The Journal of Systems and Software xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Development of Java based RFID application programmable interface for heterogeneous RFID system

Mohammed F.M. Ali*, Mohammed I. Younis, Kamal Z. Zamli, Widad Ismail

School of Electrical & Electronic Engineering, University of Science Malaysia, Engineering Kampus, 14300 Nibong Tebal, Penang, Malaysia

ARTICLE INFO

Article history:
Received 7 April 2010
Received in revised form 4 June 2010
Accepted 8 July 2010
Available online xxx

Keywords:
RFID
Interoperability
API
Sequence diagram
Tracking
Monitoring
Multithreading

ABSTRACT

Developing RFID based applications is a painstakingly difficult endeavor. The difficulties include non-standard software and hardware peripherals from vendors, interoperability problems between different operating systems as well as lack of expertise in terms of low-level programming for RFID (i.e. steep learning curve). In order to address these difficulties, a reusable RFIDTM API (*RFID Tracking & Monitoring Application Programmable Interface*) for heterogeneous RFID system has been designed and implemented. The API has been successfully employed in a number of application prototypes including tracking of inventories as well as human/object tracking and tagging. Here, the module has been tested on a number of different types and configuration of active and passive readers including that LF and UHF Readers.

1. Introduction

Since ancient times, humans have quested for the improvement of their quality of life. Solutions were sought. They invented the wheel and developed other tools to enhance and facilitate daily work such as in agriculture, irrigation, and trade. With the passage of time, new requirements have emerged. Factories, laboratories, and advanced technology proliferated to meet human needs. Inevitably, the increasing populations, together with the increasing human needs, have further led to the spread of business and industries. At this day and age, and as a result of amplified burdens, humans have once again begun to look for ways to reduce such complexities in the field of information exchange and data transfer, for example, among the branches of companies, factories, warehouses, and across suppliers and recipients. Increased complexity has also been noted in the tracking of goods and document information. These reasons have led humans to use modern technology, of which at present is deemed the prime solution for humans' problems. In addition, it has been well accepted that technological evolution makes life easier. Developments in electronics and *Radio Frequency Identification* (RFID) can even be considered one of the most wondrous developments on the basis that it supports human life in many aspects. Modern (Intel, 2005; Srivastava, 2005) technologies have changed the ways in doing business and work, not only because of their apparent modernity, but also because these technologies keep up with humans' comfortability. Hence, this modernizing has an important role in the style of our life! The indoor tracking and automatic management for information can possibly develop given its relation to ensuring smoother businesses (Huifang and Junbin, 2009), apart from expanding its general purpose of tracking and monitoring both indoor and outdoor. There is one technology has achieved that such: Radio Frequency Identification (RFID).

Nowadays the access to information easily, without delay and any complication has been considered one of the important aims in many fields. RFID is one of the best solutions for this aim because RFID is supporting real-time access, fast and smart detection for targets (tags), and has the ability for long-range area coverage reading. In fact, the real-time availability of RFID information is deemed critical for many RFID applications such as in process control systems, manufacturing automation systems, critical applications, and coupling of active transponders for GPS and supply chains (Wang et al., 2005; IEEE, 2009). In short, RFID can improve operations and solve very complex and time sensitive problems. There are many applications for the RFID system, and the more dominant applications include the following: supply chain management, inventory tracking, access control, library book checkout, cattle tracking, and passport tagging (Scassa et al., 2005). Some hospitals even use RFID to provide real-time tracking to locate doctors and nurses within the hospital and to track a patient's location and sets of equip-

* Corresponding author. Tel.: +60 135135561.
E-mail address: mofmah2009@gmail.com (M.F.M. Ali).

0164-1212/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2010.07.030

Fig. 1. RFID Tag inside a living body.

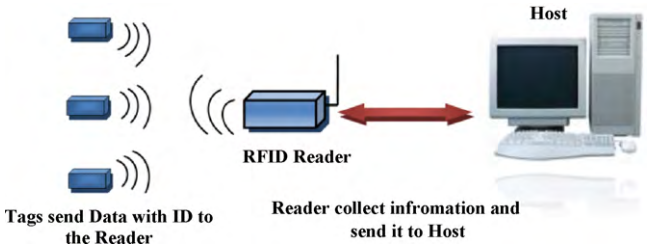

Fig. 2. RFID System Components.

ment (Supply Insight, 2006). The development in RFID has become beyond measure. Imagine people from the near future driving cars, opening doors, entering houses, and using various devices without the use of keys or passwords. All those have become possible with the use of RFID devices. For example, a small tag implants inside a living body like an animal or a human being as depicted in Fig. 1 (Kranenburg, 2006).

The RFID refers to small electronic devices consisting of tags, readers, and software. The RFID System Components are shown in Fig. 2. Tags are small transponders that respond to queries from an RFID reader by wirelessly transmitting a unique identifier (BITKOM, 2005; Jackson, 2004). These RFID tags are categorized as either passive, semi-active or active (Intermec, 2007; BITKOM, 2005). They are usually capable of carrying kilobytes of data, and each tag provides a unique identifier by which a reader can recognize between tags (Bolotnyy and Robins, 2007). An RFID reader is a device used to collect information from tags and send data to computers which have the software application. The software for RFID system can be divided into two parts: application programming interface (API) and application software.

Defined loosely, RFID API itself is used to develop middleware works, because middleware can be seen as a level layer higher than API (Shien-Chiang, 2007). Here the API (Middleware) will response to fetching information (Ajana et al., 2009), which is captured by the reader from the tags and then filtered and sent to the end-user, which consequently processes data using a software application. Actually the purpose of APIs is to support and increase the reusability of codes using existing software artifacts or knowledge to create new software.

Despite ease of use, there still remain obstacles for users and developers, which include diversities in applications and hardware. Indeed, Non-standard hardware and software implementation does not have the positive impacts of RFID for many users. On the other hand, with RFID's popularization and wide usage, development should then focus on software applications. It is worth noting that developers can build several software applications using a single RFID product. However, the real problem stems from the fact how to widely support libraries or identify flexible API that can support heterogeneous RFID Readers and different connection interfaces on heterogeneous operating system. Moreover, support is also needed to accommodate a diversity of users and their different applications. As such, using API for RFID purposes, the developer can develop numerous applications software depending on the need of one engine. In short, if an RFID system is to be made pervasively available, there is a need for a flexible API design that promotes reusability and redundancy, thus avoiding time wastage, as well as supporting interoperability within different operating systems and hardware implementations. This specific requirement has been highlighted amid the high prevalence of RFID use and increased demand in many modern applications. Finally, both developers and users opt for products that can adapt in different environments. This raises the demand for independency and cross-functionality of products.

Motivated by such aims and challenges, this paper proposes a new RFID API which is called *Radio Frequency Identification for Tracking and Monitoring* (RFIDTM); this API can dynamically and seamlessly support different operating systems, different RFID systems, and different connection interfaces. It aims to mind the gap currently filled by third-party programs and bridge the gap between industry and academic researchers by providing a shortcut development cycle. Such APIs enable both programmers and researchers in both hardware and software domains to take vital advantages of API.

The rest of this paper is organized as follows. Section 2 highlights the state of the art for developing RFID based APIs. Section 3 gives the architecture design and design criteria for the proposed RFIDTM library. Section 4 highlights the core classes and implementation issues. Section 5 gives the general evaluation of RFIDTM. Section 6 stresses and demonstrates the usage of the proposed APIs in developing: a network-based RFID system for monitoring human attendance, RFID Distributed Database Management system, RFIDTM Protection System as an anti-thief system, Contact less RFIDTM system, and RFID Terminal. These APIs have been used as a starting point in building all of these application programs using different readers (as proof of concepts). Section 7 states the lessons learned from the case studies. Finally, Section 8 states our conclusion and suggestions for future work.

2. Related work

There are some middleware available in the commercial and research domain which try to provide an environment for the application development there have been some proposals and research work involving middleware design and RFID data processing. Some of middleware design to support one reader and other to support different readers. However, there are a few proposals and research work involving middleware design and RFID data processing. In this paper, some of RFID middleware are also illustrated as they behave like APIs. The RFID Anywhere, a subsidiary of Sybase (Sybase, 2008), also provides RFID API. It integrates business logic and processes with a variety of automatic data collection and sensor technologies, including different readers (passive and active RFID readers), RFID printers, barcode scanners, mobile devices, location tracking systems, environmental sensors, and feedback mechanisms (Sybase, 2008). The RFID Anywhere includes a Microsoft Visual Studio .NET Extension that creates custom business modules. These business

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

modules can process data and integrate with existing enterprise applications. With this extension, RFID Anywhere automatically generates C# code containing all of the necessary environmental references, allowing developers to focus on core business logic. The framework is tightly integrated with Visual Studio .NET to provide an easy-to-use and compact environment. This integration allows developers to use a set of pre-defined templates and wizards to speed up the development process, and to incorporate newly created components with some standard functionality (Solutions and Sybase, 2008). As such, developers and integrators can focus on writing business logic and avoid low-level hardware interfaces. The RFID Anywhere is positioned as an intelligent sensor network management system that integrates RFID and sensor technologies with business logic to enable the development, deployment, configuration, and maintenance of distributed intelligent sensor networks.

The WinRFID (Prabhu et al., 2005) is developed at the University of California Los Angeles (UCLA). This middleware enables rapid RFID application development and uses web services developed on Microsoft .NET framework. The WinRFID has a set of application programming and integration interfaces for supporting application development with provision of location services and sensors. Additionally, WinRFID also supports simultaneous distributed working of readers and tags at different frequencies using different protocols. The architecture of this middleware consists of five main layers. The first layer is the physical layer which deals with the hardware side like readers, tags and other sensors. The second layer is the protocol layer abstracts the reader-tag protocols. Above that lies the data processing layer, which is the third layer that deals with processing the data streams generated by the reader network. The fourth layer is the XML framework layer. Finally, the data presentation layer which presents information as per the requirements of end-users or different application requirements. WinRFID has limited for support cross-platform applications. Savant is a middleware developed by Auto-ID to act as middleware layer between RFID reader and databases (Glasser et al., 2007). In fact, Savant sits between tag readers and enterprise applications in order to manage the information retrieved from the tags. This middleware is also capable to collect, accumulate, and processes tag information from several RFID readers. The architecture of this middleware consists of three key elements (Ishikawa et al., 2003). The first element is the Event Management System (EMS) which provides a set of Java API for communication. The second element is real-time in-memory data structure (RIED) that manages event information generated by Tag reads. The final element is the Task Management System (TMS) which manages the tasks of processing the tag. While this middleware supports connection with different types of readers, it has limited built-in functionality for dealing with all types of sensor devices and providing data dissemination, filtering, and aggregation. Savant has a hierarchical architecture that directs the flow of data by gathering, storing, and acting on information and communicating with other Savants. In a Savant system, lower level Savants process, filter and direct information to the higher level ones. Consequently, massive flow of information and network traffic is reduced. The FlexRFID is developed at Alakhawayn University in Ifrane Morocco (Ajana et al., 2009). This middleware design provides the applications with a device neutral interface to communicate simultaneously with many different hardware devices, support different connection interfaces (USB, UTP, and RS-232), filter raw information and send the final useful information to end-user in order to enable an intelligent RFID network. It also provides an interface to access the hardware for the management and monitoring purposes. The FlexRFID provides all data processing capabilities along with the security and privacy features included in the data processing layer and enforced by a policy based management module for the business events, referred to as the Business Rules layer. The modular and layered design of FlexRFID allows integration of new features with little effort. The design also permits seamless integration of different types of enterprise applications. The FlexRFID middleware architecture is organized as a three-tier architecture consisting of backend applications layer, FlexRFID middleware layer, and hardware layer consisting of diverse types of sensors and devices (Ajana et al., 2009).

In fact, cross-platform support is considered important, as this feature allows software to run identically across different operating systems. The APIs designed by Java language can support cross-platform and even TCP/IP. With these two facilities, APIs not just support distributed monitoring through TCP/IP, they can also support distributed monitoring through TCP/IP for heterogeneous operating systems in the network. Each of Sybase, WinRFID, and FlexRFID provide no support for cross-platform functionality because it is designed based on Microsoft .NET.

Reusability is also considered a very important feature. It reduces the development cycle, a major concern of most developers, and supports easy development for applications. Reusability can helps users/developers, and it can provide specific support for them as APIs can adopt open architecture that can be extended horizontally (e.g., by adding functionalities) and vertically (e.g., by supporting different vendors). Here each of Sybase, Savant, WinRFID, and FlexRFID do not support an open architecture that can be extended both (horizontally and vertically). Also, they do not support shorten development cycle with highly reusable and reconfigurable API, limited providing only specific type of support. Here the specific support refers to the potential of users to get final information from readers after filtering and processing, and use that information in their application without using customized filtration. In the first place, all previous APIs have mentioned “readers” and “different types of readers,” but there has been no clear information about the exact meaning of “different types,” or whether they involve APIs communicating with other APIs.

Indeed, there are passive and active readers that work as a client (e.g., USM UHF Readers) or as a server (e.g., RF Code Mantis II Reader), so the meaning of supporting different readers should cover the real meaning for this term. In these instances, the ability to communicate with other APIs is identified as an important feature. As all previous APIs are designed to support raw information, processing that information before sending is useful to users. Sometimes, readers have APIs coupled with the *digital signal processing* (DSP) (e.g., RF Code Mantis II Reader). Thus, users need to send special commands to get that useful information. In some cases, processing is done inside the reader, and the final information is sent to the user. In other cases, users need to undergo steps to connect with the reader and perform custom filtrations on the received information. As such, supporting communication with other APIs and customizing the filtering of information is one avenue for support to users and developers.

Furthermore, readers have multiple connection interfaces (e.g., RS-232, USB, and Ethernet). In critical applications, redundancy is very important. Supporting reads from different connection interfaces for the same reader increases the read reliability by increasing the redundancy. Building from earlier approaches, this research proposes to combine the features from these different approaches with the aim of supporting researchers in the RFID field, and thus give developers and users the maximum assistance in ensuring easier ways to communicate and build their applications. The main aim of this research is to design, produce, and evaluate the new Application Programmable Interface (API) called RFIDTM (RFID Tracking and Monitoring), which can solve some of the perceived deficiencies in the current RFID systems.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

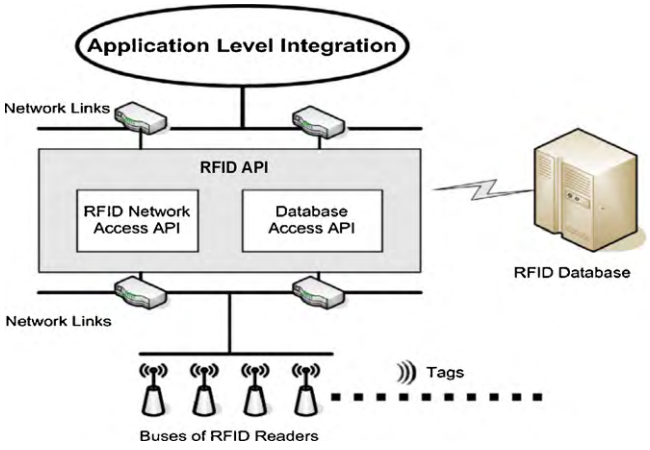

Fig. 3. RFIDTM API architectural design.

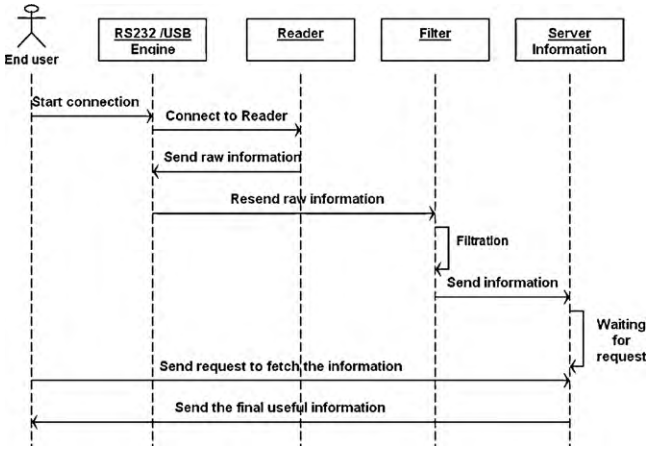

Fig. 4. RS232/USB sequence diagram.

3. RFIDTM architectural design

The design criteria for developing RFIDTM are stated as follows.

- To develop RFIDTM with an open architecture that can be extended horizontally (by adding functionality) and vertically (supporting different vendors or providers).
- To ensure that the RFIDTM supports heterogeneous RFID system consisting of heterogeneous RFID readers and operating system.
- To evaluate the applicability of the RFIDTM for RFID system development.

Herein, RFID API serves as the interface from the application level to achieve system level connectivity to (possibly) buses of readers as well as RFID databases. Viewing from RFID system perspective, the API is illustrated in Fig. 3.

With the large number of readers including differentiations in types of hardware and APIs, there is a need to solve the problem (or at least reduce the limitation) of communication with RFID systems. As such, we shall delve deeply into "trying to find solutions", and suggest and implement the best design solution for RFIDTM. An RFIDTM is targeted to be an API designed to work on a PC and not on chips or on hardware (reader). In this case, there is an expected reduction in hardware cost and complexity. An RFIDTM is also aimed to support communication with heterogeneous RFID systems through the support of different readers (passive and active), frequencies (UHF and LF), connection interfaces (i.e., RS-232, UTP, and USB), ways of communication (supporting client reader or server reader), types of information (raw and non-raw), and real-time reading. As such, RFIDTM intrinsically has a robust and flexible design. The RFIDTM must allow developers to communicate with different types of readers in real time (with or without integrated API); the building of applications can be done without any use of third-party programs for RFIDTM. Herein, RFIDTM supports the developer with real-time useful information, which means that it can avoid time wastage and reduce memory usage. In order to highlight how the requirements of RFIDTM fits into its design, the following high level sequence diagrams highlights the possible interactions between objects involved in a typical RFID system. Fig. 4 represents the sequence diagram for RS232 and USB connection interface. Here, the sequence diagram involves 4 classes (i.e. RS232/USB Engine, Reader, Filter, and ServerInformation). The RS-232/USB Engine class response to communicate with both RS-232 and USB by enable the developer to select the type of connection.

Fig. 5 represents the sequence diagram for UTP interface to the reader at the client. Here, the sequence diagram involves 4 classes (i.e. UTPServerEngine, Reader, Filter, and ServerInformation).

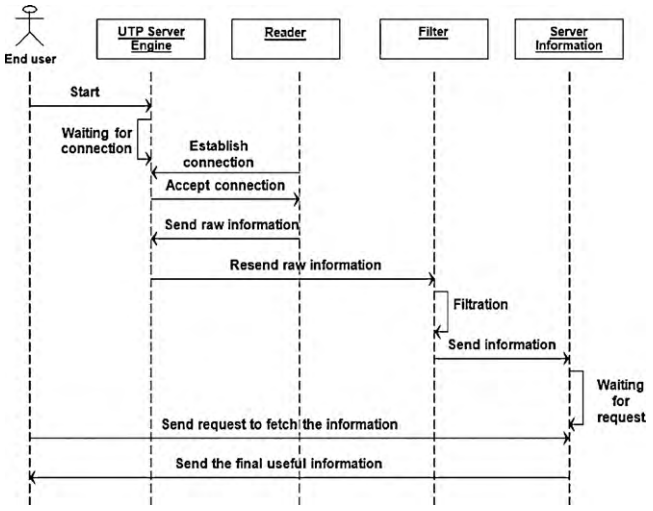

Fig. 5. UTP server sequence diagram.

Fig. 6 represents the sequence diagram for UTP interface to the server. Here, the sequence diagram involves 4 classes (i.e. UTPClientEngine, Reader, Filter, and ServerInformation).

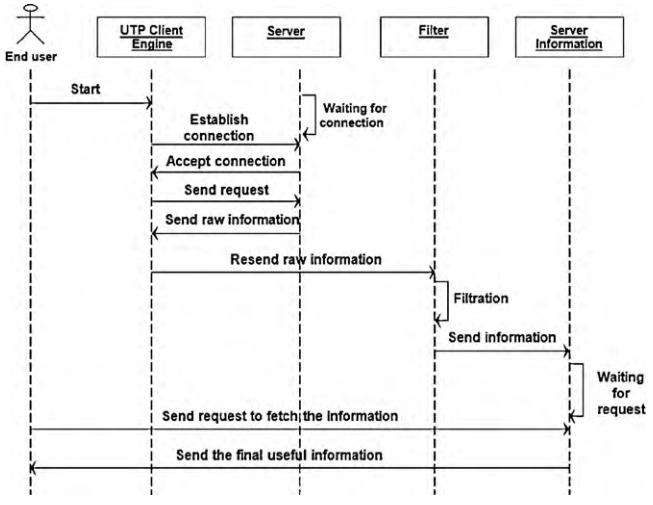

Fig. 6. UTP client sequence diagram.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

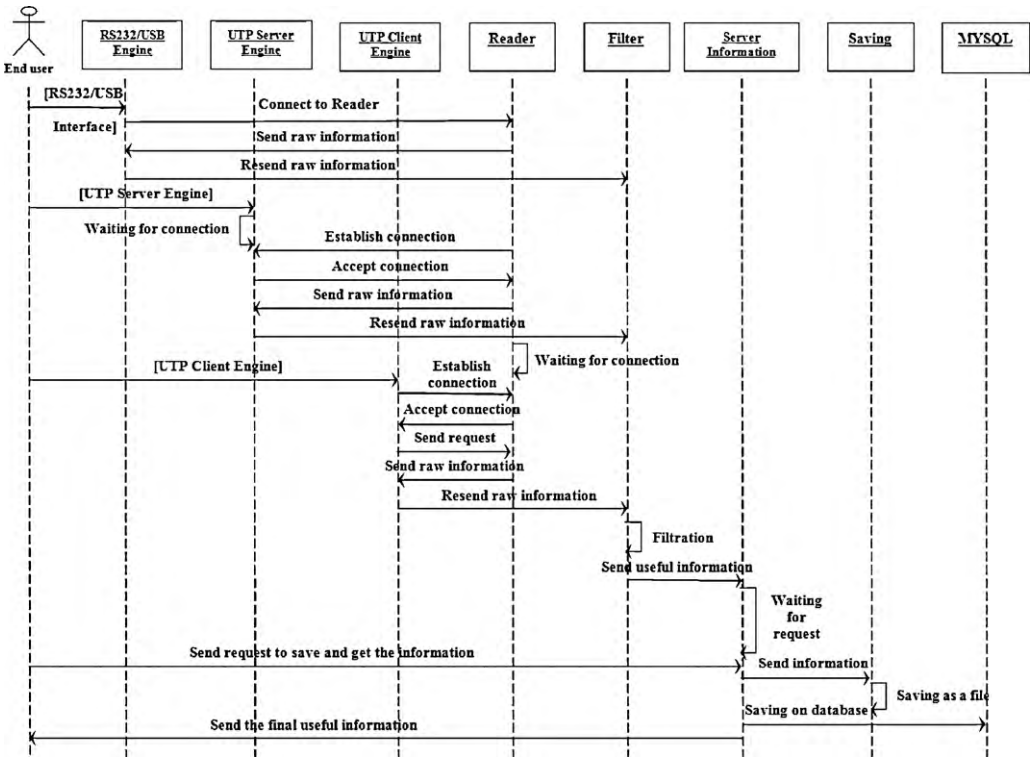

Fig. 7. RFIDTM overall usage sequence diagram.

In general, there is another class response to save final useful information both locally and remotely as a file or in a database. Fig. 7 shows the sequence diagrams for the overall usage of RFIDTM. Here, the sequence diagram involves 8 classes (i.e. RS232/USB Engine, UTPServerEngine, UTPClientEngine, Reader, Filter, and ServerInformation, Saving, and MYSQL classes).

As will be seen later, all these aforementioned classes will make up the complete RFIDTM with the exception of "Reader" class. It should be noted that the reader class is physical reader object and not actual class implementation. RFIDTM adopts Java in order to provide APIs platform-neutral.

Overall, RFIDTM is designed to support heterogeneous RFID systems (i.e. implemented in portable programming language and with standard database support), and with lower cost and stronger supports. To understand RFIDTM more deeply, this paper will explain ideal RFIDTM designs, features, and characteristics. An RFIDTM should acquire a new and simplified design that can promote reusability, redundancy, as well as having the ability to support interoperability within different operating systems and hardware implementations.

4. RFIDTM core classes

The main idea advocated in RFIDTM design is the use of a core with a functionality that can connect with readers and follow standardized communication methods (e.g., TCP/IP and RS-232). After connecting with readers, this core can communicate with readers and fetch information. The core can also communicate with different types of readers and with other integrated APIs, and can fetch information after conducting additional processes such as automatically calling for suitable special commands that require getting information from readers. Next, all information will be sent to the developer, who can utilize other functionalities provided by RFIDTM. An RFIDTM consists of many classes, with each class designed to support special functions, although some of these are designed to act as shortcuts for developers to support their applications. There are three main classes considered as the core for RFIDTM: RS-232/USB Engine, UTP Server Engine, and UTP Client Engine (see Fig. 8).

The RFIDTM uses two kinds of client server models and multiple threads. The first one represents two different engines for the RFIDTM (UTP Server Engine and UTP Client Engine), as shown in Fig. 8. The RFIDTM uses the first client server to communicate with different types of readers and fetch information. This model consists of a server/client considered as the core of the RFIDTM, and a client/server that represents the reader device. The second client–server model consists of a server present in each engine in the core of RFIDTM and the client which is used in the application

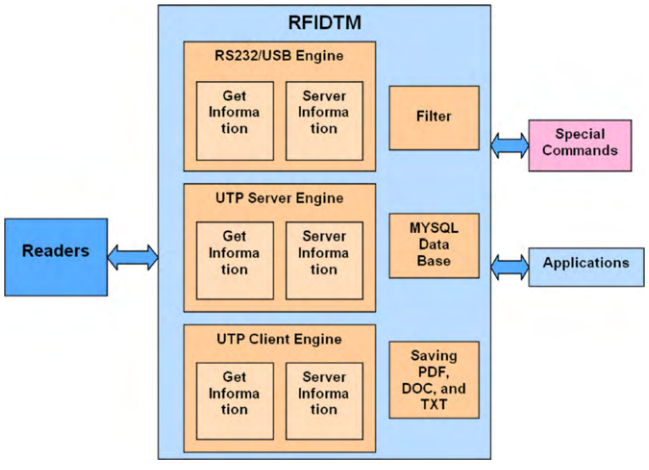

Fig. 8. RFIDTM design.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

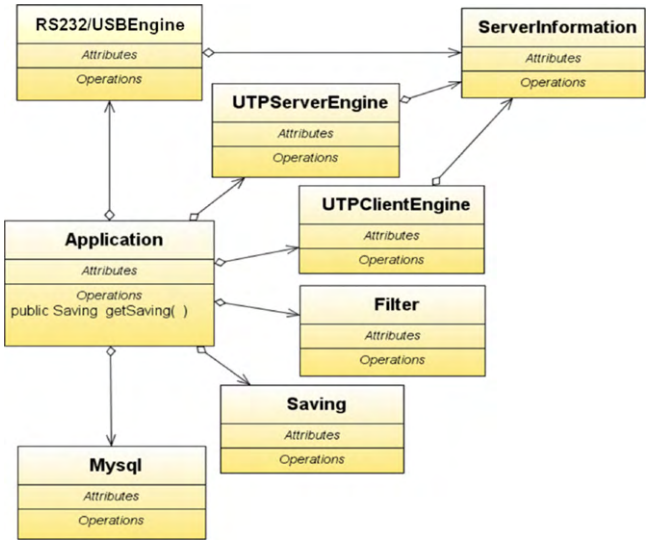

Fig. 9. RFIDTM API's class diagram.

part. When the core communicates with the reader and fetches the information using the first model, this information is directly sent to the server information (the second model) in real time. During this period, the server information waits for a request from the client via a socket interface. Then, when the application client initiates a communication session with the server information, the server sends the information to the application client. Thus, RFIDTM allows flexibility as the client can be designed independently on a remote host. In such cases, RFIDTM can be considered as a node in a distributed system over the network. The RFIDTM creates a socket and listens for incoming client connections on the TCP port. The port number is specified by the developer, and there is no fixed number used. This facility provides an advantage for RFIDTM by preventing the interruption from other application using the same port numbers.

In addition, RFIDTM uses a multiple threads which allow two parts of the same program to run concurrently. Multithreading increases the efficiency of a resource by running multiple tasks at the same time. In this case, the RFIDTM can support many different readers and handle various actions simultaneously. This also gives advantage to the RFIDTM in terms of supporting redundancy. In the case of connection to multi-readers, a thread is created upon connection with the first reader; in turn, the created thread constantly listens to the first reader. With each new connection to the same reader (in cases using different interface connections) or to different readers, a new thread is then created. As such, using a multiple thread has an advantage on multi-core CPU. Fig. 9 depicts the class diagram of the RFIDTM. This class diagram facilitates the understanding of the relationships between RFIDTM classes.

5. Evaluation and discussion

The RFIDTM has many important features, An RFIDTM is an ***open architecture*** API, which allows for future upgrades. It can be extended horizontally by (adding functionalities) and vertically (by supporting different vendors). In this case, RFIDTM can support the future applications needed by developers. For example, RFIDTM supports RS-232, USB and UTP connections with open architecture features so that developers can easily add a mobile connection class to RFIDTM API or new connection interfaces. In other words, open architecture software can be considered as a flexible approach to decentralized software evolution. Hence, any developer can design a new class and add that class to RFIDTM. Because the implementation dwells on portable language, that is, Java programming language, RFIDTM also has the ability to support numerous other operating systems (e.g. Linux, Macintosh, and Windows).

Actually RFIDTM is ***flexible***; here the flexibility of RFIDTM gives it the ability to modify software applications according to new requirements. Changes in functionality can be made either by the developer, they can develop their own cover by adding their desired components in the system (i.e., develop their own engine) or by changing the software architecture of the application. Since RFIDTM provides features such as standard protocol communication, filtration, and database access, minimal changes are required when developing complete RFID applications. In addition, an RFIDTM is a ***reusable*** API such that it gives advantage to the developers to reduce implementation time by shortening development cycle with highly reusable and reconfigured API; increases the likelihood that prior testing and use has eliminated errors or bugs; reduces development cost; reduces effort needed to build new software applications; and localizes code modifications when a change in implementation is required. Therefore, RFIDTM is a collection of pre-defined classes grouped into package. Indeed, an RFIDTM based application concurrently reads and detects from three different connection types (i.e., RS-232, USB and UTP). It can potentially enhance the reliability of the applications (i.e., through ***redundancy***) whenever the system is adopted. On one hand, redundancy of reading helps improve the availability and accessibility of the implemented system, especially when multiple readers are employed. On the other hand, RFIDTM is capable of supporting information from one system to different vendors. In fact, RFIDTM is ***Object Oriented*** (OO), when generalized; the application can be extended through new components or reused using some existing components. While this approach appears appealing, care must be taken to avoid increasing the complexity of the software application. Class hierarchy, which is designed for reuse, must be built with caution such that it does not create complexities. In fact, building shallow hierarchies without deep trees can make the software application more understandable. Moreover, ***cross-platform*** is considered as one of the important features of RFIDTM API. It can fully support heterogeneous operating system and RFID system over networks or the Internet. Through this, it does not matter what the operating system the computer works with or what the RFID API design uses. What matters is gaining connection using RFIDTM API in a distributed computer and network. Finally, RFIDTM has capability to communicate, execute programs, or transfer data among various functional units in a way that requires the user to have no familiarity about the unique characteristics of those units. In this way, RFIDTM supports the ***interoperability***. In short, RFIDTM has the following capabilities:

- Provides high level and highly customizable API, hides the internal difficulties of low-level RFID programming, and gives the simpler RFID API to deal with.
- Supports general network-based applications by supporting heterogeneous readers, heterogeneous operating system as well as enable different connection interfaces.
- Facilitates interoperability and reusability between non-homogeneous hardware and operating systems.
- Supports multi-readers at the same time (sequentially), hence, useful for highly available system, and allows user to monitor different systems in different locations from the same computer.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

Fig. 10. Networked based RFIDTM system.

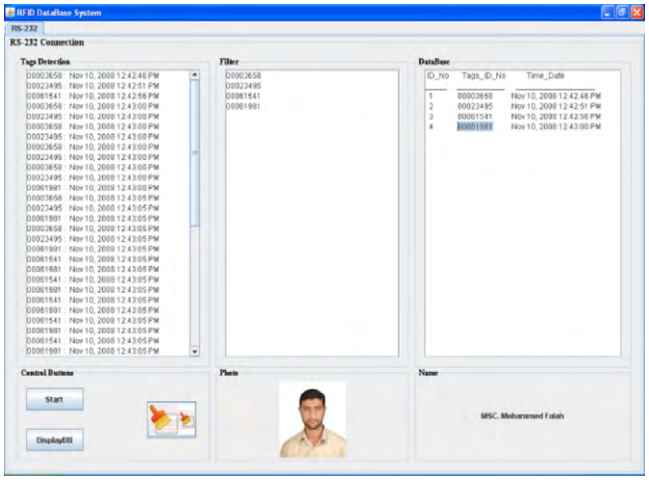

Fig. 11. RFID distributed database management system.

6. Assessment

RFIDTM encourages short development cycle for many RFID software applications. In this section, several RFID software applications are demonstrated to illustrate the features of RFIDTM. In fact, these applications are developed by software Engineering team in Universiti Sains Malaysia. These different applications use the RFIDTM to communicate with many heterogeneous readers like passive reader, active reader, high and low frequency reader, different connection interfaces (i.e. RS-232, USB and UTP), and implementing on different operating systems. Furthermore, this section demonstrates the RFIDTM support for networked application.

The first application is the Networked Based RFID for Tracking and Monitoring system which is uses the RFIDTM API to communicate with RF Code Reader which has embedded API. Here, the RF code reader functions as a server reader with the supporting frequency of 433 MHz Thus, the objective of developing networked based RFID Tracking and Monitoring system is to demonstrate that RFIDTM API does support typical commercial reader although with non-standard embedded API. The networked based RFID Tracking and Monitoring system is targeted to be customized for any monitoring and tracking usage. Here, as demonstration application, RFIDTM used to implement the application for tracking student in the lab as shown in Fig. 10. In this application, each student is given a tag with unique identification number. The Networked based RFIDTM system, is a general implementation of a RFIDTM based system. As the RF Code Reader supports two different connection types, the application is programmed to support concurrent read and detection from two different connection types (RS-232 and UTP). In this manner, this system can potentially enhance the reliability of the applications (i.e. through redundancy), that is, the system can still function even if one of the connections is broken. From this perspective, redundancy of reading help improves the availability and accessibility of the implemented system especially when multiple readers are employed. Additionally, this system also works with heterogeneous systems with different connection interfaces.

The second application is RFID Distributed Database Management System, The objective for developing the RFID Database System is to demonstrate the feasibility of RFIDTM API to support saving data from the reader to web server as database information. Here, the RFID Database System is implemented using *Java Data Base Connectivity* (JDBC) with MySQL Database on a local area network. Here, the system show that RFIDTM can make remote procedure call to fetch information from remote database, filter raw information, omits the redundant data, sends and save the useful information remotely on database in real time, and calls remote database from the remote computer, Fig. 11.

The third application is RFIDTM Protection System. The main objective of the development of RFIDTM protection system is to demonstrate that RFIDTM API can support heterogeneous RFID Readers. Furthermore, we also aim to demonstrate that RFIDTM can be extended horizontally by adding new functionality (i.e. which are not parts of the RFIDTM design such as Photo recognition, Sound, and Missing Detection classes) with live database management system (i.e. on the fly addition and removal of items in real time). Also this application aim to demonstrate that RFIDTM can support different connection modes (RS-232, USB, and UTP). Here, the RFIDTM Protection System adopts the 2.45 GHz USM CAIRFID (Contact less Active Integrated RFID) and RF Code Reader. In a nut shell, this system is designed to track and monitor specified items within a specific area as illustrated in Fig. 12.

The fourth application is Contact less RFIDTM. Here, the main objective for developing Contact less RFIDTM is to demonstrate the fact that RFIDTM supports remote connection with commercial reader. In this case, the RFIDTM API is used to communicate with

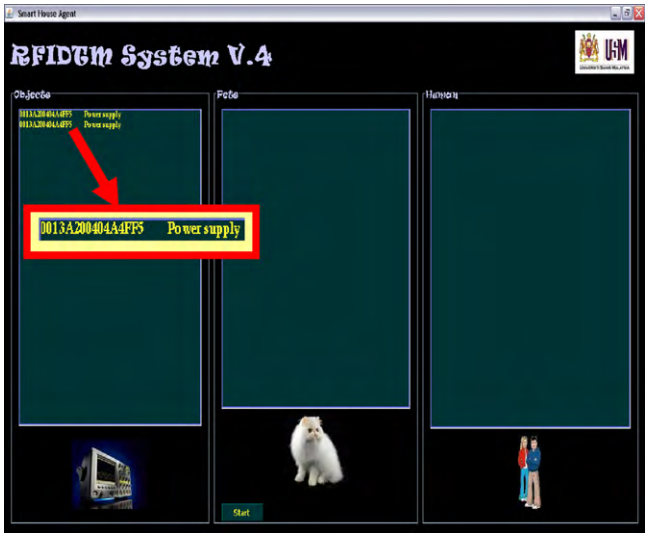

Fig. 12. RFIDTM protection system.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

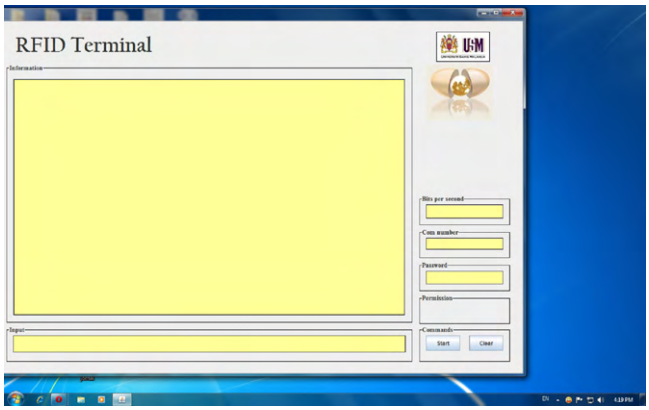

Fig. 13. Generic RFID terminal on Windows.

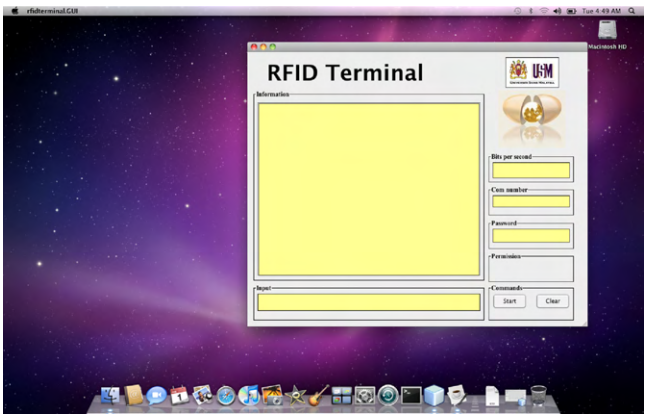

Fig. 15. Generic RFID terminal on Macintosh.

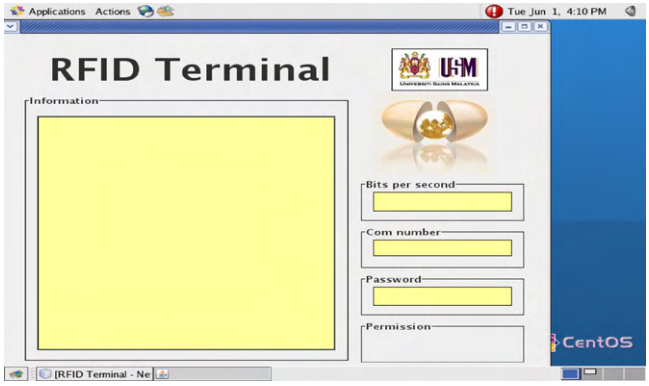

Fig. 14. Generic RFID terminal on Linux.

RFID module (cc1020ua Texas instrument), operating at 433 MHz frequency. This application (termed Contact less RFIDTM) is used to transfer large amount of information over large distance. With such information, the operator can control the remote RFID system without relying on network connection. With this functionality, remote monitoring of any RFID system can easily be achieved.

The fifth application is an RFID Terminal; the objective for RFID Terminal is to demonstrate the applicability of the RFIDTM API for any readers that use standard TCP/IP protocols. Here, RFID Terminal is a generic application designed to support most RFID readers supporting standard UTP or RS232 connection and even USB. Ideally, the development of RFID Terminal is to help debugging most of RFID readers. The RFID terminal has been used to debug the USM 2.45 GHz Contact less Active Integrated Reader (CAIR), RFCode Reader, Texas instrument 433 MHz and UHF passive reader on different OS (e.g. Windows, Linux, and Macintosh). The RFID Terminal application shows in Figs. 13–15.

7. Lessons learned

This section states a number of lessons learnt by applying RFIDTM API in different scenarios.

Firstly, RFIDTM API is indeed a reusable API. The fact that different applications can be straightforwardly developed is an indication of the usefulness of the APIs. Secondly, considering that all the applications have used different kinds of readers, the RFIDTM API supports heterogeneous RFID readers and is implemented on windows, Linux, and Macintosh, thus supporting the cross-platform functionality. Unlike API developed in other languages (e.g. C#, or C++) where portability amongst different operating systems can sometimes be questionable, RFIDTM API is guaranteed to be portable provided that the operating system does support the Java programming language. Thirdly, RFIDTM API also promotes redundancy. As seen in the Networked Based RFID Tracking and Monitoring System, RFIDTM API seamlessly supports concurrent read from RS232 and UTP connection. In fact, RFIDTM API also supports standard networked protocol. With such a feature, adding redundancy to user application would be a straightforward endeavor. Actually, the successful development of contact less application and generic RFID Terminal demonstrate the scalability and efficiency of the RFIDTM API. Even without any network connection, RFIDTM API can be employed to support and control long-range RFID system. Finally, RFIDTM API also appears to be sufficiently generic as the RFID Terminal demonstrates the capability to work with many kinds of readers.

8. Conclusion and future work

This paper proposed a new RFID API design for heterogeneous RFID system called the RFIDTM. The paper also discussed the lessons learned in terms of designing and implementing RFIDTM, as well as the experiences encountered with using the RFID system.

In fact, RFIDTM aims to support many developers and researchers in these two fields: software and hardware. On the software side, RFIDTM gives the developers a very short program development cycle due to its reusability, such as in cases where in many applications are implemented in a short time, allowing developers to avoid building their applications from scratch. Through these positive steps, RFIDTM helps software developers to improve the quality of their application, finishes the application in a short time, and prevents developers from writing hundreds of lines of source code, which often limits them to use a very low-level software language by providing RFIDTM reusable API. Similarly, RFIDTM helps many hardware developers to test their devices and read information from these devices. The RFIDTM likewise supports RFID systems and RF hardware such as those from Texas Instruments. As such, we can say that while RFIDTM appears to be generic enough, it can be used in a wide range of applications. With all the above mentioned advantages, we can say that the newly found RFIDTM can make life easy for the developers. Also, all the objectives have been achieved by different applications, different developers, and different operating systems.

In fact, the RFIDTM strictly obeys software engineering design for reusability, is very easy to set up, and can be easily used by developers. The RFIDTM design uses free software language as provided by Sun, specifically, Java. Similarly, the API classes are independent in terms of the underling system (cross-platform functionality) and have low-level native interface, such as JDK118-javaxcomm for Windows or RXTX for Linux and Mac OS. In other

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

words, RFIDTM is adaptable for upgrading (i.e., availability of new version).

By comparing the RFIDTM with other APIs, we can say that RFIDTM has ability to support cross-platform like Savant especially as this API was built using Java, which claim to supports cross-platforms. Also, Sybase, WinRFID, and FlexRFID, these different APIs, lack cross-platform functionality, and as such, they are limited in supporting distributed monitoring over a network for heterogeneous operating system. In addition, each of Sybase, Savant, WinRFID, and FlexRFID does not support adopt an open architecture that can be extended both (horizontally and vertically), do not support redundancy for highly available systems, and do not also connections and customize filter with other APIs and middleware. In comparison with the RFIDTM, Sybase support distributed monitoring, which are based on Microsoft clusters. In fact, all the APIs claim to support the different RFID systems, although it seems there is no clear meaning for the word "different". As such, it seems that the other APIs support specific different RFID systems, while RFIDTM covers the general meaning for supporting different RFID systems. Moreover, RFIDTM has proven its reusability, supports for heterogeneous readers with different connection interfaces, availability of the customized filter with other APIs, and ability for distributed monitoring through TCP/IP. In addition, RFIDTM is designed to work on computers not to be embedded in readers, even if many in the past have used embedded RFID readers. A comparison of the RFIDTM with other embedded APIs shows that the former (RFIDTM) has greater advantages including the following:

- Using RFIDTM give users the ability to reduce hardware cost by removing the memory chip for recording information inside the reader. In fact, the API replaces this chip by a software buffer inside the hosting PC for each connection. Moreover, filtering process is done by the software layer rather than on chip middleware layer, that is reduced the complexity, and hence, the overall cost of the reader.
- The RFIDTM can communicate with many and different RFID readers (with or without embedded APIs), while each embedded API is merely designed to communicate with its reader.
- It can be recalled that an RFIDTM defines a set of reusable APIs through various possible arrangements of service components.

The RFIDTM is designed to support heterogeneous RFID system but with a lower cost and stronger support (a general model for supporting RFID system in a distributed system). This paper explained the RFIDTM's design, features, and characteristics. As such, the RFIDTM is a novel API design that promotes reusability, redundancy, as well as the ability to support interoperability within different operating systems and hardware implementations. Overall, the RFIDTM has good features that can support the developers in different RFID fields. The supporting distributed computers are also considered to be very important especially in the industrial fields, hospitals, education, and even in reducing the cost of using multi-readers (i.e., to enable many users access the same reader from different places to get knowledge about the RFID information type and format).

As a part of our future work, we will consider the GPS system. The mix of outdoor tracking using GPS and the indoor/outdoor tracking using RFID system by RFIDTM, will give an important advantage for tracking and monitoring applications in many fields. In this case, developers can add a new class to the RFIDTM API to establish connection with a GPS, allowing the exchange of information between the two systems. Through these, tracking and monitoring of fields can be achieved at a high coverage area with a lower cost.

References

Ajana, M.E., Harroud, H., Boulmalf, M., Hamam, H., 2009. FlexRFID: a flexible middleware for RFID applications development. In: IFIP International Conference on Wireless and Optical Communications Networks, 2009 (WOCN'09).

BITKOM, 2005. RFID White Paper Technology, Systems, and Applications. German Association for Information Technology, Telecommunications, New Media e.V., Berlin.

Bolotnyy, L., Robins, G., 2007. The case for multi-tag RFID systems. In: IEEE International Conference on Wireless Algorithms, Systems and Applications, 2007 (WASA 2007).

Glasser, D.J., Goodman, K.W., Einspruch, A.N.G., 2007. Chips, tags and scanners: ethical challenges for radio frequency identification. Ethics and Information Technology 9, 101–109.

Huifang, D., Junbin, C., 2009. Design and implementation of business logic components of RFID Midware for logistics customs clearance. In: International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government, 2009 (EEEE'09).

IEEE, 2009. Developing National Policies on the Deployment of Radio Frequency Identification (RFID) Technology. IEEE, Washington, DC.

Intel Corporation, Autentica, Cisco Systems, San Raffaele Hospital, 2005. Using RFID Technologies to Reduce Blood Transfusion Errors. San Raffaele Hospital, Milan, Italy.

Intermec, T.C., 2007. Supply Chain RFID: How It Works and Why It Pays. Intermec, USA.

Ishikawa, T., Yumoto, Y., Kurata, M., Endo, M., Kinoshita, S., Hoshino, F., Yagi, S., Nomachi, M., 2003. Applying Auto-ID to the Japanese Publication Business to Deliver Advanced Supply Chain Management, Innovative Retail Applications, and Convenient and Safe Reader Services. Auto-ID Center, Keio University.

Jackson, R.J., 2004. Radio Frequency Identification (RFID), available at http://www.scribd.com/doc/25761881/identificarea-prin-radio-frecventa-rfid (accessed March 2010), White Paper.

Kranenburg, R.V., 2006. RFID Implants, available at http://www.clubofamsterdam.com/content.asp?contentid=640 (accessed March 2010).

Prabhu, B.S., Su, X., Ramamurthy, H., Chu, C.-C., Gadh, R., 2005. WinRFID—A Middleware for the Enablement of Radio Frequency Identification (RFID) Based Applications. Wireless Internet for the Mobile Enterprise Consortium (WINMEC), Los Angeles.

Scassa, T., Chiasson, T., Deturbide, M., Uteck, A., 2005. An Analysis of Legal and Technological Privacy Implications of Radio Frequency Identification Technologies Office of the Privacy Commissioner of Canada.

Shien-Chiang, Y., 2007. Implementation of an innovative RFID application in libraries. Library Hi Tech 26, 398–410.

Solutions, I., Sybase, I., 2008. RFID Anywhere™ Developer's Guide, Documentation.

Srivastava, L., 2005. The Case of Radio Frequency Identification. ITU Workshop on Ubiquitous Network Societies.

Supply Insight, I., 2006. RFID Applications in Hospital Equipment Tracking, available at http://www.supplyinsight.com/RFID_in_Hospital_Equipment_Tracking.htm (accessed March 2010).

Sybase, I., 2008. RFID Anywhere and Ekahau RTLS Overview, available at http://www.sybase.com/detail?id=1056007 (accessed May 2009).

Wang, W., Mcfarlane, D., Brusey, J., 2005. Timing Analysis of Real-Time Networked RFID Systems. Cambridge Auto-ID Lab, Cambridge, UK.

Mohammed F.M. Ali obtained his B.Eng. in Computer Engineering- Technical college of Mosul- Iraq in 2006 and his MSc degree in Electrical and Electronic Engineering-University science of Malaysia in 2010. He is currently a PhD candidate in the School of Electrical and Electronic Engineering-University science of Malaysia. His research interests include RFID system, computational intelligence, computer networking, and computer languages.

Mohammed I. Younis obtained his BSc in computer engineering from the University of Baghdad in 1997 and his MSc degree from the same university in 2001. He is currently a PhD candidate attached to the Software Engineering Research Group of the School of Electrical and Electronic Engineering, USM. He is a Senior Lecturer and a Cisco instructor at Computer Engineering Department, College of Engineering, University of Baghdad. He is also a software-testing expert in Malaysian Software Engineering Interest Group (MySEIG). His research interests include software engineering, parallel and distributed computing, algorithm design, RFID, networking, and security. These research works have been published in more than 40 international journal papers, conferences and other publications. He is also a member of Iraqi Union of Engineers, IEEE, IAENG, IACSIT, and IJCTE.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030

Kamal Zuhairi Zamli obtained his BSc in Electrical Engineering from Worcester Polytechnic Institute, Worcester, USA in 1992, MSc in Real Time Software Engineering from CASE, Universiti Teknologi Malaysia in 2000, and PhD in Software Engineering from the University of Newcastle upon Tyne, UK in 2003. He is currently an Associate Professor attached to the Software Engineering Research Group, in the School of Electrical and Electronic Engineering, USM. His research interests include software engineering, software testing automation, and algorithm design.

Widad Ismail graduated from University of Huddersfield, UK in 1999 and earned First Class Honors in Electronics and Communications Engineering and she received her PhD in Electronics Engineering from University of Birmingham, UK in 2004. She is currently a Senior Lecturer at the School of Electrical and Electronics Engineering, USM in Nibong Tebal, Penang, Malaysia. She has contributed extensively in research and in the areas of Radio Frequency Identification (RFID), Active Integrated Antennas (AIA), RF systems and Wireless Systems Design. She has initiated Auto-ID Laboratory (AIDL), Malaysia in 2008 as a research and commercialize oriented centre where the main objective is to become a hub for research and commercialization activities. These research works have produced 8 filed patents, 4 international awards, 3 commercialized research products & more than 50 publications including international journal papers, conference/seminars and other publications. She is also a member of IEEE and Wireless World Research Forum (WWRF). Email: eewidad@eng.usm.my.

Please cite this article in press as: Ali, M.F.M., et al., Development of Java based RFID application programmable interface for heterogeneous RFID system. J. Syst. Software (2010), doi:10.1016/j.jss.2010.07.030