The PhoneMob System: Detailed Models

Companion report to the book:

Building Enterprise Systems with ODP
An Introduction to Open Distributed Processing

Peter F. Linington, Zoran Milosevic, Akira Tanaka and Antonio Vallecillo

June 25, 2011

http://theodpbook.lcc.uma.es/
## Contents

1 Introduction 3

2 Enterprise viewpoint specifications 3
   2.1 Overall organization 3
   2.2 The Phone Repair Community 4
      2.2.1 Enterprise objects of the Phone Repair community 4
      2.2.2 Enterprise roles of the Phone Repair community 4
      2.2.3 Filling of roles 5
      2.2.4 Policies of the Phone Repair community 5
      2.2.5 Processes of the Phone Repair community 5
      2.2.6 Interactions of the Phone Repair community 12
   2.3 The CustomerOrg Community 16
      2.3.1 Enterprise objects and roles of the CustomerOrg community 16
      2.3.2 Policies of the CustomerOrg community 16
      2.3.3 Processes of the CustomerOrg community 16
      2.3.4 Interactions of the CustomerOrg community 19
   2.4 The Logistics Provision Community 21
      2.4.1 Enterprise objects and roles of the Logistics Provision community 21
      2.4.2 Processes of the Logistics Provision community 22
   2.5 Relationships between communities 24

3 Information viewpoint specifications 25
   3.1 Overall organization 25
   3.2 Information object types 26
   3.3 Information action types 27
   3.4 Static schemata 28
   3.5 Dynamic schemata 29

4 Computational viewpoint specifications 31
   4.1 Overall organization 31
   4.2 Data Types 32
   4.3 Software Architecture 33
   4.4 Behaviour 41
      4.4.1 User interactions 42
      4.4.2 HQ staff interactions 46
      4.4.3 Courier staff interactions 51
      4.4.4 Repair Centre staff interactions 53
      4.4.5 Manufacturer interactions 60
      4.4.6 Flows 62

5 Engineering viewpoint specifications 63
   5.1 Overall organization 63
   5.2 Engineering objects 63
   5.3 Objects distribution 63
   5.4 Engineering channels 64

6 Technology viewpoint specifications 67
   6.1 Overall organization 67
   6.2 Technology objects 67
   6.3 IXIT 67
   6.4 Implementable standards 67
   6.5 Technology processes 67

7 Correspondences 72
   7.1 Correspondences between Enterprise and Information viewpoints 72
   7.2 Correspondences between Enterprise and Computational viewpoints 76
   7.3 Correspondences between Enterprise and Engineering viewpoints 80
   7.4 Correspondences between Information and Computational viewpoints 82
   7.5 Correspondences between Computational and Engineering viewpoints 85
   7.6 Correspondences between Engineering and Technology viewpoints 89
1 Introduction

This document describes the specifications that make up the PhoneMob example, focusing on their structure, organization and internal contents. It contains a full version of the models that comprise the ODP specification of the system, as described in the book Building Enterprise Systems with ODP — An Introduction to Open Distributed Processing [1]. These models, in MagicDraw format, as well as this document, are available from the book’s companion website: http://theodpbook.lcc.uma.es/.

Figure 1: The overall structure of the PhoneMob system specification.

Figure 1 shows the overall structure of the PhoneMob specifications, expressed in the UML4ODP notation [2]. It is organized in eleven UML packages: five for the individual viewpoints, and six for the correspondences between them (since not all correspondences are of relevance for this system). These packages are detailed in the following sections.

2 Enterprise viewpoint specifications

2.1 Overall organization

The enterprise specification of the PhoneMob system is expressed by one model, stereotyped «Enterprise_Spec», which contains the contracts of all the communities that make up the specification. This is shown in figure 2. The diagram also shows the field of application of the PhoneMob specification, which describes the properties of the environment of the ODP system for which that specification is to be used. It is expressed by means of a tagged value of the «Enterprise_Spec» model.

Figure 2 shows that the enterprise specification of the PhoneMob system is composed of the specifications of three communities: Phone Repair (section 2.2), CustomerOrg (section 2.3) and Logistics Provision (section 2.4). The specification of each community is defined in terms of its contract. The contracts for the three communities are shown in figures 3, 25 and 36; these are detailed below.

The enterprise specification modelled here corresponds to the more detailed specification of the communities, and in particular of the Phone Repair community as described in the ODP book. The refinement steps taken in developing the community, which were mentioned in the book, are not presented here.
2.2 The Phone Repair Community

The contract for the Phone Repair community is shown in figure 3. All community contracts have the same structure: one objective (which in a real contract would be more specific and detailed) and four main packages with the specification of the community roles, object types, processes and policies.

2.2.1 Enterprise objects of the Phone Repair community

Figure 4 shows the contents of the first package, setting out the object types defined in the Phone Repair community. The enterprise object type PhoneMob represents the whole business unit.

Two of the enterprise objects (Handset and Loan Handset) have associated state machines. These are shown in figures 5(a) and 5(b) respectively.

2.2.2 Enterprise roles of the Phone Repair community

The roles of the Phone Repair community are shown in figure 6. We can see how the Phone Repair Provider role has been decomposed into the four roles that refine it at a more detailed level of the specification, as described in chapter 2 of the ODP book. This role has been maintained in this model because it is useful in the specification of some interactions (see for example figures 18 to 23) allowing the representation of the business at an appropriate level of abstraction, independently from the particular roles that refine it at a more detailed level. This role is filled by the PhoneMob enterprise object (see figure 7).
2.2.3 Filling of roles
We also need to specify the assignment policies that describe the conditions under which enterprise objects can fill the enterprise roles in a community. As long as an object satisfies the behavioural conditions stated in the role definition, it can be assigned to the role and thus participate in the community. Figure 7 shows the filling of roles in the Phone Repair community.

2.2.4 Policies of the Phone Repair community
In an enterprise specification, policies are used wherever the desired behaviour of the system may be changed to meet particular circumstances. Figure 8 shows the contents of the Policies package of the Phone Repair contract. It defines two policies, which are used later in figures 20, 21, 22 and 23.

2.2.5 Processes of the Phone Repair community
The third component of an enterprise community contract is the specification of its processes. In UML4ODP they are expressed as UML activities. In the activity diagrams, actor roles are expressed by «EV_Role» activity partitions, and artefact roles as object nodes stereotyped «EV_Artefact». Figure 9
shows the processes defined for the Phone Repair contract. These processes are detailed in the following diagrams (figures 10 to 16).
It is assumed that these SLAs include contract conditions such as performance targets and payment schedule; these will set the conditions in the related policy envelopes (see policy diagrams).

Figure 10: Process Establish Repair SLA in the Phone Repair community.

Figure 11: Process Establish Logistics SLA in the Phone Repair community.
Figure 12: The Repair Process in the Phone Repair community.
Figure 13: The Repair Process in the Phone Repair community, refined by expanding role Phone Repair Provider into its refining roles (Branch system, Branch staff and HQ system).
The following diagrams describe the performance review processes carried out in the Phone Repair community. The first one is led by a staff member in the headquarters, who interacts with the external agents (customers and service providers) to collect their performance indicators and provide them with feedback. The staff member uses the headquarters system to store and process the collected data and to generate the reports. The second diagram (figure 15) describes the process that reviews the performance of the company branches. In this case the process is automated and happens between computerized systems, led by the headquarters system and without any direct involvement of any staff member.

Figure 14: Process Review Performance in the Phone Repair community.

Figure 15: Process Review Branch Performance in the Phone Repair community.
Figure 16: Process Phone Loan Policy Setting Behaviour in the Phone Repair community.
2.2.6 Interactions of the Phone Repair community

Behaviour can be modelled in the ODP enterprise language in terms of processes or in terms of interactions. Modelling of behaviour in terms of interactions between roles in a community is appropriate when the modelling focus is placed primarily on the roles and artefacts involved in the behaviour, and on the relationships between them.

Figure 17 shows the interactions defined in the Phone Repair contract. These interactions are detailed in the following diagrams.

Figure 17: Interactions specified in the Phone Repair community.

Figure 18: The Repair interaction, focusing on the Phone Supplier.
Figure 19: The Repair interaction, focusing on the User.

Figure 20: The Repair interactions.
Figure 21: BillingSupplier interaction.

Figure 22: BillingCustomer interaction.
Figure 23: BillingLogistics interaction, and its policy constraints.

Figure 24: Get Loan Phone interaction, and its policy constraints.
2.3 The CustomerOrg Community

The contract for the CustomerOrg community is shown in figure 25.

![Diagram of CustomerOrg community contract]

Figure 25: Community contract for the CustomerOrg community.

2.3.1 Enterprise objects and roles of the CustomerOrg community

Figure 26 shows the enterprise object types, roles and assignment policies that describe which enterprise objects can fill the enterprise roles in the CustomerOrg community.

![Diagram of enterprise objects and roles]

Figure 26: Filling of roles in the CustomerOrg community.

Some of the enterprise objects and roles of this community have associated state machines. For example, figures 27(a) and 27(b) show the state machine of the Document enterprise object and Employee role, respectively.

2.3.2 Policies of the CustomerOrg community

Figure 28 shows the contents of the Policies package of the CustomerOrg contract. They are shown later in figures 34 and 35, respectively, associated to the corresponding interactions.

2.3.3 Processes of the CustomerOrg community

Figure 29 shows the processes defined in the CustomerOrg contract. These processes are detailed in the following diagrams (figures 30, 31 and 32).
(a) The Document enterprise object state machine.

(b) The Employee role state machine.

Figure 27: State machines of some enterprise objects and roles in the CustomerOrg community.

Figure 28: Policies defined in the CustomerOrg community.

Figure 29: Processes defined in the CustomerOrg community.
Figure 30: Repair Request Process process in the CustomerOrg community.

Figure 31: Repair Report Process process in the CustomerOrg community.
2.3.4 Interactions of the **CustomerOrg** community

This community defines two interactions, Request Repair and Temporary Phone Request. The first one is shown in figures 33 and 34. The second interaction and its governing policy is shown in figure 35.

Figure 33: A Request Repair interaction in the **CustomerOrg** community.
Policy Value must either allow or block the Request Repair interaction; it must do so on the basis only of the states of the User and the Mobile Phone and values computed from them.

Allow interaction if:
- A requester is an Employee; and
- His/her Mobile Phone is broken; and
- Repair request submitted during daytime from 9:00 until 15:00.

Figure 34: The Request Repair interaction and its governing policy.

(Policy Value must either allow or block the Temporary Phone Request interaction; it must do so on the basis only of the states of the Employee and the Repair Request and values computed from them.)

{PolicyValueRule=Allow interaction if:
- A requester is an Employee, and
- Repair request was submitted in less than two days before, and
- Temporary phone request has been submitted during daytime from 9:00 until 15.00.}

Figure 35: The Temporary Phone Request interaction and its governing policy.
2.4 The Logistics Provision Community

The contract for the Logistics Provision community is shown in figure 36.

Figure 36: Community contract for the Logistics Provision community.

2.4.1 Enterprise objects and roles of the Logistics Provision community

Figure 37 shows the enterprise object types, roles and assignment policies that describe which enterprise objects can fill the enterprise roles in the Logistics Provision community.

Figure 37: Filling of roles in the Logistics Provision community.
2.4.2 Processes of the Logistics Provision community

Figure 38 shows the processes defined in the Logistics Provision contract. They are detailed next.

![Diagram showing processes defined in the Logistics Provision community](image)

Figure 38: Processes defined in the Logistics Provision community.

![Diagram showing process Inquiry](image)

Figure 39: Process Inquiry.

![Diagram showing processes BilledByCarrier and BillCustomer](image)

(a) Process BilledByCarrier.
(b) Process BillCustomer.

Figure 40: Process BilledByCarrier and BillCustomer in Logistics Provision community.
Figure 41: Process Delivery.
2.5 Relationships between communities

In the PhoneMob system specifications, the communities interact because some enterprise objects fulfil roles in more than one of them. This is shown in figures 42 and 43, respectively.

---

**Figure 42:** Objects filling roles in the Logistics Provider and Phone Repair communities.

**Figure 43:** Objects filling roles in the Phone Repair and CustomerOrg communities.
3 Information viewpoint specifications

3.1 Overall organization

All the elements that constitute the information specification of the system are gathered together within a single model, stereotyped «Information_Spec». This model contains the set of packages that express the invariant, static and dynamic schemata of the information specification, structuring them in different packages for organizational purposes. Figure 44 depicts the «Information_Spec» model for the PhoneMob system.

![Figure 44: Overall structure of the PhoneMob information specification.](image)

One set of packages describe the information object types (figure 45) and their associated attributes, associations and state machines; the other give a selection of the information action types (figure 46), and the specification of some of the static schemata relevant to the system (figures 47 and 48).

Note that in the «Information_Spec» model shown in figure 44 there is no separate package for the specification of dynamic schemata because they are modelled in UML as state machines associated with the corresponding object types.
3.2 Information object types

The following diagram shows the information object types of the PhoneMob system specification.

![Information object types diagram](image-url)

Figure 45: Information object types.
3.3 Information action types

The following diagram shows the information action types of the PhoneMob system specification. They characterize the information actions that model the information processing in the PhoneMob system. Actions cause state changes in the objects that participate in them. Such changes are specified by the corresponding dynamic schemata, see section 3.5.

Figure 46: Information actions types.
3.4 Static schemata

The following diagrams show the two static schemata defined in the PhoneMob system specification. The first one describes the initial state of the system, with only one repair centre, one staff member associated to it, and two loan handsets. The second one represents an example of the state of a RepairOrder at one given moment in time.

Figure 47: A static schema stating the initial state of the system.

Figure 48: Current state of an example RepairOrder.
3.5 Dynamic schemata

The following figures show the UML state machines that represent the dynamic schemata of the information objects described in figure 45. The possible triggers for the possible transitions are information actions, whose types have been defined in figure 46.

Figure 49: State machine for the Handset information object.

Figure 50: State machine for the LoanHandset information object.

Figure 51: State machine for the Company information object.
Figure 52: State machine for the RepairOrder information object.

Figure 53: State machine for the ServiceCentre information object.

Figure 54: State machine for the UserHandset information object.

Figure 55: State machine for the User information object.

Figure 56: State machine for the Staff information object.
4 Computational viewpoint specifications

4.1 Overall organization

The overall structure of the computational viewpoint specification of the PhoneMob system is shown in figure 57. It contains three high-level packages. One describes the software architecture of the application. The second deals with the behavioural aspects of the system. The last one contains the basic data types used in the specification. The model for the complete computational specification has some associated tag values that determine which transparencies are required for the system. In this case the transaction and replication transparencies are needed, together with the access and location transparencies which are mandatory for any computational specification — and thus there is no need to explicitly specify them.

Figure 57: The overall structure of the computational specification.
4.2 Data Types

Figure 58 shows the data types used by the computational objects. All elements are stereotyped «dataType», although this is not shown in the diagram for simplicity.
4.3 Software Architecture

The elements that comprise the Software Architecture UML package are shown in the following four figures. Each one progressively adds details to the previous one. Thus, figure 59 shows the computational objects, grouped in packages for organizational purposes. The outer packages correspond to the architecture layers. Inside them, objects are grouped depending on the kind of services they provide. Auxiliary services (such as login, for instance) have been included because they provide a set of common functions to the rest of the objects.

The second diagram (figure 60) shows the UML ports of the components that represent the computational objects (ports represent ODP interfaces) and the primitive bindings between them (represented by means of UML dependencies stereotyped «CV_PrimitiveBinding»).

These UML dependencies are an abstraction of the more detailed way of expressing the bindings than the one used in figure 61, which uses the UML ball-and-socket notation to specify the UML interfaces that represent the operational signatures of the ODP interfaces and how they are interconnected.

Finally, figures 62 to 63 describe the UML interfaces described in the previous figure 61 to explicitly show the signature of their operations.

Figure 59: Structure of the software architecture of the PhoneMob system.
Figure 60: PhoneMob system software architecture — ODP interfaces and primitive bindings.
Figure 61: PhoneMob system software architecture — primitive bindings detailed.
Figure 62: Operation interface signatures for the user interface objects.
Figure 63: Operation interface signatures for the external objects.

Figure 64: Operation interface signatures for the administrative user interface objects.
Figure 65: Operation interface signatures for the application objects.
Figure 66: Operation interface signatures for the administrative application objects.
Figure 67: Operation interface signatures for the data management objects.
4.4 Behaviour

Figure 68 shows the structure of the behavioural specification of the PhoneMob system, from the computational viewpoint. It is divided into two main packages: Streams and Interactions. The first one describes the specification of a stream that describe a continuous flow of data. The contents of this package are described in figure 107 in section 4.4.6.

![Figure 68: Structure of the computational viewpoint behavioural specification.](image)

Operational interactions in the computational viewpoint are described in this section by a set of UML sequence diagrams that specify the pairwise interactions between the computational objects involved in realizing a piece of behaviour for the PhoneMob system.

The structure of the UML package with these interactions is shown in figure 69. For organizational purposes it is divided into a set of subpackages, according to the computation object that initiates the behaviour (of course, other organizations would be equally valid).

![Figure 69: Structure of the computational interactions specification.](image)

The following sections will describe the contents of these packages in detail.
4.4.1 User interactions

Figure 70: Computational interaction: a User object logs in.

Figure 71: Computational interaction: a User object requests a repair.
Figure 72: Computational interaction: a User object requests information about a repair order.

Figure 73: Computational interaction: a User object changes the return address.
Figure 74: Computational interaction: a User object gets his/her personal details.

Figure 75: Computational interaction: a User object closes a repair request.
Figure 76: Computational interaction: a User object gets the estimated closing date for repair order.
4.4.2 HQ staff interactions

Figure 77: Computational interaction: a HQStaff object logs in.

Figure 78: Computational interaction: a HQStaff object invoices a customer.
Figure 79: Computational interaction: a HQStaff object monitors performance.

Figure 80: Computational interaction: a HQStaff object negotiates the SLA with a customer.
Figure 81: Computational interaction: a HQStaff object negotiates the SLA with a logistics provider.
Figure 82: Computational interaction: a HQStaff object negotiates the SLA with a manufacturer.
Figure 83: Computational interaction: a HQStaff object pays pending bills.

Figure 84: Computational interaction: a HQStaff object sets the Phone Loan Policy.
4.4.3 Courier staff interactions

Figure 85: Computational interaction: a CourierStaff object logs in.

Figure 86: Computational interaction: a CourierStaff object gets pending deliveries.
Figure 87: Computational interaction: a CourierStaff object closes a successful delivery.

Figure 88: Computational interaction: a CourierStaff object closes a failed delivery.
4.4.4 Repair Centre staff interactions

Figure 89: Computational interaction: a RepairCentre Staff object logs in.

Figure 90: Computational interaction: a RepairCentre Staff object checks a User.
Figure 91: Computational interaction: a RepairCentre Staff object registers a deceased Handset.

Figure 92: Computational interaction: a RepairCentre Staff object registers a repaired Handset.
Figure 93: Computational interaction: a RepairCentre Staff object assigns a loan Handset.

Figure 94: Computational interaction: a RepairCentre Staff object searches for repair orders.
Figure 95: Computational interaction: a RepairCentre Staff object gets the details of an order.

Figure 96: Computational interaction: a RepairCentre Staff object gets the actions in a repair order.

Figure 97: Computational interaction: a RepairCentre Staff object registers an action in a repair order.
Figure 98: Computational interaction: a RepairCentre Staff object registers a returned loan handset.

Figure 99: Computational interaction: a RepairCentre Staff object registers a repair order.
Figure 100: Computational interaction: a RepairCentre Staff object updates a repair order.

Figure 101: Computational interaction: a RepairCentre Staff object returns a handset to a user.
Figure 102: Computational interaction: a RepairCentre Staff object sends a handset to the manufacturer.
4.4.5 Manufacturer interactions

Figure 103: Computational interaction: an Manufacturer object logs in.

Figure 104: Computational interaction: an Manufacturer object closes a repair.
Figure 105: Computational interaction: an Manufacturer object sets a new estimated repair date.

Figure 106: Computational interaction: an Manufacturer object gets pending repairs.
4.4.6 Flows

Figure 107 shows the architecture of the videoconference services, composed of two kinds of computational objects (the repairer and the adviser). Then, figure 108 shows a different architecture that uses a binding object to connect the participants.

Figure 107: Using streams when specifying a multimedia application.

Figure 108: Using a binding object in the multimedia application.
5 Engineering viewpoint specifications

5.1 Overall organization

The overall organization of the engineering specification of the PhoneMob system is shown in figure 109. It contains two main packages, one with the basic engineering objects and other with the distribution structure.

```
Figure 109: The overall structure of the engineering specification.
```

5.2 Engineering objects

Package NV_Objects in figure 109 describes the basic engineering objects. They correspond to the computational objects described in the computational specification (see figure 59). The same grouping structure is used again here, although just for packaging purposes, since it does not not imply any constraints on the distribution. One package contains the ODP objects that implement either common functions (such as the Trader) or services that provide the required transparencies (the TransactionMgr: remember that the computational specification stated that transaction transparency was required).

5.3 Objects distribution

The distribution of the engineering objects is described by the elements of the ObjectDistribution package, which defines seven nodes and seven channels. Nodes represent independent processing units (such as computers) and channels represent communication objects (such as computer networks). These elements are represented in figure 110, which provides an overview of the distribution of the system processing elements. Nodes contain capsules and clusters that represent groupings of elements for protection and migration management purposes, according to the policies and strategies defined for the system.
5.4 Engineering channels

All the channels have a similar structure, composed of a stub, a binder and a protocol object for each engineering object connected to the channel, and a common interceptor that provides all gateway functionality required when there is a need to cross any organizational, security, system management, naming, or protocol domain boundary between the connected objects.

Normally channels do not need to be specified in detail, because they are provided by the underlying middleware platform. However, there are situations in which their structure needs to be explicitly modeled to specify some channel properties or non-functional requirements on the channel elements, such as performance, QoS or security constraints. The internal structure of the channels used in the PhoneMob specification is depicted in figures 111, 112, 113 and 114.
Figure 111: Internal structure of channel PhoneMob GeneralChannel.

Figure 112: Internal structure of channel PhoneMob DedicatedChannel.
Figure 113: Internal structure of channel Stock Data Channel.

Figure 114: Internal structure of the four channels that specify the connection between the systems of the external providers and the PhoneMob main node.
6 Technology viewpoint specifications

6.1 Overall organization

The technology specifications focus on four main issues: identifying the main types of technology objects used to implement the system, describing the requirements on these objects in terms of implementable standards, stating the extra information for testing (IXIT) that needs to be associated with the technology objects and describing the relevant processes and activities involved in the provision, deployment, maintenance and evolution of the systems and its parts. In our example, the technology specifications are structured accordingly, with packages for each of these descriptions (see figure 115).

![Diagram of technology specifications]

Figure 115: The overall structure of the technology specification.

6.2 Technology objects

Figure 116 presents the overview of the node configuration for the system, showing the different types of technology objects involved.

6.3 IXIT

The implementation extra information for testing, or IXIT, is the additional information from the implementer that describes how the implementation is structured and how the tester is supposed to access the conformance points declared in the specification. Figure 117 shows IXIT defined for the objects of the PhoneMob system.

6.4 Implementable standards

Figure 118 shows the implementable standards that each technology object of the PhoneMob system should conform to. The list of the implementable standards used is shown in figure 119.

6.5 Technology processes

The technology viewpoint defines an implementation as the result of a process that instantiates a specification and whose validity can be subject to test. These processes may include development, system deployment plans, configuration guides, criteria for the selection of technology and of providers, change management plans, maintenance processes, and so on. In UML4ODP these processes are expressed using activity diagrams stereotyped «Technology view» «Technology_Spec». Figure 120 shows a technology process for the selection of a technology object.
Figure 116: Technology objects showing an overview of the node configuration.
The implementer shall state in the IXIT:
- required features and languages (support for HTML5, XHTML Mobile Profile, IPv6, etc.);
- List of capabilities/compatibilities with W3C standards including supported versions.

The implementer shall state in the IXIT:
- the maximum number of users that can be supported;
- – the predicted MTBF minimum for the system;
- – operating system supported;
- – hardware characteristics (CPU, RAM, ...);
- – browser types supported, including versions;
- – languages and platforms (e.g. JVM, .NET) supported, including versions.

The implementer shall state in the IXIT:
- the maximum number of users that can be supported;
- – thin or thick client technology;
- – types of browser supported, including versions;
- – languages/platform supported, including versions.

The implementer shall state in the IXIT:
- – types of servers supported in the LAN (web servers, mail servers, FTP servers, VoIP servers, DNS servers, ...) and supported versions.

The implementer shall state in the IXIT:
- – types of protocols supported.
- – speed.

The implementer shall state in the IXIT:
- – type of firewall supported (network or application layer)
- – network address translation supported

The implementer shall state in the IXIT:
- – types of protocols supported.
- – speed

The implementer shall state in the IXIT:
- – type of firewalls supported (Ethernet, WLAN, ...);
- – speed.

The implementer shall state in the IXIT:
- – list of industry standards supported;
- – max number of reads/second.

The implementer shall state in the IXIT:
- – the maximum number of users that can be supported;
- – how internationalization is supported and managed;
- – the minimum predicted mean time between failures (MTBF).

The implementer shall state in the IXIT:
- – how internationalization is supported and managed;
- – the minimum predicted mean time between failures (MTBF).

Figure 117: IXIT for the technology objects of the PhoneMob system.
Figure 118: Technology objects showing their associated implementable standards.
Figure 119: Implementable standards used in the PhoneMob system specification.

Figure 120: A TV_Implementation activity that selects a technology object.
7 Correspondences

The PhoneMob system specification defines six pairwise correspondences between the viewpoints (see figure 1). These are described in the following sections.

Correspondences between individual elements in two viewpoints are described by means a UML dependency stereotyped «CorrespondenceLink». Such dependencies represent in a concise way the UML4ODP correspondence classes, and allow better visual representations for them. A tool is responsible for generating these classes from the shorthand and vice versa. Note that in some of the diagrams that follow, the stereotypes «CorrespondenceLink» are not shown to improve the readability of the figures.

7.1 Correspondences between Enterprise and Information viewpoints

Figures 122 to 123 show the correspondences between the enterprise objects and roles, and their related information object types. The figures describe the correspondences defined for each community. Note the existence of correspondences between some associations, and the lack of correspondences between some enterprise elements (object, roles) for which there is no counterpart in the information viewpoint.

Figure 121: Correspondences between the enterprise objects and roles of the Phone Repair community and the related information object types.
There are also correspondences between the elements that model the behaviour of the system in the two viewpoints, either as interactions or as processes.

In the first place, figure 124 describes the correspondences between the Phone Repair community interactions and the related information elements. Then, figure 125 describes the correspondences between the processes of this community and their related information elements. Finally, figure 126 shows the same information, but expressed using UML4ODP correspondence classes.
Figure 124: Correspondences between the Phone Repair community interactions and the related information elements.

Figure 125: Correspondences between the Phone Repair community processes and the related information elements.
Figure 126: Correspondences between the Phone Repair community processes and the related information elements, expressed here as correspondence classes (same information as shown in figure 125).

Finally, figure 127 shows the correspondences between the Phone Repair community policies and the related information elements.

Figure 127: Correspondences between the Phone Repair policies and the related information elements.
7.2 Correspondences between Enterprise and Computational viewpoints

Figures 128 and 129 describe the correspondences specified between the enterprise objects and roles, and their related computational objects and data types. Figure 130 shows these correspondences using UML4ODP «CorrespondenceLink» classes.

Figure 128: Correspondences between the enterprise roles and the related computational objects.

Figure 129: Correspondences between the enterprise roles and the related computational data types.
There are also correspondences between the elements that model the behaviour of the system in the two viewpoints, either as interactions or as processes.

In the first place, figure 131 shows the correspondences between the Phone Repair community policies and the related information elements. Then, figure 132 describes the correspondences between the Phone Repair community interactions and the related information elements. Finally, figure 133 describes the correspondences between the processes of this community and their related information elements.
Figure 132: Correspondences between the Phone Repair community interactions and the related computational elements.
Figure 133: Correspondences between the Phone Repair community processes and the related computational elements.
Finally, figure 140 shows the correspondences defined between the enterprise and computational elements as UML4ODP «CorrespondenceLink» classes.

Figure 134: Correspondences between enterprise and computational elements expressed as «CorrespondenceLink» classes.

7.3 Correspondences between Enterprise and Engineering viewpoints

This kind of correspondences is normally used to specify some system requirements on the appropriate engineering elements, which need to fulfil particular requirements coming from the engineering specifications. In the first place, figure 135 shows the correspondences defined between the enterprise processes that deal with the negotiation of service level agreements (with the manufacturer and courier companies) and the engineering protocol objects in their corresponding external channels. These correspondences are of type require encryption, which is indicated by the appropriate constraint in the correspondence. Figure 136 shows these correspondences expressed as UML4ODP «CorrespondenceLink» classes.
A second group of correspondences determine the need to implement audit trace logs between specific enterprise interactions and the appropriate engineering interceptors. These correspondences, of type audit trace recorded by are shown in figure 137.

Figure 136: Correspondences between enterprise processes and protocol objects requiring encryption, expressed as «CorrespondenceLink» classes.

Figure 137: Correspondences between enterprise interactions and interceptors requiring audit trace logs.
7.4 Correspondences between Information and Computational viewpoints

Figure 138 shows the correspondences between the information object types and the computational data types. In this case there are no direct correspondences between information object types and computational object types. The latter encapsulate the functions that handle the computational data types, and therefore they are indirectly related by them (in the model, computational data types are related to the computational object types that encapsulate them by means of component realizations).

Figure 138: Correspondences between information object types and computational data types.
Then, figure 139 shows the correspondences between information object actions and computational interactions.

Figure 139: Correspondences between information object actions and computational interactions.
Figure 140: Correspondences between information and computational elements expressed as «CorrespondenceLink» classes.
7.5 Correspondences between Computational and Engineering viewpoints

The set of relationships between computational and engineering objects is shown in figure 141. Note that object names do not always coincide.

Figure 141: Correspondences between computational and engineering objects.
Figure 142 shows the one-to-one correspondences between the computational and engineering interface signatures.
The «CorrespondenceLink» classes that specify the correspondences between these computational and engineering elements are shown in figures 143 and 144 respectively.
Figure 144: Correspondences between computational and engineering interface signatures expressed as UML4ODP «CorrespondenceLink» classes.
7.6 Correspondences between Engineering and Technology viewpoints

Correspondences are used in this case to specify the choice of technology required to implement and deploy the engineering nodes and channels. This is illustrated in figures 145 and 146, where the correspondences represent usage relationships between the related elements.

Figure 145: Correspondences between engineering nodes and channels, and technology objects.
Figure 146: Correspondences between engineering and technology objects.
Finally, figure 147 shows the correspondences between engineering and technology elements expressed as UML4ODP «CorrespondenceLink» classes.
References
