

# CODE: Description Language for Wireless Collaborating Objects

#Raluca Marin-Perianu, Hans Scholten, Paul Havinga  
Department of Computer Science, University of Twente  
P.O. Box 217, 7500 AE Enschede  
{r.s.marinperianu, j.scholten, p.j.m.havinga}@ewi.utwente.nl

## Abstract

*This paper introduces CODE, a Description Language for Wireless Collaborating Objects (WCO), with the specific aim of enabling service management in smart environments. WCO extend the traditional model of wireless sensor networks by transferring additional intelligence and responsibility from the gateway level to the network. WCO are able to offer complex services based on cooperation among sensor nodes. CODE provides the vocabulary for describing the complex services offered by WCO. It enables description of services offered by groups, on-demand services, service interface and sub-services. The proposed methodology is based on XML, widely used for structured information exchange and collaboration. CODE can be directly implemented on the network gateway, while a lightweight binary version is stored and exchanged among sensor nodes. Experimental results show the feasibility and flexibility of using CODE as a basis for service management in WCO.*

## 1. INTRODUCTION

Wireless Sensor Networks is an emerging technology that has led to extensive studies concerning the new challenges that researchers and programmers have to overcome: energy efficiency, scarce computing and storage resources, unreliable communication, harsh environments, etc. Therefore, most of the initiatives have focused on tackling these difficulties rather than providing rich functionality within complex world scenarios. Consequently, usual applications utilize sensor nodes for monitoring or tracking purposes within a static pattern: collect data, perform some in-network processing (optional) and forward results to a central system. Wireless Collaborating Objects (WCO) is a paradigm that extends the traditional model of wireless sensor networks by transferring intelligence and responsibility to sensor nodes. Nevertheless, a single sensor node is limited in terms of hardware performance and energy available. In these conditions, the overall performance of the network can only be improved through cooperation among sensors.

The vision of WCO is that nodes self-organize into dynamic groups and offer services based on the capabilities of each member. Groups are formed either as a result of a system configuration or simply “on-demand”, as a result of a service request. The latter case is more dynamic, as groups

are formed based on context information and are dissolved when the service is accomplished.

Another idea of WCO is the dynamic nature of service deployment in the network. New services can be composed from existing ones, while other services can be deleted from the network. The service discovery protocol has to be adapted to such a highly dynamic environment and to enable service composition through a flexible vocabulary.

In this paper we propose CODE, a service description language as a means of describing collaborating objects and the services they offer. It can be used to support methods and tools to achieve service management in WCO. In particular, CODE allows for extensible descriptions of service interface, sub-services, service requirements, consuming entities and service attributes. The contributions of our work are as follows:

1. We identify the basic components of service management for WCO.
2. We propose CODE, a generic and extensible method for describing dynamic entities and services in WCO.
3. We show that the binary version of CODE is feasible to be stored and processed on sensor nodes.

The rest of this paper is organized as follows: Section 2 presents a scenario that motivates our approach for service management for WCO. Section 3 describes the basic components of service management and the role of the service description language. Section 4 covers the description of services and groups in WCO. Section 5 demonstrates the feasibility of binary CODE to work on sensor nodes. Section 6 discusses related work and Section 7 presents a summary and future work.

## 2. MOTIVATING SCENARIO

We present an example that illustrates most of the concepts of service management for WCO. The scenario concerns the process of transporting goods from providers to consumers. The products are placed shelves which are part of rolling carts. Each cart is equipped with a wireless sensor node, called a *micronode*. The nodes on the carts communicate with each other wirelessly, creating an ad-hoc network. They function on a battery, so energy saving is a major issue. Sensors called *piconodes* are placed on each shelf of the cart for monitoring the environmental conditions, which are

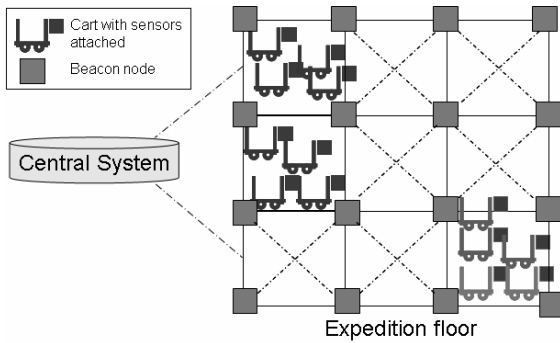


Fig. 1: Process diagram.

Carts gather on allocated grid rectangles within the expedition floor.

essential when the transport involves delicate and perishable goods (such as flowers).

In this scenario we focus on the expedition floor, which is a place used by the transport company to regulate the shipment process. A large grid is painted on the expedition floor and each cell of the grid is associated with a certain shop. Loaded carts arrive to the expedition floor and are placed depending on the shop they are assigned to. The carts belonging to one shop are grouped together and occupy a part of, one or more adjacent grid cells, depending on the number of carts.

A fixed infrastructure of sensor nodes is placed regularly in the grid on the expedition floor. These nodes are referred to as *beacons* and will enable the localization process. They have enough energy, as they are connected to the mains. Beacons will also act as communication gateways between the cart nodes and a central coordinator, called *Central System (CS)*, which is a computer in charge of coordinating all actions within the expedition floor.

CS informs the beacons on the assignment of carts to cells, on the environmental conditions that have to be observed and on the deadlines for carts gathering. The beacons actively calculate positions of the carts and verify correct gathering, environment conditions and deadlines. The events and results of the monitoring process are transmitted back to CS.

Figure 1 illustrates the process. The filled carts are placed on the expedition floor on certain grid cells, which are delimited by beacon nodes. Groups of carts placed within adjacent cells are transported together in the same trailer.

#### A. Group functionality

As the fixed infrastructure attached to the expedition floor is power supplied, it is desirable that it should take over much of the computation and monitoring services present in the network. The set of beacons which form the fixed infrastructure organize into groups for providing the following services:

1. Localization service, for the carts placed on the expedition floor.
2. Shop monitoring service, for the Central System.

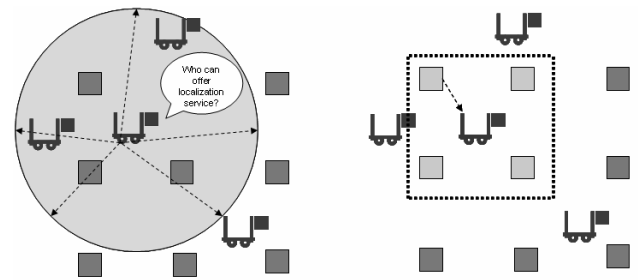


Fig. 2. Localization service.

The cart sends a broadcast message asking for localization service. Beacons that hear the broadcast organize into a group and deliver the service to the cart.

The second service is provided by groups of beacons formed as a result of a configuration message from CS, which assigns the shops and carts to be monitored and decides who are the members of the groups. The first service is offered on-demand and is provided by dynamically formed groups.

#### B. On-demand services

Services can be offered on-demand as a result of a service discovery message. Figure 2 shows an example of how beacons dynamically organize into a group that deliver a localization service for a nearby cart. On the left hand side of the picture, one cart broadcasts a service discovery message. Beacons that receive the broadcast organize into a group that is able to deliver the localization service, based on the information owned by each member. On the right hand side of the picture, the beacons that become members of the group are highlighted.

#### C. Service composition

The monitoring service performed by the group of beacons assigned to a shop is composed of a set of sub-services, concerning monitoring of carts gathering, monitoring of environmental conditions and verification of deadlines. Monitoring of carts gathering uses the location information service offered by each cart belonging to the shop being monitored. Carts, in their turn, use the localization service offered by groups of beacons from the neighbourhood. The monitoring of environmental conditions service uses the humidity, light and temperature services, which further rely on the environmental services offered by each cart. Figure 3 shows an example of service composition for monitoring the carts on the expedition floor. It can be noticed that taking advantage of the existing services, complex ones can be built and consequently, the level of functionality increases.

### 3. SERVICE MANAGEMENT

The above scenario points out the new concept of service management for WCO. Services are offered not only by single devices, but also by groups. Service offer is not fixed, but it changes depending on the context. Simple services can be combined resulting in more complex ones, and this process

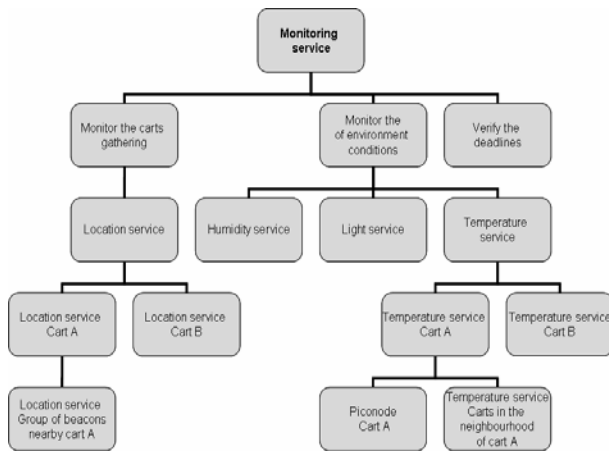


Fig. 3. Composition of services.

Each complex service is composed of a number of sub-services.

can be repeated. The outcome is a whole hierarchy of services.

We call *service management* the framework of service manipulation in WCO. It consists of the following modules:

1. *Service deployment*. Services are deployed on the network nodes in WCO.
2. *Service composition*. Complex services are constructed from the existing ones.
3. *Service discovery*. Discovery is the action of finding a service provider for a requested service.
4. *Service policy*. Service policy controls the access rights of the service consumer to the requested service.
5. *Service usage*. Service usage is the process of making use of the available service.

All the components of service management are directly dependant on an additional ingredient: *Service description*.

Deployment is assisted through description of service requirements. Services can be mapped on specific nodes of the network, or on entities which fulfil certain characteristics. The service requirements specify the desired features of service providers. Following the scenario explained in section 2, the temperature service is deployed only on carts outfitted with a piconode that measures temperature.

Service description enables composition through the definition of service interface. Interfaces dictate how the service can be accessed and controlled. A new service can be created by using the exposed interfaces of services already present in the network. It will have a set of sub-services and its own interface.

In order to enable discovery, it is necessary to describe the service through its name, type and other attributes. The service request message contains a partial description which is sent out to the network by the requesting entity. Nodes have to match the oncoming description with the one stored in the local memory to decide whether the service can be provided.

Service policy specifies which are the entities in the system that can access the service, more precisely the service consumers. They can be just enumerated, or they can be

identified based on certain characteristics specified within the service description.

Finally, the usage of the service is done through the exposed service interface, defined in the service description.

#### 4. SERVICE DESCRIPTION

Our notation for service description is based on XML, which provides a set of guidelines and conventions for structuring and representing data. It is generic and easily extensible and it is widely popular with W3C [11] as a means for structured information exchange and collaboration. By using XML as a building block for our description language, it is possible to take advantage of various publicly available open-source tools designed to work with XML.

CODE implements an XML schema which defines the elements that can appear in the description document. The key features of the schema are:

1. Attributes are not defined, in order to impose strict ordering of elements. This facilitates implementing efficient algorithms for tree traversal of descriptions.
2. The schema allows partial descriptions of elements. In this way, only important aspects can be expressed while keeping the size of the description to minimum.
3. CODE allows every entity and service in the system to have application-specific attributes.
4. Entities can be identified based on context information (e.g. neighbours).
5. The schema can operate with variables that denote entities in the system and take values according to the context.

There are two components that can be described using CODE: (1) *services* and (2) *entities*. An entity is a *device* or a *group* of devices. An entity can play the role of service provider and service consumer.

A device is described in terms of *hardware*, *software*, *dynamics* (fixed or mobile), *location*, *groups* that it is member of, *connected entities* (context information such as neighbours) and *services* it offers. A special type of device is the *sensor*, capable of measuring environmental conditions. Sensors are characterised by the type of measurements they perform and by the quality of service (range, resolution, accuracy). Groups are described by the *number of members*, the entity which is the *initiator* of group, the *group members*, the *leader* and the offered *services*.

Services are divided in two categories: *monitoring* and *non-monitoring*. Non-monitoring services are similar to an instantaneous function performed by the service provider, such as the average temperature measured by a group of nodes. They usually return a value as a result of a measurement or computation. Monitoring services perform timely supervision of a phenomenon, they are controlled through commands and issue events.

CODE defines the *interface* of a service, composed of inputs, outputs, commands and events. Moreover, it characterizes the *service consumers*, the *sub-services*, the

requirements for competent service providers and the attributes of the service.

The following subsections focus on the challenges addressed by service management in WCO, in particular the description of dynamic groups, on-demand services, service interface and sub-services.

#### A. Groups

Groups can be dynamically formed depending on the context. Context is defined by specifying a certain attribute, such as location, neighbouring devices or another parameter defined by the application.

Taking as example the scenario described in the previous section, beacons being in range of a cart form a group and provide a localization service to the cart. The cart for which the service is provided is known only at the moment it sends a service request. The cart is called the initiator of the group and its identity is not specified at the moment of service deployment. Instead, a variable is used to identify the cart, as shown in the following example:

```
<Group>
  <Initiator>
    <Device>
      <RefId>var_cart</RefId>
      <Type>Cart</Type>
    </Device>
  </Initiator>
  <MemberList>
    <Device>
      <Type>Beacon</Type>
      <ConnectedEntity>
        <ConnectedTo>
          <Device>
            <RefId>var_cart</RefId>
          </Device>
        </ConnectedTo>
        <Type>InRangeOf</Type>
      </ConnectedEntity>
    </Device>
  </MemberList>
  <OffersService>
    <Name>Localization</Name>
  </OffersService>
</Group>
```

The description of the group contains the field *Initiator*, which designates the cart requesting for localization service. Devices which are members of the group are the *Beacons*, with the restriction that they are *inRangeOf* the cart. The group offers the service named *Localization*. In the initialization phase, the variable denoting the cart is replaced with the appropriate cart ID.

#### B. On-demand services

CODE allows a service to be configured for being offered on-demand. The service requirements identify the potential providers of the service. In case that the provider is a group dynamically formed as a result of a service discovery request, the requirements specify the group characteristics. Here is a partial description of the localization service that uses the group description as requirements for service provider:

```
<Service>
  <Name>Localization</Name>
  <Type>
    <NonMonitoring>
      <ServiceOffer>on-demand</ServiceOffer>
    </NonMonitoring>
  </Type>
  <Requirements>
    [Group description...]
  </Requirements>
</Service>
```

#### C. Service interface

The service interface is composed of the following entities:

- *ParameterList* - Service parameters can be:
  - the *input* information, compulsory for service initialization
  - the *output* data, provided to the service consumer at the end of service utilization
- *EventList* - Events are messages sent to the service consumer to specify a state of the service. CODE defines the following categories of events: error, warning, alert, information and application defined.
- *CommandList* - Commands are control messages issued by the service consumer. They can trigger a previously defined event or can query the service for a specific piece of information.

A simple interface for localization service defines an output parameter, representing the calculated position of the cart:

```
<Interface>
  <ParameterList>
    <Parameter>
      <Direction>out</Direction>
      <Name>Position</Name>
      <Type>integer</Type>
    </Parameter>
  </ParameterList>
</Interface>
```

#### D. Service policy

Service policy consists of a set of entities which have the right to use the service. CODE allows specification of service policy by defining the characteristics of service consumers. Consumers do not need to be fully identified, rather they are characterised by an attribute or by the context. For example, a policy may state that entities which are allowed to access the temperature service are those located maximum two hops away from CartX.

```
<ServiceConsumer>
  <ConnectedEntity>
    <ConnectedTo>
      <Device>
        <Name>CartX</Name>
      </Device>
    </ConnectedTo>
    <Type>hops</Type>
    <Value>2</Value>
  </ConnectedEntity>
</ServiceConsumer>
```

### E. Sub-services

By means of interface definition, complex services can be developed from the existing ones. CODE allows definition of sub-services, which are a result of service composition. We take as an example the monitoring of carts gathering service, described in the transport scenario. Monitoring of carts uses the location service offered by each cart which has to be delivered to the specified shop.

```
<Service>
  <Name>Monitoring of gathering</Name>
  <Type>
    <Monitoring>
      <Status>running</Status>
    </Monitoring>
  </Type>
  <SubServices>
    <Service>
      <Name>Localization</Name>
      [other characteristics...]
    </Service>
  </SubServices>
</Service>
```

## 5. IMPLEMENTATION ON SENSOR NODES

CODE descriptions are based on XML, which provides flexibility and richness but is quite expensive in terms of memory usage and processing power needed for parsing. This is the reason why the original XML descriptions are stored at the gateway level, while on sensor nodes we have implemented a binary XML version. The binary CODE preserves XML flexibility and extensibility, while optimizing for low storage, fast processing, as required by the hardware limitations of sensor nodes. Firstly, we will present the platform on which the tests have been performed. Secondly, we will analyse the storage space needed for binary CODE descriptions. Thirdly, we will show the processing time needed for matching a service discovery message to the service description stored on the node.

### A. Platform

The implementation has been done on a hardware platform developed by NEDAP for the Eyes project [7]. It consists of MSP430f149 micro-controller from Texas Instruments that operates at 4MHz. It has 60kB of program flash memory and 2kB of RAM. Other features include a radio transceiver and an RS232 interface. The operating system used is DCOS (Data Centric Operating System) [8], which occupies 5kB out of the 60kB of flash memory.

### B. Binary descriptions

The XML schema which defines CODE is used as the basis for the binary encodings. We have used Java API for XML Processing (JAXP) [9] for parsing the XML descriptions through the Document Object Model (DOM) [10] interface. The XML DOM views XML documents as a tree structure of elements embedded within other elements. From this tree structure we extract only the sub-tree containing the necessary information, leaving out the text descriptions and comments. The nodes in our tree can be *element* or *text*. Considering that

TABLE 1: IMPLEMENTATION RESULTS

Service name	Size of uncompressed description	Size of binary description	Number of tree nodes (E+T)	Matching time of binary description
Localization	1.65 kB	114 B	57	7 ms
Temperature monitoring	2.58 kB	202 B	101	9 ms
Monitoring of gathering	3.52 kB	296 B	148	10 ms

the XML schema contains 51 element types, we used one byte for encoding, thus leaving space for further extensions. We have pre-defined 5 types of text nodes: string, byte, int\_16, int\_32 and float. We have chosen to encode also the strings, as the storage space is very limited. The resulting tree is further linearized, in depth first tree traversal order. Considering that we have  $E$  number of element nodes and  $T$  number of text nodes, the number of bytes  $M$  occupied by the binary description is  $M = 2 * E + x * T$ , where  $x$  is a number between 2 and 5 and it represents the average space occupied by a text node, depending on its type. Table 1 shows the results of compression for three of the services described in Section 2. The monitoring of gathering service defines a complex interface that allows the Central System to fully control the service. It has 96 element nodes and 52 text nodes, resulting in a binary description of only 296 bytes, which is still small enough to be stored in RAM.

### C. Matching binary descriptions

In order to support service discovery, each node in the network stores the descriptions of the services it offers. Incoming service discovery messages are only partial descriptions of the service and they have to be kept as small as possible, for achieving fast transmission and small power consumption. In order to decide whether the service request is satisfied, we have implemented a matching algorithm between the general description and the partial one. Taking advantage of the fact that (1) the trees to be matched have the same root, so matching of nodes is done at the same tree level, (2) matching of branches goes until the level of leaves and (3) the element nodes are placed in strict order, we have implemented a non-recursive algorithm that traverses the logical trees of the two binary descriptions. The tree corresponding to the complete description is traversed in one way, without returning, while the tree associated with the partial description is traversed back and forth, depending on the matching result. At each step of the traversal we keep the current levels of the trees equal. The upper bound of the algorithm complexity is  $O(N)$ , where  $N = E+T$  is the total number of nodes in the tree associated with the general description. The lower bound of the algorithm is  $O(n)$ , where  $n$  is the total number of nodes in the tree associated with the partial description.

The results of the experiments are shown in Table 1. The implementation revealed an execution time of 7ms for localization service, considering a general description with  $N=57$  (41 element nodes and 16 text nodes, which represent

the leaves of the tree) and a partial description with  $n=15$  (12 element nodes and 3 text nodes). The implementation of the matching algorithm occupies 1.5kB of flash memory.

## 6. RELATED WORK

The challenges that WCO bring into light change the traditional vision of service discovery protocols [4-6]. These protocols allow discovery of services in networks where the only possible service providers are stand-alone devices. The notion of on-demand services does not exist, as the service offer does not depend on the context. Moreover, there is no support for service composition.

Nevertheless, two description languages have been proposed particularly focusing on sensor networks: SensorML [1, 2] and TinyML [3]. SensorML provides an XML schema for defining the geometric, dynamic, and observational characteristics of a sensor in support of data discovery. It supports the processing and analysis of the sensor measurements, the geolocation of observed values and provides performance characteristics. TinyML describes sensor platforms, sensor fields (a collection of sensor nodes) and virtual devices (a group of sensor nodes that perform a certain task). Both SensorML and TinyML define the notion sensor group, which is composed of multiple sensors that operate together to provide a collective observation or related group of observations. Unfortunately, these languages do not describe services in a sensor network, they only concentrate on defining the characteristics of devices. Moreover, XML descriptions are placed on the external interface, not on the sensor nodes. In-network processing of sensor descriptions is not one of the goals of Sensor ML or TinyML.

## 7. CONCLUSIONS

This paper introduces CODE, a description language for Wireless Collaborating Objects (WCO). WCO improve the traditional model of wireless sensor networks by enabling complex services to be offered either by single nodes or by groups of collaborating nodes. The purpose of CODE is to support service management for WCO, more specifically service deployment, service composition, service discovery, service policy and service usage. CODE is based on XML, widely used for structured information exchange and collaboration. The XML descriptions are stored at the gateway level, while the sensor nodes store and process the binary version. The binary CODE preserves XML flexibility and extensibility, while optimizing for low storage, fast processing, as required by the hardware limitations of sensor nodes. We have showed that a binary description of service is small enough to be stored on sensor nodes with 2kB of RAM. Moreover, we have implemented an algorithm that matches a partial description against the complete one, thus providing the response to a service discovery message. The processing time is the order of milliseconds, negligible in comparison with the communication overhead.

Future work will include extending the matching mechanism by allowing more complex validations, such as range checking. This is clearly useful for practical situations, for instance the case of requesting a localization service with accuracy over a certain degree. We will also add functionality to the manipulation of binary descriptions, such as updating variables or changing fields which depend on context. Additionally, we will explore architectural choices for building efficient service discovery mechanisms on top of CODE descriptions.

## REFERENCES

- [1] Sensor Model Language (SensorML), 2004. <http://vast.nsstc.uah.edu/SensorML/>
- [2] M. Botts (ed.) "Sensor Model Language (SensorML) for In-situ and Remote Sensors", OpenGIS Interoperability Program Report, OGC 04-019r2, Version 1.0.0 beta, Nov. 2004, [http://vast.nsstc.uah.edu/SensorML/SensorML\\_04-019\\_1.0\\_beta.pdf](http://vast.nsstc.uah.edu/SensorML/SensorML_04-019_1.0_beta.pdf)
- [3] Nathan Ota and William T.C. Kramer, "TinyML: Meta-data for Wireless Networks", <http://kingkong.me.berkeley.edu/~nota/research/TinyML/project-paper-1.pdf>
- [4] UPnP Forum, "UPnP device architecture", Version 1.0, June 2000, <http://www.upnp.org/>
- [5] Sun Microsystems, "Jini architecture specification" version 2.0, June 2003, [http://www.sun.com/software/jini/specs/jini2\\_0.pdf](http://www.sun.com/software/jini/specs/jini2_0.pdf)
- [6] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," IETF, RFC 2608, June 1999, <http://www.rfc-editor.org/rfc/rfc2608.txt>.
- [7] Energy-Efficient Sensor Networks (EYES), <http://www.eyes.eu.org/>
- [8] T.J. Hofmeijer, S.O. Dulman, P.G. Jansen and P.J. Havinga, "DCOS, a Real-Time Light-Weight Data Centric Operating System", In *ACST 2004*.
- [9] Sun Microsystems, Inc., Java API for XML Processing (JAXP) 1.1 Public Review 2. <http://java.sun.com/aboutJava/communityprocess/revi ew/jsr063/jaxp-pd2.pdf>.
- [10] A. L. Hors, P. L. Hegaret, L. Wood, G. Nicol, J. Robie, M. Champion and S. Byrne (Eds). "Document Object Model (DOM) Level 3 Core Specification" Version 1.0, W3C Recommendation, Apr. 2004, <http://www.w3.org/TR/DOM-Level-3-Core/>
- [11] W3C World Wide Web Consortium, <http://www.w3.org/>