Man Machine Interface in RFID Middleware: DEPCAS User Interface

Carlos Cerrada, Ismael Abad, José Antonio Cerrada and Ruben Heradio

Departamento de Ingeniería de Software y Sistemas Informáticos
Escuela Técnica Superior de Ingeniería Informática, UNED
C/ Juan del rosal 16, 28040 Madrid, Spain

Abstract. Between the RFID middleware main components the editing and management tools support the way to configure and deploy the RFID solutions. The most important functions of Man Machine Interface for RFID middleware component are to allow access to the monitoring, the management and the control included in the middleware. Among the key features that are necessary for this component we can mention the flexibility, adaptability, and scalability to the system middleware that implements it. We present a study on the interfaces in some of the existing RFID middleware. We also include the prototype of the user interface made for the RFID middleware architecture called DEPCAS (Data EPC Acquisition System). This interface called GUV (Graphical User Viewer) was developed through language EFL (Exemplar Flexibilization Language)

1 Introduction

The use of information in the RFID business applications is marked by the middleware exploits that allows the transfer of information from acquisition sensors to the application servers. The fundamental requirements of this type of middleware are related to the management of infrastructure devices reading, the processing information received from auto identification, the business needs to incorporate auto identification and sharing of information [1]. To address these requirements, software architectures that implement this type of middleware proposed a set of layers that specialize in solving different requirements. These layers are based primarily on the architecture proposed by EPCglobal organization[2], including the resolution of a protocol for reading between the middleware and the tag reading devices, called LLRP (Low Level Reader Protocol) and higher RP (Reader Protocol) according to the standards [3] [4], an event process from the reading of RFID tags [5], an assignment of semantic information to the object through an RFID identification service [6], and a service to receive or transmit information to business applications from the RFID middleware [7].

Existing middleware proposals solve, one way or another, all of these functions. For this purpose they use a set of common elements that allow the integration of functionalities such as: software architecture supporting a framework for

Cerrada C., Abad I., Cerrada J. and Heradio R. (2009).
Man Machine Interface in RFID Middleware: DEPCAS User Interface.
In *Proceedings of the 3rd International Workshop on RFID Technology - Concepts, Applications, Challenges* , pages 125-132
DOI: 10.5220/0002202201250132

developing, a service platform, an specification language or environment for the visual interface user.

Furthermore, the implementations include what is called generically management console that allows the access to certain RFID system settings and query of the information being handled by the modules that are part of the RFID middleware These graphical interfaces include functions as: allow the configuration of devices in the environment, show the overall performance of the middleware, locate and identify the tagged objects, display system alarms and events or access to consolidated information on the RFID system. In this article we focus on this software component that is usually included in every RFID middleware and we present the alternative in which we are working for the development of our proposed architecture called DEPCAS (Data EPC Acquisition System).

DEPCAS define software architecture for solving the RFID middleware based on system architecture for monitoring and control (SCADA) process. DEPCAS propose the scenario concept to generalize the process that can be performed from the acquisition of auto identification information. The processing of a scenario produces results that contain elements relevant to their semantic direct use in business applications. The graphical interface is called in DEPCAS GUV (Graphical User Viewer) and is designed to standardize access to the information managed in the middleware regardless of the scenario being driving. Henceforth this paper is structured as follows: Section 2 presents the user interfaces used in proposals for middleware, which functions are included and how they are integrated with other elements of the middleware, in Section 3 we presents the flexibilization language used for defining the graphical environment of 3 DEPCAS, in Section 4 we presents the architecture and how DEPCAS integrate the GUV environment, and finally the section 5 describes the prototype GUI GUV developed by applying the EFL concepts to the MMI development in DEPCAS.

2 The Man Machine Interface in RFID Middleware

The different RFID middleware proposals can be divided into commercial and academic (Table 1). Among the commercial proposals two types are included: those that correspond to large global software companies such as Sun MicroSystems, IBM, Microsoft, ORACLE, Sybase, Verisign or SAP, and the solutions provided from companies that are software (middleware, distribution, ..) specialists as RedPrairie, OatSystems, Vue Technologies, etc. Between the academic solutions we can mention the solutions from autoid like the AutoId center in the Massachusetts University (MIT), the StGallen AutoId Lab in St. Gallen University or the research on logistics such as the Los Angeles University or the Hong Kong University.

Table 1. RFID Middleware and MMI Components.

	Name	WebSphere RFID
Sun MicroS.	Sun Java RFID System v3.0 RFID Management Console	Sun Java RFID System v3.0 RFID Management Console
IBM	Web Sphere RFID Information Server	RFID Information Server Management
Microsoft	BizTalk RFID Solutions	BIZTalk RFID Monitor
ORACLE	ORACLE RFID And sensor based services	ORACLE Enterprise Manager
Sybase	RFID Anywhere	RFID Anywhere Manager Enterprise
OATSystems	C4 Architecture	C4 Architecture
VuewTech.	TrueVUE platform	TrueVue Site Manager
AutoID-MIT	MENTOR	
AutoID-StGallen	Fosstrak (previous Accada)	Reader Test Client
Chinese University of Hong Kong	CUHK RFID	CUHK Management Console
UCLA	WinRFID	WinRFID Management Console

The study of different RFID MMIs allows obtaining a set of characteristics that are implemented to solve the user interface in the RFID middleware. Between these features we include:

- Web Client
- Collection of dynamic information.
- Managing the database infrastructure of RFID.
- Managing the process database.
- Lists of information.
- Editing and management of graphical elements.
- Management and configuration of alarms.
- User management and privileges.

These features and functions define the operation of the user interface in RFID middleware. For example, the SunMicrosystems RFID middleware incorporates a client web-based interface called RFID Console Management which provides two basic options, one for administration and another for RFID monitoring information. In this interface there is no option to add new elements and the information displayed is stored in the database and organized according to the topology of the reading equipment that is currently on the system. Besides incorporating lists to access the event information system middleware with different filtering options. As an alternative approach, the middleware TrueVue Technologies incorporates TrueVue Manager with two options of operation in local mode TrueVue called Site Manager and corporate TrueVue called Enterprise Manager. Next table shows the cross results between existing RFID middleware implementation and the MMI set of characteristics presented in this section:

Table 2. Middleware RFID/MMI Characteristics.

	Sun	IBM	Sybase	VueTech	MENTOR	FOSSTRAK	WinRFID
Web Client	Yes	Yes	No	No	No	Yes	No
Dynamic Information	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Configurable Dynamic Information	No	Yes	Yes	Yes	No	No	Yes
RFID Database Management	Yes	Yes	Yes	Yes	No	Yes	Yes
RFID process configuration	No	Yes	Yes	Yes	No	No	Yes
Lists	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Graphical edition	No	No	Yes	Yes	No	No	No
Alarms	No	Yes	Yes	Yes	No	No	Yes
Events	Yes	Yes	Yes	Yes	Yes	Yes	Yes
User Management	Yes	Yes	No	No	No	Yes	No

3 Overview of the EFL (Exemplar Flexibilization Language)

This section introduces the main characteristics of EFL (Exemplar Flexibilization Language) [8]. A technique for developing a DSL interpreter quickly is embedding it into a dynamic general purpose language [9]. This way, all the host language capabilities are implicitly available from the DSL. Unfortunately, the pay-off is that the DSL concrete syntax has to fit in the host language concrete syntax. EFL is currently implemented applying this technique: it is a library of the Ruby object oriented language. As we will see, thanks to the Ruby extensibility, the EFL concrete syntax is reasonably usable.

3.1 Defining Generators

Figure 1 shows a simplified EFL metamodel. EFL supports the writing of generators that transform input exemplar files into output final product files according to input DSL specifications. EFL generators are written as Ruby classes that extend from the Generator class

This way, the generators can be easily reused by mean of the Ruby composition and inheritance capabilities. Alternatively, there is available the next syntactic sugar to write generators as objects of the Generator class:

```
my_generator = generator {<<generator definition>>}
```

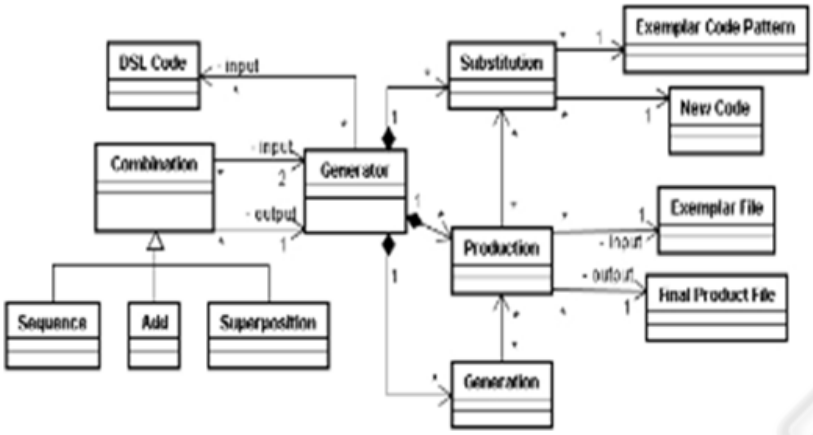

Fig. 1. TrueVue Site Management.

One generator definition is composed of substitutions, productions and generations:

1.- A substitution describes the interchange of an exemplar code pattern, expressed with a regular expression3, to new code. Crosscutting generators often apply the same substitutions over different exemplar files. To avoid the repetitive writing of substitutions and support their reuse, substitutions are independent from the exemplar files and the final product files. The main Generator methods to define substitutions are: sub and gsub. A local substitution (sub) expresses the interchange of the first occurrence of the reg_exp regular expression to the text string. A global substitution (gsub) expresses the interchange of all the reg_exp occurrences.

2.- A production describes the application of a substitutionlist to an exemplar file to produce a final product file. Generator provides the next method to define productions. EFL supports the detection of undesirable overlaps among the code patterns of the sub_list substitutions.

3.- A generation executes a list of productions. Generator provides the next method for generations. EFL supports the detection of undesirable collisions among the productions of a generation.

3.2 Combining Generators

For writing complex exemplar transformations, EFL provides the next binary operators to combine two generators g1 and g2:

- Sequence. Executes g1 first and g2 later:
- Add. Returns a new generator which substitutions and productions are the union of the substitutions and productions of g1 and g2:
- Superposition. Updates the substitutions and productions of g1 with the substitutions and productions of g2. Those with the same name are overwritten and the remaining ones are added:

4 GUV (Graphical User Viewer) in DEPCAS

DEPCAS (EPC Data Acquisition System) is a proposed architecture for RFID middleware systems dedicated to data acquisition in real and heterogeneous environments. The scheme proposed is based on the supervisory and control systems known as SCADA (Supervisory Control and Data Acquisition).. This central acquisition RID system is the proposal of DEPCAS architecture (Figure 2).

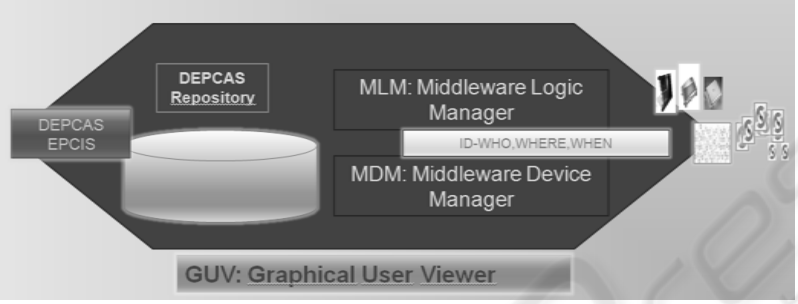

Fig. 2. DEPCAS Architecture.

The basic structure of DEPCAS is divided into four sub-systems: the MDM (middleware device manager) for the management of infrastructure, MLM (middleware logic manager) for auto identification process, the GUV (graphical user viewer) or man-machine interface, and finally the EPCIS (EPC Information System) as software component to communicate with other systems.

The main objectives of GUV are:

- to include a single common environment and information management in the middleware. This environment is common to all management information system configuration data regardless of being related to infrastructure, the processing logic sets or walkways with information modules EPCIS
- a common graphical object to support list information. This list can be used like dynamic queries with real time information or static queries to show database information.
- information about a particular object in real time. You may include in new graphics another graphical object with real time attributes.
- an alarm system and alarm processing operations that can be configured according the domain what is used. Treatment includes alarm management levels alarm and the realization of automated operations.
- the treatment process and system events for the acquisition of auto identification data.
- the management of graphical objects that are used and which allows their reuse in different graphical interfaces.

Given the characteristics of DEPCAS architecture, the GUI should be able to be adapted to the environment that uses the middleware. It is necessary to have the graphical tools that allow creating and editing the specific configuration of the domain in which uses the environment. To achieve this we have used an approximation based on generative programming called Exemplar Flexibilization Language. This

approach exploits the flexibility of examples to build the user interfaces based on the basic issues that define the GUI adapted to each particular domain. Under these conditions, the graphical environment of DEPCAS is made up of two fundamental elements: the graphical running environment, called GUV, is the result of a generation from items handled by a graphic editing environment that we call ediGUV. The editing environment is intended to handle the language domain. This environment allows specifying from the individual copies adjusted to certain systems. The key features of the design of the editing environment allows the GUV provide additional flexibility and customization for the graphic development, including a set of graphical objects such as buttons, real time links, data, charts, summaries of alarms, etc.. which can be adapted to the domain specific needs.

5 GUV Prototype Implementation

Based on the concepts of high-level design have been shown we have developed a GUV prototype environment that allows building graphical environment from specific domain. The prototype has a graphic editing system that allows creation, modification and deletion of graphical elements. This environment ediGUV define a core container referred as DEPCASGUV in which may include other edited elements. The prototype operates two types of graphic elements: the basic elements that are provided directly by the window SDKs that is in use. And the compound elements of DEPCAS that are the result of basic elements compositions for a specific function in the DEPCAS. The basic graphic elements in the original DEPCAS are: windows, buttons edit fields , text fields, check boxes, combo boxes. DEPCAS graphic elements are: panel database (Add-Delete-Modify Data), list of dynamic database, list of static database, RFID / EPC Label, RFID Reader Point, Dynamic graph (Route / Forecast)

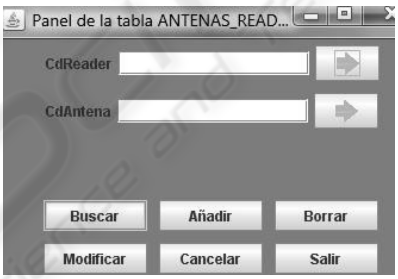

Fig 3. Database panel example from DEPCAS GUV.

Each of the above elements is defined by his example to get your specification to generate a corresponding graphical environment. For example, the following panel specification database (add-delete-modify) is instantiated as shown in figure 3. This panel includes all the fields in the table relates the antennas with the teams and readers solve all operations to find add, delete or modify information in database. The prototype has been developed that includes the use of a database in MySQL and graphical elements are using the basic elements included in the JAVA AWT library.

6 Conclusions

Regarding the development of GUI middleware for RFID, there is little alternatives. Most of the work developed in these middleware focuses on resolution of academic or commercial layers from the proposed standards. Only some of the proposals have focused on solving tasks related to the user interfaces for this type of middleware.

With regard to the proposed development of a graphical environment using an adaptive generative approach based on the flexibility of predefined charts that copies can be adapted and combined we can say that this is an adequate approximation for the resolution of the basic tasks to be solved in the desired graph DEPCAS architecture. About future work there are different issues to be resolved. First, to extend the graphical elements in GUV with more modern graphic elements such as tabs, transparent data, etc. It is also required extensive work to enable us to provide an environment WYSIWYG (what you see is what you get) integrated with the generation-based EFL and its operators. Additionally, DEPCAS evolution demands new graphics objects to those currently employed.

Acknowledgements

This research has been carried out under contract with the Spanish CICYT through the DPI2005-03769 and DPI2007-61287 projects.

References

1. C. Floerkemeier et al., RFID Middleware Design -Addressing Application Requirements and RFID. Proceedings of SOC-EUSAI (2005)
2. EPC Global Inc., The EPC Global Architecture Framework v 1.2. (2007)
3. EPC Global Inc., The RP Standard v. 1.1. (2006)
4. EPC Global Inc., The LLRP Standard v. 1.2. (2007)
5. EPC Glocal Inc. The ALE Standard v.1.1. (2008)
6. EPC Global Inc. The ONS Standard v.1.0.1. (2008)
7. EPC Global Inc. The EPCIS Standard v.1.0.1. (2007)
8. Heradio Gil, R. Metodologia de desarrollo de software basada en el paradigma generativo. Realizacion mediante la transformacion de ejemplares. Ph. D. Thesis, Departamento de Ingenieria de Software y Sistemas Informaticos de la UNED, Spain (2007)
9. Fowler, M. Language Workbenches: The Killer-App for Domain Specific LanguagesURL: http://www.martinfowler.com/articles/languageWorkbench.html (2005)