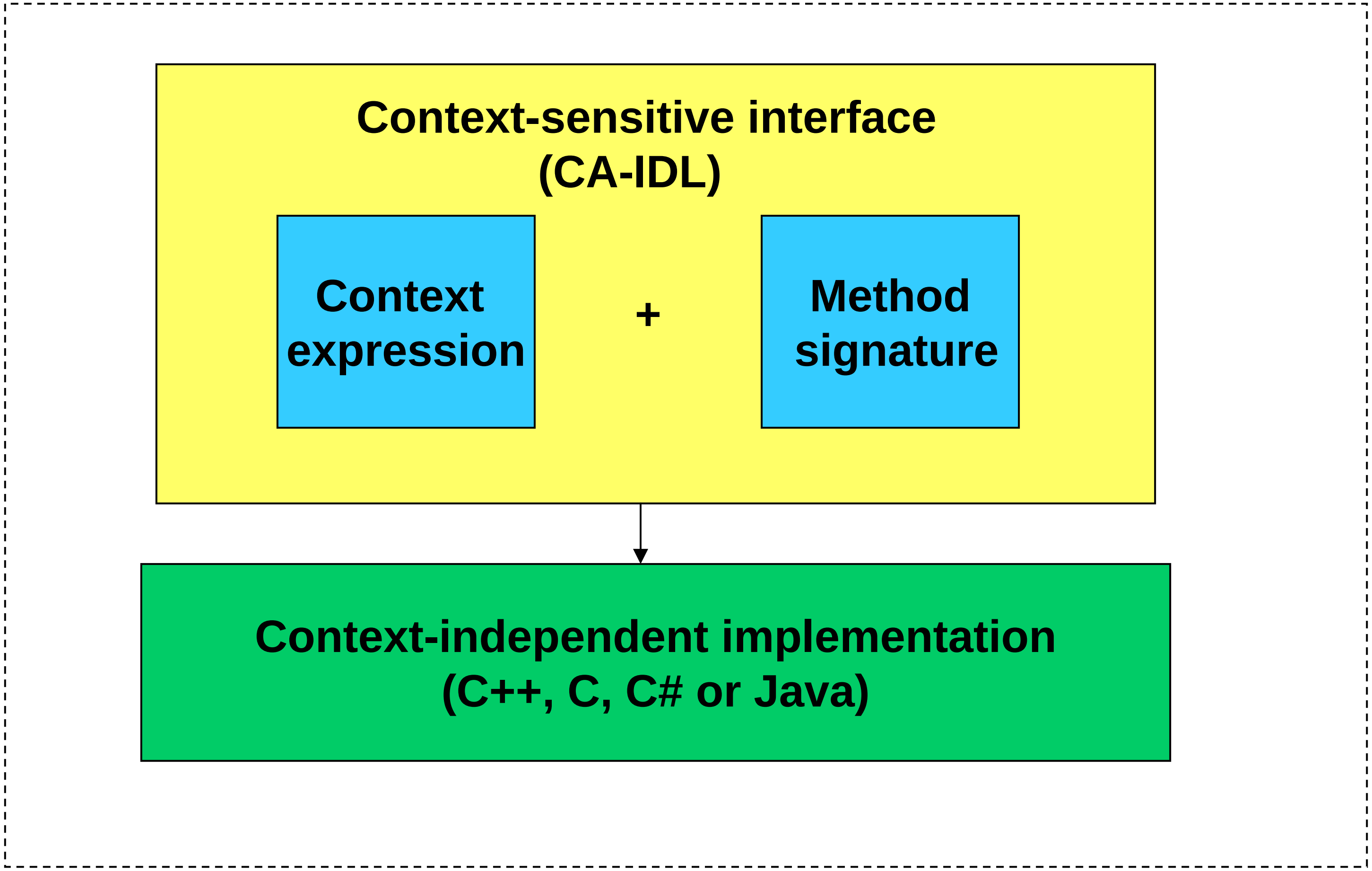
Context analyzers and R-CAP monitor and analyze context

ORB performs object discovery

ORB establishes context-triggered communication establishment

ADCs perform context-based object activation

Context-sensitive Application Object



Context-sensitive interface
(CA-IDL)

Context expression

+

Method signature

Context-independent implementation
(C++, C, C# or Java)

Specifying a Context

- Types of context data available depend on host device and its context-sensing capabilities
- Steps to port RCSM to a new device
  - Classify the context into categories
  - Define a structure type for each category

Device-specific Context

- Context information specific to a device
  - Remaining battery power, current time, number of objects running
  - Example:

```
RCSMContext
  DeviceSpecificContext {
    double battery_power
    double
  light_intensity
    double
  net_trans_rate};
```

Environment-specific Context

- Context information specific to surrounding environment
  - □ Current location, number of devices in vicinity, light intensity and current temperature
  - □ Example:

```
RCSMContext
  EnvironmentSpecificContex
  t {
    unsigned int
  num_peer_devices
    char [16] location};
```

User-specific Context

- Context information specific to the user
  - User information, number of times user runs an application
  - Example:

```
RCSMContext
   UserSpecificContext {
     unsigned int
   calendar_usage_rate};
```

Context Variables

- Use to express interest in the specific values of a context:

```
RCSMContext_var [category_type] [variable name] where
  [structure field] op [constant expression]
```

- Examples:
  - □ `RCSMContext_var DeviceSpecificContext C1 where (location = “GWC329”)`
  - □ `RCSMContext_var EnvironmentSpecificContext C3 where (num_peer_devices > 2)and (net_trans_rate >=40)`
  - □ `RCSMContext_var EnvironmentSpecificContext C2 where (num_peer_devices > 1)`

Temporal Operators

- Specify temporal relationships among multiple context variables

Operator	Usage	Description
Union: +	[(A1 + A2)*t*]	Either A1 or A2 is true for last time period *t*
Concatenation: ^	[(A1 ^ A2)*t*]	Both A1 and A2 are true for last time period *t*
Singular: ()	[(A1)*t*]	A1 has been true for last time period *t*
Precedence: ->	[(A1 -> A2)*t*]	A2 becomes true within *t* time units A1's being true

Context Expressions

- Represent relations among context variables using temporal operators
- We are interested in the condition that either C1 or C2 is true for the last 10 seconds:

```
 RCSMContext_var E1 where [(C1 + C2) 10]
```

Context-Sensitive Interface Specification

- Developer defines an interface for a context-sensitive real-time object by associating context variables and expressions with the method signature
  - [incoming] or [outgoing] tag
  - [activate-at-context x] tag with a context variable or expression

Incoming and Outgoing Tags

- Incoming: Invoke method after
  - Creating a context- triggered communication channel
  - Data is available from a remote object
- Outgoing
  - Invoke method first
  - Method generates data to transmit to a remote method with an incoming tag
- Compatibility

Interface Example

- ContextSensitivePrinter interface for an object that facilitates printing services by dynamically discovering printers in room GWC 329
- Two methods:
    - void SendDocumentstoPrinter(…)
    - void NotifyUser(…)

InterfaceExample

```
Interface
 ContextSensitivePrinter{
  [outgoing][activate at C1]
     void SendDocumentstoPrinter(…);
  [outgoing][activate at (C1 ^ C2)5]
     void NotifyUser(…);
```

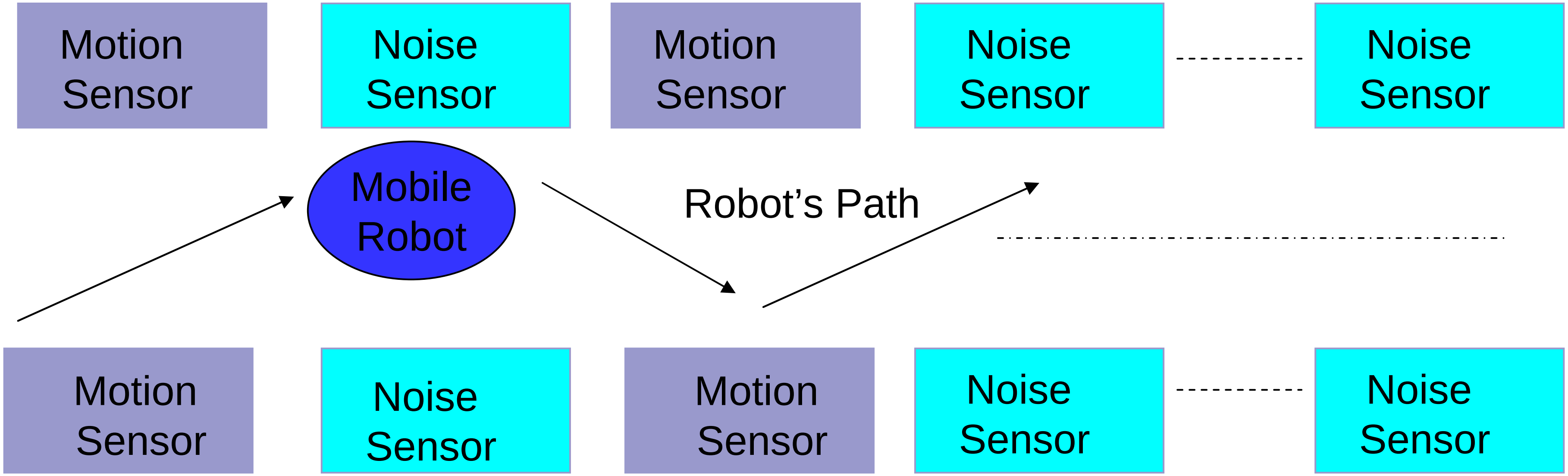
- Invoke SendDocumentstoPrinter whenever device detects it is in room GWC329,
  - Outgoing tag indicates method should generate data if a channel is established with another device (i.e. a printer)
- Invoke NotifyUser to ask user’s preference when more than one printer detected for more than 5 seconds

Example: Sensor Network

- System is a network of embedded sensors
- Two different types of sensors monitor network:
  - Motion
  - Noise
- Both types are stationary
  - Radio transmission range of up to 10 meters
- Mobile Robot
- Assume Object M, Object N and Object MB provide functionality for motion sensors, noise sensors and mobile robot

Example: Sensor Network

- Mobile robot collects data from sensors whenever robot within 10m of either sensor



Object MB: Mobile Robot Object

```
//Name: Mobile Robot Object
//Define a context variable
RCSMContext_var EnvironmentSpecificContext C
  where (num_peer_devices > 0);

//Interface Definition
Interface MB {
[incoming][activate at C]
  receive_noise_data([in] string data);
[incoming][activate at C]
  receive_motion_data ([in] string data);
};
```

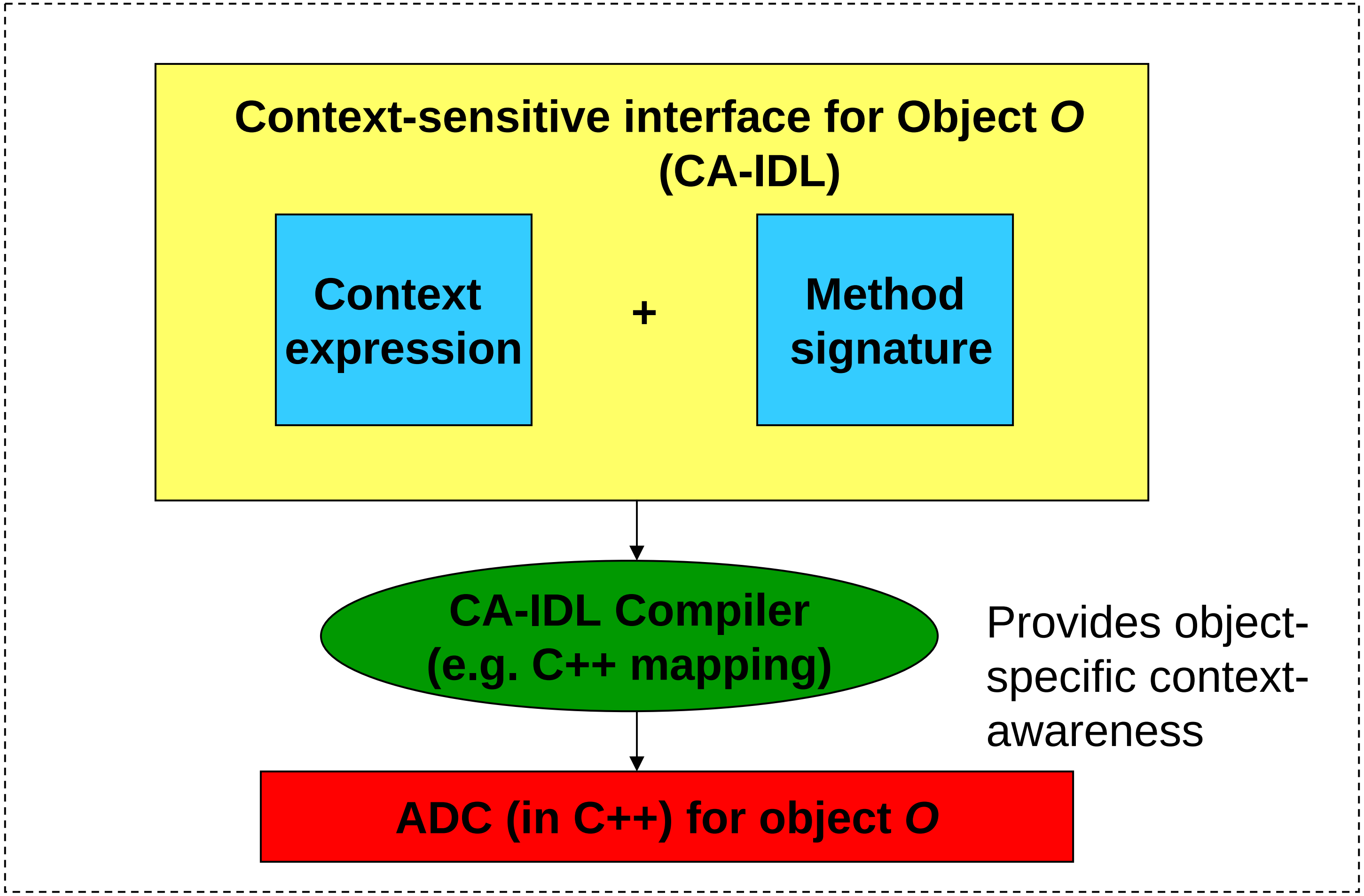
Object M: Motion Data Collector

```
//Name: Motion Data Collector
//Define a context variable
RCSMContext_var EnvironmentSpecificContext C
  where (num_peer_devices > 0);

//Interface Definition
interface M {
[outgoing][activate at C]
  exchange_motion_data([out] string data);
};
```

Object N: Noise Data Collector

```
//Name: Noise Data Collector
//Define a context variable
RCSMContext_var EnvironmentSpecificContext C
  where (num_peer_devices > 0);

//Interface Definition
interface N {
[outgoing][activate at C]
  exchange_noise_data([out] string data);
};
```

Adaptive Object Containers



Context-sensitive interface for Object *O* (CA-IDL)

Context expression

+

Method signature

CA-IDL Compiler (e.g. C++ mapping)

Provides object-specific context-awareness

ADC (in C++) for object *O*

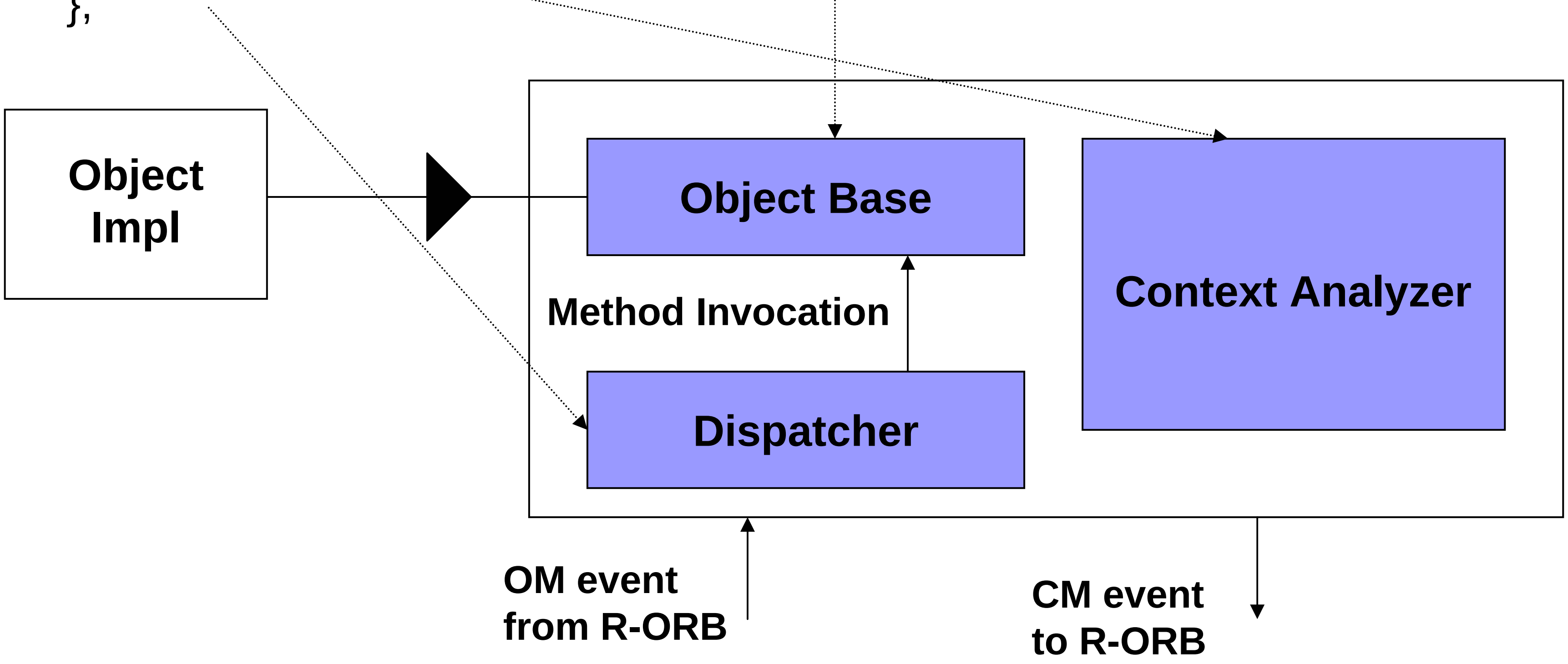
Adaptive Object Containers

- Register context-sensitive object and its interests with the R-ORB
- Receive context data from R-ORB
- Analyze data to check if context is true
- Activate context-sensitive object and invokes appropriate method

ADC Architecture

RCSMContext_var DeviceContext C where num_peer_devices > 0

```
Interface MB {
  Incoming  Activate at C void  receive_noise_data([in] string data)
};
```



Generated ADC

Context variable table for Object MB in the mobile robot

Row	Context Variable	Operator	Constant Expression	Specified Duration	*V*	True for duration	Method Id
1	Num_peer_devices	>	0	-	-	-	1
2	Num_peer_devices	>	0	-	-	-	2

Context variable table for Object M in the motion detectors

Row	Context Variable	Operator	Constant Expression	Specified Duration	*V*	True for duration	Method Id
1	Num_peer_devices	>	0	-	-	-	1

Context variable table for Object N in the noise detectors

Row	Context Variable	Operator	Constant Expression	Specified Duration	*V*	True for duration	Method Id
1	Noise_level	>	0	-	-	-	1

Context Propagation

- Sensors cannot detect each other
- Mobile Robot not within 10 m of any sensor
  - R-CAP propagates number of peer devices (0) to ADCs

Context Match Event

- Object MB and Object M both satisfy condition C
  - num_peer_devices > 0
- ADCs generate a “context match” event
  - Notifies R-ORB that context variable or expression is true

Object Discovery Messages

- Allow R-ORB in other devices to discover objects in the local device
- Robot’s R-ORB broadcasts:

```
{192.168.0.12, MB, receive_noise_data, {data,
  string}, none}
{192.168.0.12, MB, receive_motion_data, {data,
  string}, none}
```

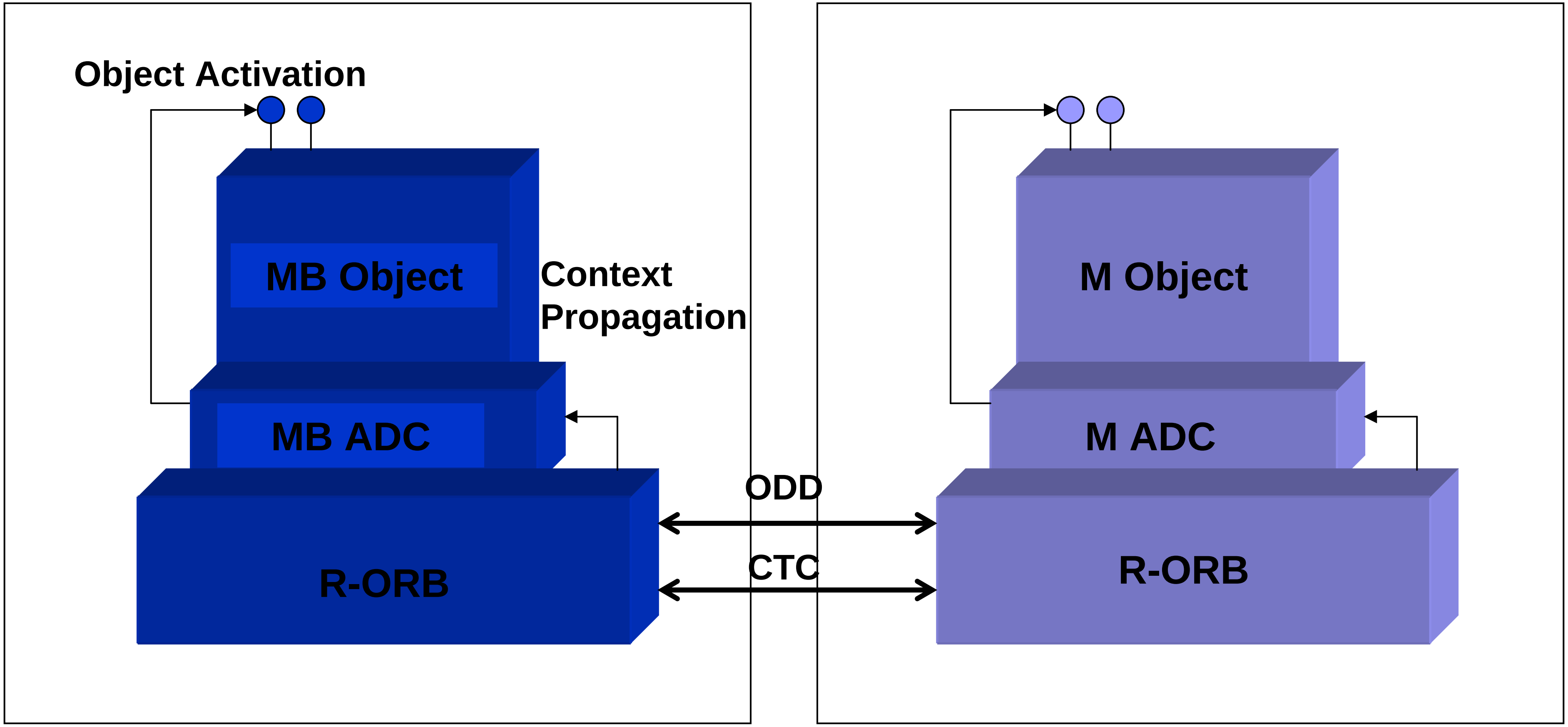
- Motion Detector’s R-ORB broadcasts:

```
{192.168.0.14, M, exchange_motion_data, {data,
  string}, none}
```

Object Match Events

- R-ORB in mobile robot checks for compatible methods
    - receive_motion_data
    - exchange_motion_data
- Generates an “object match” event
    - Notifies ADC that a compatible remote object is found

Inter-Object Communication



Mobile Robot

Motion Detector

Motion Detector Sends Data

- Notify ADC to invoke exchange_motion_data method and retrieve results
- Periodically check ADC to see if object data passed to R-ORB
- Create point-to-point communication channel with MB’s R-ORB.
    - Transmit data.
    - Terminate channel.

Object MB Receives Motion Data

- Check for data transmission from remote R-ORB
- Notify ADC of receive_motion_data to invoke method and pass in data to ADC.

R-ORB Implementation

- R-ORB also a context-sensitive object
- Context variables
  - Number of new devices detected
  - Number of existing devices no longer detected
  - Any CM event pending?
  - Any OM event pending?
- Initiate object discovery communication

Future Directions

- Situation Awareness
  - Capture and analyze context and interrelationships between users actions and devices
  - More intelligent; captures patterns over time
- Improving performance and energy efficiency
  - Hardware: Field programming gate arrays
  - Scalable cellular automata based coordination model
- Provide context-sensitive real-time scheduling support

Future Directions

- Smart Classroom for teaching and collaborative learning among college level students
    - Example: Instructor assigns students to work in groups
        - PDA’s form ad hoc networks
        - Instructor can dynamically join each group
- http://www.eas.asu.edu/~rcsm