Rifidi Toolkit: Virtuality for Testing RFID Systems

Andreas Huebner*, Christian Facchi* and Helge Janicke†
*Institute of Applied Research, University of Applied Sciences Ingolstadt, Germany
{andreas.huebner, christian.facchi}@haw-ingolstadt.de
†Software Technology Research Laboratory, De Montfort University, Leicester, United Kingdom
heljanic@dmu.ac.uk

Abstract—**The Rifidi Toolkit is an open source framework for virtual Radio Frequency Identification (RFID) environments. It allows to emulate RFID devices and can be used as a basis for testing RFID applications. The concept behind the Rifidi Toolkit is already widely adopted in industry and has been accepted in science. This paper gives an introduction on the the toolkit's architecture and design. It further points out how to use the toolkit for testing RFID applications and proposes new features and functionality needed for robust RFID application testing with the Rifidi Toolkit.**

Keywords-RFID; test-data generation; software testing; virtualisation; Rifidi

I. Introduction

RFID [1] is gaining momentum in industry as an increasing number of RFID applications are deployed. The trend towards automatic identification of objects also increases the demand for qualitative and fail proof RFID applications. Therefore research on testing RFID systems speeds up and methodical approaches on testing RFID systems are needed. Even though most publications on testing RFID focus on performance evaluation, a commonly encountered problem with all approaches on testing RFID, is that it is very expensive to fund the physical RFID test environment. To address this problem the Rifidi Toolkit [2] was developed. The toolkit allows to virtualise the RFID environment, therefore, drastically reduce the test costs and in consequence also reducing the overall project costs.

This paper gives an introduction into the open source framework Rifidi and shows the design ideas behind the software. It points out what features are required to use the toolkit for testing RFID applications. Furthermore, it exposes additional improvements to the Rifidi Toolkit, so it can be applied as a basis for functional testing of RFID applications.

The remainder of the paper is structured as follows: In Section II, an overview of the tools included in the Rifidi Toolkit is given. The *Emulator Engine*, the central part of the toolkit, is explained in detail in Section III. Then, the architecture of the different modules is explained in Section IV. An introduction to requirements for testing RFID is given in Section V. Section VI presents the related work and shows the acceptance of the Rifidi Toolkit and the principles behind it. Drawbacks and proposed enhancements can be found inSection VII. Finally, a conclusion and a future perspective are given in Section VIII.

II. Rifidi Toolkit Overview

The Rifidi Toolkit enables the virtualisation of RFID readers of various vendors for flexible and improved testing of RFID applications. Utilizing virtual RFID readers in this context means less physical readers need to be available for testing. Therefore, it lowers the barrier to enter the RFID world and allows to save resources and time. The Rifidi Toolkit is available since 2006 as open source software on SourceForge.net [3] and already established on the market as a valuable resource for many users, which can be seen on the number of downloads (over 50,000 since 2006 [4]).

The Rifidi Toolkit consists of three software tools; namely the *Emulator*, the *Designer* and the *TagStreamer* [5]. All these are based on the *Emulator Engine*, the central part of the Rifidi tool-suite which is capable of emulating virtual RFID devices.

Emulator - The Rifidi Emulator is a graphical interface for controlling and interacting with virtual readers. It allows the emulation of readers and tags as well as read and write events. Additionally, it provides access to the virtual readers like their physical counterparts.

Designer - The Rifidi Designer is a tool to build custom 3D production environments, that can be used for visually simulating the RFID data flow. It is also based on the *Emulator Engine* and allows the emulation of RFID readers and the interaction with them. It is not actively maintained anymore but can still be accessed on the SourceForge.net website.

Tag Streamer - The Rifidi Tag Streamer is a performance testing tool that allows to generate large numbers of virtual readers and tags to evaluate the RFID system.

III. The Rifidi Emulator Engine

The *Emulator Engine* is the core part of the Rifidi Toolkit. It is responsible for managing and controlling the emulation of the RFID devices. One Instance of the *Emulator Engine* can control multiple virtual readers of various vendors at the same time. All parts of the Rifidi Toolkit are implemented in Java and use different technologies around the Eclipse Framework, e.g Equinox, JFace and SWT.

The basic tasks of the *Emulator Engine* are:

- Handle communication between RFID reader and client
- Execute commands issued to the virtual reader
- Management of the antennas and the related field of sight
- Control the reader specific components, like signals on the *general purpose input/output* ports (GPIO ports)

Because of a Service Oriented Architecture (SOA) [6] approach, the functionality of the *Emulator Engine* is distributed in two core services. These services can be obtained through the *ServiceRegistry* and used to manage virtual readers, virtual tags and keep track of the tags in the Rifidi Environment.

ReaderManager - This service is responsible to manage the devices and offers of the following actions:

- Create a virtual reader
- Delete a virtual reader
- Start a virtual reader (power on)
- Stop a virtual reader (power off)
- Add a virtual tag to a reader's field of sight
- Remove a virtual tag from a reader's field of sight
- Get a list of virtual tags currently in the readers field of sight

IRifidiTagService - This service is used to create and handle virtual tags. It consists of the following functionality:

- Create a virtual tag
- Delete a virtual tag
- Track virtual tag data changes

All existing Rifidi tools are based on the *Emulator Engine* and allow an intuitive, graphical access to the presented functionality. However, the Rifidi Toolkit can also be used as a basis for other tools, which rely on the emulation of virtual RFID devices, e.g., test data generators. But, to use the capabilities of the tool-suite for improved testing of RFID applications, a better insight on the framework is necessary.

IV. Emulator Engine Architecture

This section gives more insight on the architecture of the Rifidi tool-suite and describes how to use this architectural model as a basis for virtual readers. It concludes with an overview of the command flow through the *Emulator Engine*.

A concrete implementation of a virtual reader is called *reader module*. To seamlessly integrate into the framework each *reader module* implements interfaces of the components, presented below.

The architecture of the *Emulator Engine* is based on the structure of a physical RFID reading device. The idea was to develop the virtual readers similar to their physical counterparts and therefore to use the same logical components. Figure 1 gives an overview of the different components of the *Emulator Engine*. Each box represents a logical counterpart of a physical reader and can be seen as a generally valid abstract component of all virtual readers.

A. Emulator Engine Components

The components of the *Emulator Engine* are Communication, Command Processor, Radio, General Purpose Input/Output and Shared Resources. The different tasks and functionalities of the virtual reader's components are explained in the following:

Communication - Even though most modern RFID devices use an Internet Protocol (IP) connection for communication,

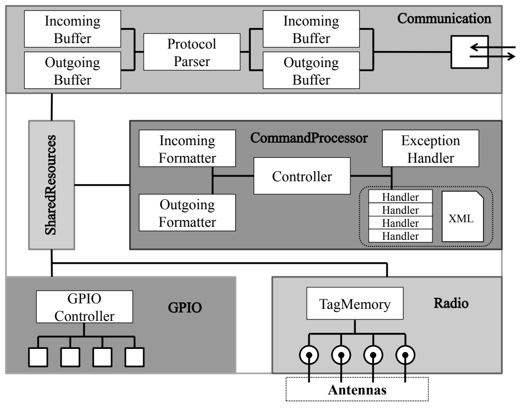

Figure 1. Emulator Engine Architecture Overview [5]

there are still some vendors using other interfaces like serial link, e.g., RS232, or other proprietary ones. To map these diverse ways of communication to the framework a generic communication part for each virtual reader needs to be considered. In the Rifidi environment this is realized through the communication component. It enables receiving and sending of messages as well as forwarding the encoded messages to or respectively from the command processor. The communication is divided into two parts:

Protocol Parser: The protocol parser converts the messages received in binary format to a message the command handler can deal with. This can be either a reader specific message format or an plain Java object. The *Protocol Parser* is also used this way when it converts a message from the command handler to a binary message, which can be sent through the reader specific communication. With respect to IP-based communication, especially regarding fragmented packets, the parser is also responsible to determine if a messages was received completely.

Buffer: The buffer is used to store messages and enable asynchronous communication from the communication channels. There is one buffer for incoming and outgoing messages.

Command Processor - The interaction with common RFID readers is based on command/response protocols. Subsequently the issued commands need to be parsed, executed, and finally, responded. In the Rifidi environment, the command processor can be seen as the central processing unit and is used for this purpose. For each command it invokes the appropriate methods of the virtual reader. A list of commands and the corresponding actions, implementing the functionality of the command, are defined in a XML file called `reader.xml`. The Rifidi framework utilizes *Java Reflections* [7] to execute the classes and methods specified in this XML file. The command processing layer is consisting of three parts:

Formatter: As for the buffers, there are formatters for each direction of communication. The incoming formatter is used to strip and decompose the commands. Usually, the first parts of a command determine the command type, all additional information are arguments and parameters. The outgoing formatter assembles outgoing messages for further processing in the communication layer.

Controller: The controller, also called command handler, is the central processing unit for commands. Each command issued to the virtual reader is executed here. The *HandlerMethods* are invoked through reflection.

Exception Handler: The exception handler is a special *HandlerMethod*, which takes care of unknown, undefined or misspelled commands. Usually, error descriptions are send back to provide feedback.

Radio - The radio component represents the air interaction capabilities of the virtual RFID device and allows to interact with virtual tags. In the Rifidi Emulator, for example, a user can control when a tag is added to an antenna or when a tag is removed. Additionally, to the users interaction the reader can also, depending on the reader capabilities, write or read the tag. Furthermore, it is vendor specific how ”tag events” are stored and handled in a reader, therefore each virtual reader has its own radio component and memory structure. As an example, the *Alien 9800* (Alien Technology Corporation) RFID reader allows either to list all read events since the last poll or to list just the currently available tags in the field of sight.

Tag Buffer: The tag buffer implements the memory structure and the over-the-air interaction capabilities of the virtual reader. The Tag Buffer is associated to the virtual antennas of a reader. The antennas represent the interfaces for the *Emulator Engine* to simulate RFID tag operation events. It has to be distinguished between read and write operations. Reading tags means a tag’s information was read by the reader and writing tags means the tag was modified during time the tag was available in the field of sight. An Event is either a tag appears on the antenna or disappears. In more detail, this means a list of virtual tags is either added to the antenna or removed from the antenna.

General Purpose Input/Output - Some RFID devices allow additional sensors to be connected to it. This is widely known as *General Purpose Input/Output* (GPIO) and realised as small electrical connectors on the RFID reader. The presence of GPIO ports is also reader specific and the virtual functionality is provided by this part. Currently, only the Low Level Reader Protocol (LLRP) Reader and the Alien 9800 Reader leverage this functionality in the Rifidi Framework, yet.

Shared Resources - The shared resources are used as a housekeeping component for each virtual reader. It is a central instance holding together the different components and allowing to transfer data objects from the above described layers in this section.

B. Overview of the Command Flow

Concluding with the different components of an virtual reader, Figure 2 gives an overview of the command flow through the different parts of the *Emulator Engine*. It shows a schematic view of the data exchange and interactions of the different components. The dotted lines around the components indicate implementation specific parts, which can be different in each virtual *reader module*. All incoming commands are received by the *IncomingBuffer*, encoded in a binary format. Once all bytes of the message are read, it is forwarded to the *Protocol*. The *Protocol* decodes the binary messages to a processable format and hands it further to the *IncomingMessageBuffer*. The *IncomingMessageBuffer* stores the Message until the *CommandProcessor* is available for processing. The *CommandProcessor*, consisting of the *CommandFormatter*, the *CommandSearcher* and *CommandAdapter*, is parsing the message and looking up the corresponding *HandlerMethod*. Finally the located method is executed and the intended actions are performed. If there is a response or a result of the previously executed command the data is going through all components again in reverse order. Following the dotted arrows, the reply first goes through the *CommandFormatter* again. Afterwards, it is given to the *OutgoingMessageBuffer*, then encoded by the Protocol, and finally, sent to the *OutgoingBuffer*.

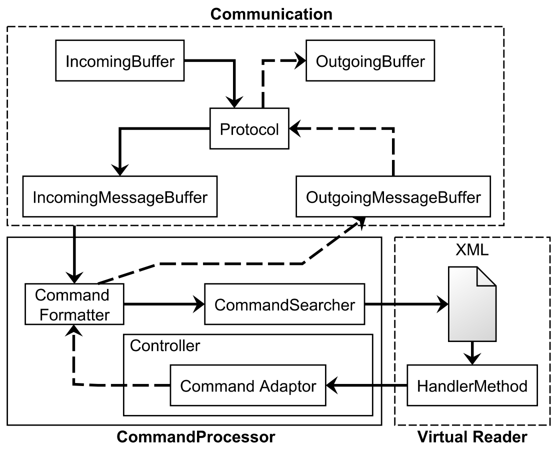

Figure 2. Rifidi Command Flow Overview [5]

V. Requirements for testing RFID

In this section, an overview of a generic RFID system is given. Based on the description of the sketched system, a transformation into a virtual system is performed, while both systems are used to demonstrate the testing capabilities. Finally, constraints as well as requirements to be fulfilled for testing with the virtual system are listed.

A. Structure of a simplified RFID system

A simple RFID system is composed of one or more RFID readers, a RFID middleware and an application. The RFID tags are usually attached to objects the RFID system is interested in. Once an object with an attached RFID tag gets close to a reader’s antenna the tag will be read. This is often referred to as a tag read event. As long as the tag stays in the field it can also be written, which is then referred as write event.

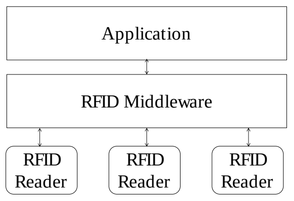

Figure 3. RFID System

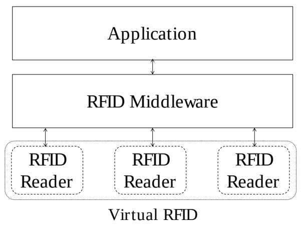

Figure 4. Virtual RFID System

Figure 3 shows a schematic RFID system and the interaction between the components. RFID Tags are read on the antennas of the reader and reported to the middleware. This middleware filters redundant data, applies logical evaluations and generates business events for the RFID application.

To test a system like this, certain RFID events, usually read events, need to be generated. These events will then be processed by the middleware and finally handed to the RFID application. Depending on the business logic of the application a specific result is produced.

In today’s practical testing approaches the generation of tag read events is produced by someone walking through the readers field of sight. The resulting effects in the application are then evaluated.

B. *Virtuality for testing RFID*

To improve the manual test process the real world devices can be substituted by virtualised RFID, with virtual readers from the Rifidi Toolkit, with an otherwise unchanged tool chain. From the perspective of the RFID middleware and application the simulated system cannot be distinguished from a real system. This ”virtuality”-system can even be a system composed of randomly mixed real and virtual devices. Figure 4 shows the changed RFID system. In this case, all RFID devices have been substituted by virtual readers. A tester can now place virtual tags on the virtual readers, and therefore, generate the same input as with the previously described real environment.

C. *Testing with Rifidi*

To enhance testing with this ”virtuality”-system and enable automatic testing, the newly created virtual environment needs to be instrumented. For the instrumentation of the virtual environment a new test controller can be used. In the Rifidi Toolkit this functionality could be provided by a new tool using the *Emulator Engine* to map the movements of tags in the real world. However, to test RFID applications a missing part is still the automatic evaluation of the RFID middleware respectively the application. This is a challenging task because of the diverse variety of RFID applications. Compared to middleware systems, where the *Application Level Event* (ALE) [8] standard can be used to communicate and evaluate the test results, for each application an evaluation adapter needs to be implemented. Additionally, to the missing interface, to evaluate the test results in applications, even more complexity is introduced by the attributes of RFID data. According to [9] RFID data consists of a data stream with the following attributes:

- Redundant Data
- Grouped Data
- Moving Route
- Noisy Data

These parameters need to be considered during the generation of test data. Rifidi supports most of the requirements in its virtualisation of RFID readers, however, it is notably lacking support to model events that occur due to physical effects such as interference and timing delays. As part of our ongoing research, we are investigating realistic interference models and also work on real-time aspects of RFID testing.

VI. Related Work

Since 2006, when the framework was published on SourceForge.net, it has been downloaded over 50,000 times [4] and used by developers, teachers and researchers. The Rifidi Toolkit has been referred in many publications, which shows the significance of the presented tool. Furthermore, in this paper we presented the architectural consideration and the requirements that underpin the current implementation of Rifidi to enable an even more widespread adoption of the framework in particular with the view of using Rifidi in the Quality Assurance of RFID applications.

However, there are similar developments capable to emulate virtual RFID readers. This section covers some of the approaches and points out what the differences of the implementations are. Additionally, it shows where the Rifidi Toolkit was used in science, research, education and industry.

A. *Comparable RFID Emulators*

Currently there are two implementations of virtual RFID emulators mentioned in scientific publications, but unfortunately neither the *Virtual Test Toolkit* from the Pusan National University nor the *RFID Performance Test Tool* from the Feng Chia University was available for download and comparison. Hence, the further statements are only based on information which could be obtained from publications and not from concrete experiments.

Pussan National University (PNU LIT) - [10], [11], [12], [13] describe a virtual reader emulator to evaluate a RFID Middleware mainly with the focus on performance issues. The Toolkit is divided into three parts; the *Virtual Application Emulator*, the *Virtual Device Emulator* and the *Toolkit Operator*. Where the *Virtual Device Emulator* is comparable to the Rifidi Toolkit. It is capable to emulate one or more virtual RFID readers and virtual RFID tags. The virtual readers can interact with the virtual tags, which are following the EPC Global Data Tag Standard. The *Toolkit Operator* controls the virtual environment and is the central part of the Virtual Test Toolkit. The *Virtual Application Emulator* is connected to the RFID Middleware and acts as a RFID Application. The purpose of the *Virtual Application Emulator* is to collect information regarding the performance of the middleware under test for performance evaluation purposes. The toolkit was firstly published in 2009 [10] and follows the similar objectives than the Rifidi Toolkit and the RFID Performance Test Tool. It differs to the Rifidi Toolkit mainly in the emulation's level of detail. For example, a fine grained access to the virtual readers including configuring the output format of tag events seems not to be possible. Nevertheless, it can be used as a basis for testing, but it is not as flexible as the Rifidi Toolkit.

Feng Chia University (FCU RFIDLab) - Jongyoung and Naesoo [14] introduce the design and implementation of a Performance Test Tool for RFID middleware, consisting of a test data generator and a result data estimation part. The test data generator supports different tag data standards and various reader protocols, e.g. from EPC Global, Alien Technology and Motorola (former Matrics). The result data estimation part implements the ALE specification and connects to the RFID middleware, which is the *Software Under Test* (SUT), as a virtual application. The reader emulator, which is part of the test generation tool, is controlled via a graphical user interface and allows to specify how many tags will be generated. Additionally to the amount of tag events the simulation should generate, it allows to specify a pattern for the encoding of the generated tag data and a timing interval for the events. After the previously specified test data was generated, it is accessible for the RFID Middleware through the "result data transmission"-part. Summarized the introduced tool is similar to the concept of the Rifidi TagStreamer. It is a performance test tool to test RFID Middleware and allows the emulation of vendor specific reader protocols. But, different to the TagStreamer, it does not allow a fine grained access or interaction with the virtual readers. Nevertheless, the advantage of the Performance Test Tool is the result estimation part, which is used as the upper tester of the test suite.

Hardware Emulators - Beside the software emulators, there is also a variety of physical emulators available. The hardware emulators are mainly used to simulate RFID readers, in order to test new air protocols, modifications to existing protocols and new command sets. But, some of the devices can even simulate RFID tags. For example, CISC Semiconductors offers the *RFID Tag Emulator* [15], a mobile device capable of emulating multiple RFID tags. This tag simulator can be used to analyse and test RFID Reader performance with certain tag populations. But, similar to real RFID readers these simulators come with the same limitations, regarding the high investment costs, compared to virtual readers. This downside is especially obvious for performance testing, where a huge number of devices is needed.

B. Publications about Rifidi

The concepts of the Rifidi Toolkit have been discussed in many publications. A not exhaustive list of publications around the Rifidi Toolkit is given in this section:

Palazzi and Ceriali [16] provided a critical investigation of the capabilities of the Rifidi Toolkit regarding RFID system testing. For this investigation they use the toolkit in a case study to demonstrate the current potentials. Siror et al. [17] use Rifidi as a basis to evaluate the usability of an an automatic evaluation of a customs verification process. In both cases the Rifidi framework was successfully used to simulate the RFID environment and therefore supported the fast and easy demonstration of the case studies. Mueller et al. [18] compare different test data generators and conclude, that the Rifidi Toolkit is an specialized data generator which can generate RFID events by emulating RFID readers.

C. Applications in Science, Education and Industry

The Rifidi Toolkit is not only used in research, it is also used in many different other areas. Rifidi provides enormous benefits especially for industry and education. The reason to use the virtual RFID environment here is mainly because of the reduced costs and the availability of readers. In summary, researchers, scientists and teachers can work with RFID readers without having the budget for the hardware and neither to argue with their colleagues when the device is available. As far as it is known to the knowledge of the authors the University of Applied Sciences Ingolstadt and University of Applied Sciences Regensburg are using the toolkit beside others [19] for educational purposes.

But, the previous mentioned benefits are also valid for industrial users. The Rifidi Toolkit is used for a wide variety of industrial applications and helps companies to realise RFID projects faster with less costs. Some of the users known to the authors are BMW, HP and IBM.

VII. Enhancements for Rifidi

The Rifidi Software can be used as a basis for testing RFID applications and its infrastructure. To truly match the requirements for testing, some improvements to the framework need to be made. Four categories can be distinguished:

Reader Support - To keep the virtual RFID framework up-to-date it is necessary to map the changes in the development of physical readers to the Rifidi framework. Even when the *Emulator Engine* already supports a great variety of readers and appropriate reader protocols the framework could be enhanced by adding more readers to it. Integrating new virtual readers is a lot of effort, especially because vendors usually do not provide the necessary information. Therefore, typically, a reader has to be reverse engineered for implementation purposes. Not only the time it takes, but also the cost of the hardware needs to be taken into account. As a result, better guidelines and methodologies to enhance the software development of the Rifidi suite have to be found.

Tag Creation - Another drawback in the Rifidi Environment is the generation of virtual tags. Currently, the creation of tags is based on random numbers which are grouped by the means of the encoding, like *Serialized Global Trade Identification Number* (SGTIN). According to Zhang et al. [9], one can distinguish between *Semantic Invalid Data* (SID) and *Semantic Valid Data* (SVD). Whether to use SID or SVD depends on the test objective. For example, SID is used when the performance of an system needs to be evaluated. In this case the meaning of the data is irrelevant for the test objective. To test functional aspects of RFID middleware or applications the identifiers used in the simulation need to have a semantic meaning determined by SVD. The semantic meaning expresses the relationship between the tag's identifier and the actual object the tag is attached to. Therefore, the tag's identifiers need to be associated with the actual serial numbers an application's database stores. This can either be achieved by establishing a connection to the database, an interface to the system, respectively an adapter, or with a manually instrumented set of tag patterns during the creation of tags.

Robustness and Stability - A necessary attribute of software used for testing is that itself does not introduce errors and failures. Therefore, using the Rifidi Framework as a basis for testing RFID systems also leads to the question: Is the software stable and reliable enough to truly be the basis for testing? Concepts and studies concerning this issue need be made and show that Rifidi is fulfilling these requirements.

Scripting and Automatic Execution - Currently, the Rifidi Framework exposes its functionality through the three tools with GUI's. To enable more complex testing, the framework need to be instrumented by a test manager. This could be achieved by introducing scripting support for the automatic execution of tag movement patterns. The scripting language could be a domain specific language to easily match the requirements for testing RFID applications and their functionality.

VIII. Conclusion and Future Work

The Rifidi Toolkit is virtual framework for the simulation of RFID devices. The architecture is based on the real layout of physical RFID readers and allows a fine grained access to the virtual RFID readers like their physical counterparts. It serves as a framework for the implementation of many different readers with its flexible and general layout. Some of the principles behind the Rifidi architecture seem to be already adopted in different areas of science and industrial use. Furthermore, it is a basis for further research and allows testing of RFID applications. The Rifidi Toolkit can be extended to improve testing capabilities and support modern testing techniques.

Physical Effects on the Air Interface - A drawback, which can be a task for future work, is that none of the presented software tools truly cover the physical constraints of the air interface. In reality, each RFID reader is susceptible for missing tag reads. To be able to truly test RFID environments the physical effects in the *radio frequency* (RF) field have to be taken into account. This means the RF field and its effects also need to be simulated in the emulation of the RFID readers. An approach could be to introduce heuristics and statistical based reasoning functions, which need to represent the physical constraints more closely. But, furthermore, also the impact of missed tag readings on the RFID application with regard to testing has to be researched.

IX. Acknowledgements

The authors would like to thank the Software Technology Research Laboratory (STRL) and the Faculty of Technology of the De Montfort University, especially Peter Norris and Stephen Ison for providing the appropriate environment for research. This research has been funded by project grants from the German Federal Ministry of Education and Research (BMBF). Project: ITERA, FKZ 01QE1105B.

References

[1] K. Finkenzeller, *RFID-Handbook, 3rd edition*. Wiley, 2010.

[2] Pramari LLC, "Rifidi Project," online: http://www.rifidi.org, 2006, [accessed: December 14, 2011].

[3] ——, "Rifidi - from rfidea to business reality," online: http://sourceforge.net/projects/rifidi/, 2011, [accessed: March 28, 2011].

[4] ——, "Rifidi Statistics," online: http://sourceforge.net/projects/rifidi/files/stats/timeline?dates=2005-01-05+to+2012-07-02, 2012, [accessed: June 01, 2012].

[5] ——, "Rifidi Emulator Documentation: Developers Guide," online: http://wiki.rifidi.org/index.php/Engine_Overview, 2008, [accessed: December 14, 2011].

[6] R. Perrey and M. Lycett, "Service-oriented architecture," in *Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on*, jan. 2003, pp. 116 – 119.

[7] G. McCluskey, "Using java reflection," online: http://java.sun.com/developer/technicalArticles/ALT/Reflection/, 1998, [accessed: June 12, 2012].

[8] EPCglobal inc., "The application level events (ale) specification, version 1.1.1," online: http://www.gs1.org/gsmp/kc/epcglobal/ale/ale_1_1_1-standard-core-20090313.pdf, 2009, [accessed: June 05, 2012].

[9] H. Zhang, W. Ryu, B. Hong, and C. Park, "A test data generation tool for testing rfid middleware," in *Computers and Industrial Engineering (CIE), 2010 40th International Conference on*, july 2010, pp. 1 –6.

[10] C. Park, W. Ryu, and B. Hong, "RFID Middleware Evaluation Toolkit Based on a Virtual Reader Emulator," in *Emerging Databases, The 1th International Conference on Emerging Databases 2009*, 2009, pp. 154–157.

[11] G. Lee, H. Zhang, C. Park, W. Ryu, and B. Hong, "Design and Implementation of Virtual Test Toolkit for Testing RFID Middleware," in *Intelligent Manufacturing and Logistics Systems, The 6th International Conference on IML 2010*, 2010, pp. 1–6.

[12] H. Zhang, W. Ryu, B. Hong, and C. Park, "A test data generation tool for testing rfid middleware," in *Computers and Industrial Engineering (CIE), 2010 40th International Conference on*, 2010, pp. 1–6.

[13] J. Park, W. Ryu, B. Hong, and B. Kim, "Design of toolkit of multiple virtual readers for scalability verification of RFID middleware," in *The Second International Conference on Emerging Databases (EDB 2010)*, 2010, pp. 56–59.

[14] L. Jongyoung and K. Naesoo, "Performance test tool for rfid middleware: Parameters, design, implementation, and features," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, vol. 1, feb. 2006, pp. 149 –152.

[15] C. Semiconductors, "Rfid tag emulator," online: https://www.cisc.at/?id=25, 2012, [accessed: June 05, 2012].

[16] M. D. M. C. E. Palazzi, A. Ceriali, "RFID emulation in Rifidi Environment," in *International Symposium on Ubiquitous Computing (UCS'09)*, 2009.

[17] J. Siror, S. Huanye, and W. Dong, "Automating customs verification process using rfid technology," in *Digital Content, Multimedia Technology and its Applications (IDC), 2010 6th International Conference on*, aug. 2010, pp. 404 –409.

[18] J. Müller, M. Schapranow, C. Pöpke, M. Urbat, A. Zeier, and H. Plattner, "Best practices for rigorous evaluation of rfid software components," in *RFID Systech 2010, RFID Systech 2010 - European Workshop on Smart Objects: Systems, Technologies and Applications*, 2010.

[19] Transcends LLC (former Pramari LLC), "Academic partners," online: http://www.transcends.co/partners/academic, 2011, [accessed: June 03, 2012].