2012 IEEE Fifth International Conference on Software Testing, Verification and Validation

An extended LLRP[1] model for RFID system test and diagnosis

Rafik KHEDDAM, Oum-El-Kheir AKTOUF, Ioannis PARISSIS
Grenoble Institute of Technology
LCIS, 50 rue B. de Laffemas
26000 VALENCE, FRANCE
firstname.lastname@lcis.grenoble-inp.fr

***Abstract*—In recent years, radio frequency identification (RFID) systems have been increasingly used in critical domains such as medical field or real-time processing domains. Although important efforts have been made to make this technology more reliable and fault-tolerant, more research is needed to meet the increasing dependability requirements. In this article, we propose a dependability approach consisting in two steps. The first one is an analysis of RFID systems in order to identify potential failures of RFID middleware components and their effects on the whole system. The second one is the modelling of the communication behaviour of RFID systems (that is represented by Low Level Reader Protocol[1]) as a finite state machine. This protocol FSM will be extended to take into consideration the failures identified in the first step and serves as a diagnosis and test tool for the RFID system.**

Keywords - failure modes; FMEA; RFID middleware; reliability; dependability; diagnosis; testing; finite state machine

I. INTRODUCTION

Nowadays, RFID systems are deployed in multiple site scenarios, usually with many readers and sensing devices as a distributed architecture, and often with high fault-tolerance requirements. As an example, a baggage handling system of an airport is presented in Figure 1. This figure shows the importance of the proper functioning of the readers in such systems; i.e. this will avoid a luggage to be loaded into the wrong aircraft (because the reader that guides the switcher is damaged), or a luggage to fall from the conveyor belt and be lost because it is impossible to locate it (tracking readers are damaged).

To manage and process the raw data coming from these various sources (readers, sensing devices…), a new software component called RFID middleware has been integrated in those systems. This component represents the heart of the current RFID systems. So, it becomes crucial to ensure the middleware fault-tolerance to enhance the reliability of the whole RFID system.

The main goal of our research[2] is the definition of on-line testing and fault tolerance facilities for RFID systems. As a first step, we apply the failure modes and effects analysis (FMEA) on a typical RFID system to identify the failures to deal with. Then, we propose to translate the behaviour of RFID systems into a finite state machine. This model takes into consideration the failures identified in the previous step so model-based diagnosis can identify their causes.

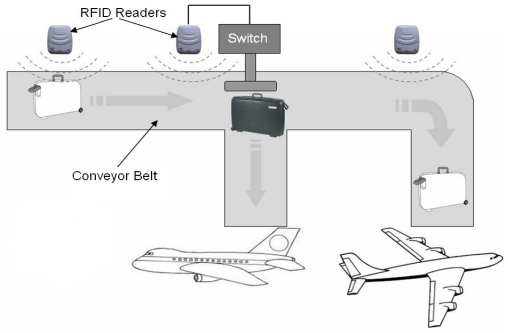

Figure 1. Baggage handling system.

The rest of the paper is organized as follows. Section 2 gives a general description of RFID technology, followed by an overview of RFID monitoring techniques in section 3. Section 4 describes FMEA and its application on the RFID middleware. Then, sections 5 and 6 present respectively the LLRP protocol and the proposed extension to make RFID systems more dependable.

II. RFID SYSTEMS

An RFID system is a contactless technology for identifying unique and remote items using radio waves [1] [2]. It consists of three main components: RFID devices, middleware and user applications as shown in Figure 2.

- RFID devices are tags, readers and other sensors.
- The Middleware (MW) processes the raw data generated by RFID devices. Figure 2 emphasizes the functional components of the middleware as a three-layer communication system that will be the subject of our analysis.
- User applications are programs operating on RFID data processed by the middleware to provide business services.

Let's focus on the functions of the middleware sub-components. The middleware can be represented as a three-layer model with specific functionalities for each layer.

[1] LLRP (Low Level Reader Protocol) is the communication standard between RFID (Radio Frequency IDentification) readers and RFID middleware.

[2] The work presented in this paper is part of the SAFERFID project supported by ANR (French research agency, *www.agence-nationale-recherche.fr*).

978-0-7695-4670-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ICST.2012.138

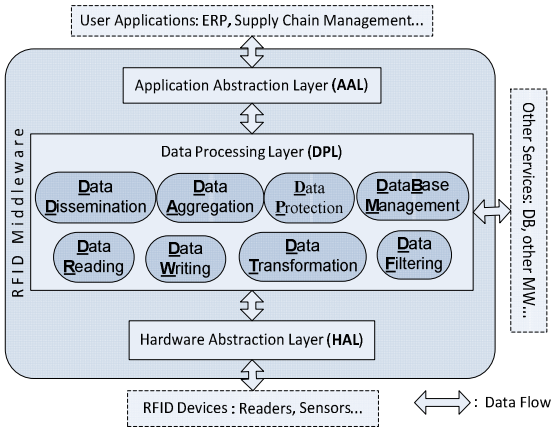

Figure 2. RFID system architecture.

A. Hardware Abstraction Layer (HAL)

HAL is responsible for interfacing operations with hardware components such as:

- Unification of the way the middleware interacts with the hardware components.
- Data filtering on the basis of tag or reader ID.
- Reception of data from various sources, data control and transmission to Data Processing Layer (DPL) or to readers; i.e. it represents a gateway between the readers and the DPL.

B. Data Processing Layer (DPL)

DPL is the main layer of the middleware. It is responsible for most functions of the middleware, such as data filtering, aggregation, etc. The main sub-components of this layer are:

1) Data Dissemination (DD): responsible for data diffusion according to predefined rules.

2) Data Aggregation (DA): responsible for counting and data aggregation according to a given criterion (same type of reader or data intended for the same application, etc.).

3) Data Protection (DP): responsible for protecting and managing application accesses (access control) to data and services provided by the middleware.

4) DataBase Management (DBM): responsible for organizing and sharing the processed data.

5) Data Reading (DR): responsible for collecting data through commands (queries) that it sends to readers.

6) Data Writing (DW): responsible for data writing on the tags.

7) Data Transformation (DT): responsible for processing and formatting of raw data in various formats for greater flexibility.

8) Data Filtering (DF): responsible for removing unnecessary data.

C. Application Abstraction Layer (AAL)

AAL provides user applications with an interface for accessing various services offered by the middleware.

III. Overview of RFID Monitoring Techniques

Many techniques are designed to enhance RFID system's reliability. They are classified in two categories [3]:

A. RFID reader status monitoring

In this kind of monitoring, the middleware checks if the readers are powered and connected to the network, if the reader antennas are operating, etc. This monitoring can easily be performed by SNMP (Simple Network Management Protocol) requests as shown in Figure 3. Therefore, it is considered as ***intrusive monitoring***; i.e. to perform this monitoring, the middleware needs to inject additional data to collect information that is not available from normal operation of the system.

B. RFID reader performance monitoring

The middleware uses the information that is available from normal operation of the RFID system (***nonintrusive monitoring***) in order to process some performance parameters such as read rate of the reader, reading accuracy, error frequency, etc. This technique, unlike the first one, takes in consideration the inaccuracy of RFID readers (the reading accuracy is about 70% [4] and the readers are very sensitive to their environment). So, it is able to detect some failures of the reader according to the variations of a given performance parameter.

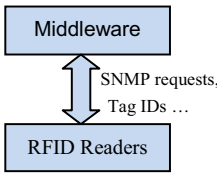

Figure 3. Status Monitoring.

All these techniques only deal with one reader at a time. This usually leads to a poor diagnosis. Indeed, when some tags are not read, it is impossible to check if the tags are faulty or if the reader is broken.

To improve such diagnosis, we study new approaches for on-line testing and diagnosis. As a first step, we have performed an FMEA on RFID systems. In this paper, we show only a part of this FMEA, mainly, the one that is performed on the RFID middleware. FMEA of RFID hardware components has been presented in [5].

IV. Failure Modes And Effects Analysis

Failure modes and effects analysis (FMEA) is a methodology for analyzing potential failure modes, their effects and their severities on a system [6]. It represents a procedure for analyzing possible unreliability problems either

- during development, to take into account the dependability of the product at the earliest stage of its development [7].
- during operation, to ease the maintenance and improve availability of the services provided by the product.

In this FMEA, we assume that the middleware is implemented as a set of layers and "independent" modules (modular architecture), and the list of DPL components is non-exhaustive. Indeed, the middleware's components may differ from one implementation to another. Nevertheless, considered components represent the main services and functions that the middleware is expected to provide.

In the following, we use standard terminology in fault tolerant design [8]: "Fault", "Error" and "Failure". The fault is any cause (event or action such as physical defect) that could cause errors which affect the behaviour of a system or its sub-components (i.e. an error is an internal system state caused by a fault, and possibly leading to a failure).

Figure 4. Cause-Effect relationship.

The FMEA is not easy to perform because there are errors hard to identify or to predict. Indeed, RFID system behaviour is very dependent on its runtime environment (presence of metal obstacles, electromagnetic disturbances, etc.) and the occurrence of an error is often the result of successive events (programming error, misconfiguration, bad use, overuse or component wear, etc.).

In our study, FMEA focuses on the middleware, which is the heart of a RFID system. The application of this technique on the three layers of the middleware described earlier is presented in some tables at the end of this paper. We deliberately omit the presentation of the whole FMEA performed on the RFID middleware, since the purpose is just to show the role this technique plays in our approach (i.e. this technique represents the starting point of our work).

The FMEA of the processing layer is presented in several tables. Each table describes an FMEA of one DPL component. For sake of simplicity, we only present some of them (Tables I and II).

The processing layer is based on a modular architecture, made of a set of components. The main advantage of this type of architecture is that the crash of a component does not break the whole system. But this is not always true. In our case, some components of the DPL are strongly connected, and the crash of one of them causes a chain reaction (i.e. domino effect) that leads to the crash of the system.

Example:

The failure of DF (e.g. the unnecessary data or the redundant data are not filtered) can lead to an overload of DT; i.e. the data flow is blocked or the transmitted data is no longer in the correct format. So, the other components such as DA, DD or DP are not able to process the received data and the system will crash.

The effects of the errors we can observe at this layer may differ depending on the component implementation and configuration. We can illustrate this on the "DP failure". Indeed, the consequences of this failure are related to the enabled security policy. Examples of policies are:

- "Anything not explicitly permitted is prohibited"; consequences: any data flow is blocked (System crash).
- "Anything not explicitly prohibited is permitted"; consequences: the system is still operational but there is a problem of data privacy (i.e. any user can access any data). We can see that this policy is more fault tolerant (high availability) than the first one, but it violates the security of the data. Then we can say that the security policy must be chosen carefully and according to our priorities. This introduces the notion of effect severity; if we consider the availability of this system, then the first policy is more severe than the second one, but if we take into account the security of data, the opposite is true.

Now, if we look at the physical layer, we see that it has almost the same errors than the upper layers but with different consequences and severities. The failure "The action performed is different from that requested" is an example of error with severe consequences and hardly detectable. This is because the system is still operational but with transmission of data different from the requested one.

To make our work compliant with the standard set by EPCglobal [3] in the field of RFID, we assume that the middleware is implemented in compliance to the Low Level Reader Protocol (LLRP) [9] and that it will accommodate our extended model.

V. Low Level Reader Protocol

The communication between the reader and the tags is performed in two phases. The first one is the tag inventory and selection. The second one is the tag access (i.e. read / write access to the different memories of the selected tags).

LLRP is a set of communication rules between the middleware and the reader as it is shown in Figure 5. It works also in two phases: A version negotiation and runtime messages transmission. The latter phase represents the main phase where the useful data is exchanged. This phase covers both phases of aforesaid tag / reader communication. This is done by two types of specifications: Reader Operations Specification (ROSpec) and Tag Access Specification (AccessSpec). These messages (specifications) represent respectively the tag inventory phase and the tag memories access phase.

A. LLRP messages

LLRP messages are XML based implementation. We can classify them into two types:

[3] EPCglobal is an organization whose mission is the development and deployment of EPC (Electronic Product Code) standard and the RFID technology (www.epcglobalinc.org)

1) Messages from middleware to readers: this class of messages includes requests for capability discovery and device configuration, access operations and inventory management that constitute the aforementioned specifications (ROSpec and AccessSpec).

2) Messages from readers to middleware: this category includes states of the readers, acknowledgements, RF-Survey[4], inventory and access operation results.

NB:

- ROSpec has three possible states {Disabled, Inactive and Active}.
- AccessSpec has two possible states {Disabled and Active}.
- The ROSpec is never executed while its current state is not "Inactive".
- The AccessSpec never runs before the ROSpec. It is also never selected for execution while its current state is different from "Active" or if it does not appear in the list of AccessSpec that might be selected by the running ROSpec.

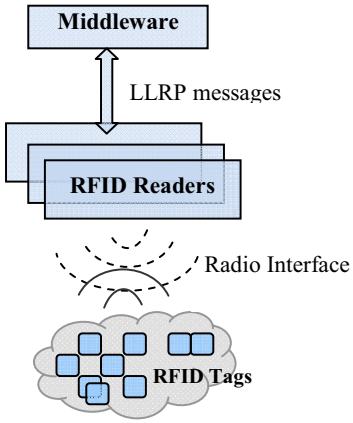

Figure 5. Communication between middleware and readers through LLRP messages.

B. LLRP finite state machine

We have translated the textual description of LLRP into a finite state machine (FSM) that contains 18 states and more than 200 transitions. This model will serve to check if the middleware or the reader conforms to EPCglobal specifications and to perform model-based diagnosis and verification. Let G be this FSM. $G = (S, I, O, \delta, \lambda)$; where I, O and S are respectively finite sets of input symbols, output symbols, and states.

δ: $S \times I \rightarrow S$ is the state transition function.

λ: $S \times I \rightarrow O$ is the output function.

[4] RF Survey is a set of operations where the reader performs a scan and measures the power levels across a set of frequencies at an antenna [9]. (RF Survey is an optional part of an ROSpec).

When the reader (or the middleware) is in a current state s in S, and receives an input a from I, it produces an output specified by $\lambda(s,a)$ and moves to the next state given by $\delta(s,a)$.

Transitions in the LLRP FSM are labeled "X/Y", where X represents the message (usually a request) coming from the middleware to the readers and Y is the message (usually a response to the middleware request) sent by the reader to the middleware.

The execution of LLRP is summarized with two phases. For the rest of this section, we only present some parts of LLRP FSM.

1) Negotiation of protocol version: the middleware and the RFID reader will negotiate a mutually supported protocol version for each session of communication. The middleware can also update the reader configuration once the negociation is done. The following FSM summarizes this protocol version negotiation stage.

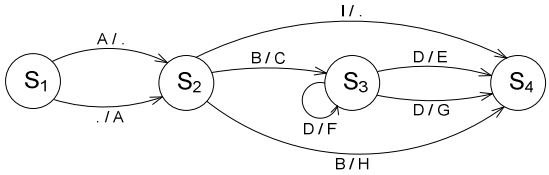

Figure 6. LLRP version negotiation.

This FSM shows the triggering conditions and the different states by which a disconnected reader passes to establish a connection with the middleware. The following tables show details of this automaton.

State	*Meaning*
S_1	The reader is disconnected from the middleware.
S_2	The connection is established between the middleware and the reader.
S_3	The middleware received the list of LLRP versions supported by the readers. (The state of the reader is still unchanged).
S_4	The middleware and the reader agree on the version of LLRP to use.

Transition triggering condition	*Meaning*
A	LLRP connection (always with version 1).
B	Getting protocol supported versions.
C	Lists of supported LLRP versions.
D	Setting protocol version (the middleware chooses the most recent version in the list given by the reader, unless it does not support it, and informs the reader).
E	Reader response about the chosen version.

Triggering condition	*Meaning*
F	Unsupported message by the reader.
G	Unexpected message by the reader (the reader has received a message different from the expected one, and then informs the middleware).
H	Error message that indicates the received message is in unsupported version.
I	This is not a request or a command but just an indication that the middleware implements only the first version of LLRP, so there is no version negotiation.
.	This means "no request or response".

2) Runtime messages transmission: it represents the main phase of LLRP protocol in which the middleware will ask the reader to get and/or update the information stored in the tag memories.

As an example, Figure 7 shows the communication behaviour with only one ROSpec (for a tag inventory) rather than the cases where there are an arbitrary number of ROSpec and AccessSpec.

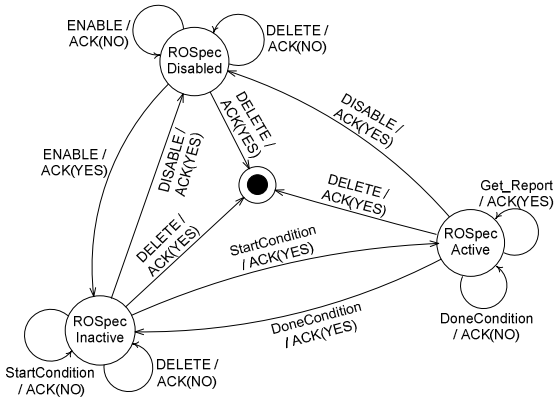

Figure 7. Reader Operation Specification FSM.

A new specification (ROSpec or AccessSpec) is always in a disabled state. In the case of our example, the ROSpec moves to an inactive state when the reader receives correctly an "ENABLE" request (ACK(YES)[5]) from the middleware. The ROSpec is still inactive, waiting for a start condition such as "there is no ROSpec under execution" (i.e. in active state), "the priority of the ROSpec is higher than those of remaining ROSpecs" or "there is an external triggering event" (e.g. events from a motion sensor), etc. The reader can start returning the results of the ROSpec once this latter is active. A running ROSpec returns to inactive state when the inventory is done or if there is another ROSpec with high priority (i.e. in this case, the ROSpec will be preempted). The ROSpec can also be disabled or deleted following a request of the middleware.

LLRP is a complex and complete communication protocol with communication error handling by Cyclic Redundancy Check (CRC), error notification messages and acknowledgements. But this protocol is not designed to diagnose the cause of these errors or failures [9]. So the main idea of our approach is to extend the LLRP model including failures identified earlier in the FMEA. This, and the observed outputs, will lead us to the causes of the failure.

VI. Extending LLRP For Model-Based Diagnosis

A. LLRP failures

RFID readers are inherently inaccurate (i.e. the reading accuracy is about 70% [4], the reader-tag interface is very sensitive to the presence of other radio waves and obstacles such as water, metal, etc.). This usually leads to the transmission of erroneous data; i.e. the failures that concern the state of the transmitted data are more likely to arise. For this reason, the extended LLRP automaton will mainly focus on the detection of such failures. However, we explain below how to use this extended FSM to detect other failures such as "masking of useful data" that are not necessarily due to external causes.

The set of failures that we are interested in can be classified according to their causes into two categories.

1) Failures due to inconsistent design

- Masking of useful data or even blocking all data by the middleware.
- Mismatch between the received and the expected data by the middleware or the reader.

2) Failures due to runtime conditions

- Slow execution / component overload.
- No data capture.
- Erroneous received data.

B. Using the existing LLRP model

The first category of failures is handled without any need to extend the existing model, using "distinguishing sequences". A distinguishing sequence is a set of transitions that allow identifying the initial and current state of the reader or the middleware. For more details about this technique, see [10][11].

To illustrate this principle, let's take the following example. It represents a three-state FSM, with two inputs {A, B}, and two outputs {0, 1}.

[5] ACK(YES) is an acknowledgement message to confirm the correct reception or interpretation of the request. Contrariwise, ACK(NO) means there are errors in the received message, or the received request is not expected…

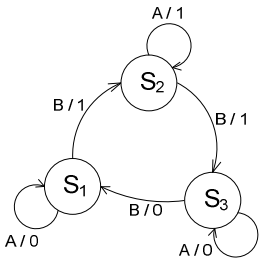

Figure 8. Example of an arbitrary FSM.

At the beginning, and after the onset of the failure, we assume that we don't know the internal state of the reader or of the middleware; i.e. the FSM can be in any state. So, we construct the successor tree[6] of this FSM that will serve as a decision tree to identify the distinguishing sequences.

Figure 9 shows the successor tree built up from the FSM of Figure 8. We can see that the input A separates the state S_2 from S_1 and S_3. Then the input B separates S_1 from S_3. For example the sequence {A/1} is a distinguishing sequence for S_2 (i.e. after applying the input A and observing an output 1, we are sure that the current and the initial state of the machine are S_2), and the sequence {A/0, B/0} is a distinguishing sequence for S_3; in the tree successor, this sequence leads to S_1. So, in the FSM of Figure 8, when we go back from S_1 with the reverse sequence of previous distinguishing sequence (i.e. {B/0, A/0}), we will end up at S_3 (that represents the initial state).

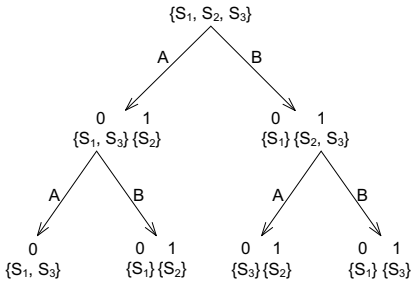

Figure 9. The successor tree of the FSM.

A distinguishing sequence should always lead to a single final state. Thus, the sequence {A/0, A/0} is not a distinguishing sequence because it does not separate the state S_1 from S_3 and we can't go back to deduce the initial state. This is not the only condition to construct distinguishing sequences. The other one is that the reverse of the constructed sequence must also lead to a single state. The following FSM shows this difficulty that may be encountered when building up distinguishing sequences. Thus, all sequences that start with the transition A/0 in Figure 10 can never be distinguishing sequences, since their reverse sequences are non-deterministic; (i.e. we can never be able to deduce whether the FSM started in S_2 or S_3).

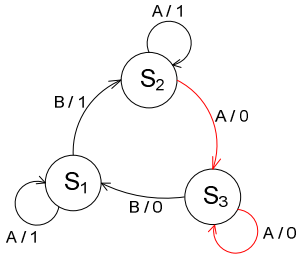

Figure 10. Difficulty of constructing distinguishing sequences.

C. *Extending the LLRP model*

The second category of failures requires an extension of the existing model. The principle of this extension is shown in Figure 11: when the communication between the reader and the middleware is running properly, the reader and the middleware pass only by the states of the LLRP FSM (the left cloud shape); when a failure occurs, both reader and middleware switch to another state in the LLRP extension to diagnose the reasons of this failure.

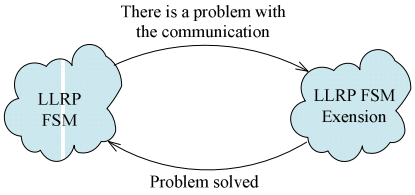

Figure 11. Extended LLRP FSM.

For sake of readability, the extended model is not presented as an FSM, but as a diagnosis tree. In the following subsections, we explain how to use it to build test sequences that detect the aforementioned failures and their causes with some examples.

1) Diagnosis of inconsistent design failures: The failures of this category are due to the fact that one of the participants (middleware or reader) in the communication is in a state other than the expected one; i.e. the middleware (or the reader) is in a state where it does not expect such type of received commands or data (e.g. result of a tag inventory). Then, as a consequence, it ignores them (although they are useful data) and may return an error message.

To diagnose this king of failures, we apply the technique of "distinguishing sequence". For example, to detect the cause of the failure "Masking of useful data (e.g. the tag memory content) by the reader" (i.e. the middleware receives only the Tag Identifiers), we will inject commands in the FSM of LLRP, and by using the previous technique, we show that it is due to a wrong internal state of the reader; i.e. the AccessSpec is not executed to access the tag memories. At the end of the diagnosis, we show that the reader's current state was one of the following:

[6] A successor tree shows the behaviour of an FSM starting from all possible states under all possible input sequences [11].

a) Case 1: ROSpec is "Active" but not the AccessSpec.

b) Case 2: ROSpec and AccessSpec are "Active" but the AccessSpec is not in the list of AccessSpec to be triggered by the running ROSpec.

c) Case 3: ROSpec and AccessSpec are "Active" but the identified tags do not satisfy the triggering conditions of the AccessSpec.

2) Diagnosis of runtime conditions failures

a) The failure **"**Slow execution**"** (which can lead to **"**component overload**"**) may be caused by the amount of raw data to process (e.g. too many readers) that leads the middleware to slow down. We shoud know that the middleware associates to each new reader an instance of the LLRP FSM to always maintain consistent communications with it. So, to validate that the source of this problem comes from the huge amount of data that are fed back to the middleware, we just need to read all the instances of LLRP FSM looking for all the readers that are in a communicating state. Then, it only remains to verify that the number of the current communicating readers does not exceed the capacity of the middleware. If it does, as a possible solution, the middleware will ask the readers which have internal memories to pause the communication and to temporary store the data in their memories.

b) The failures "No data capture" and "Erroneous received data" are the most likely to arise. These failures are caused by the same reasons as the first kind of failures discussed earlier, but they may also be due to other reasons:

- The reader antenna is damaged or broken.
- The presence of external disturbances (radio waves of other devices, water, metal objects…).
- The tags are out of the reader scope or the signal power level of the reader is too weak.
- The tags move too fast.
- The type of tags is not supported by the reader or on the contrary, the selected air protocol is not supported by the tags.

The idea is to include these causes of failures in an extended LLRP FSM to make the system more dependable. For sake of simplicity, Figure 12 shows a part of the LLRP extension as a flowchart. The lozenges represent the inputs in the FSM, and the leaves (orange circular shapes) are the causes of the failure.

As we can see in Figure 12, when the reader is executing an ROSpec with an AccessSpec (ROSpec and AccessSpec are active), the middleware may ask it for the result of these specifications. If the report is incomplete (there are tags that are not inventoried correctly, impossible tag memories read / write …), the middleware triggers the diagnosis process. The failure causes are hierarchically sorted according to their diagnosis difficulties. So a "broken antenna" is easier to diagnose than a "faulty communication link".

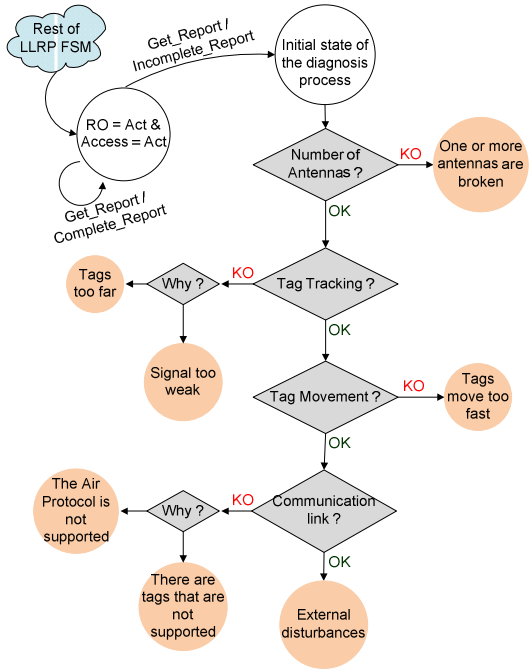

Figure 12. LLRP Diagnosis Tree.

- The middleware can verify if an antenna is damaged or broken by asking the reader to give it back the number of operational antennas, or by setting an RF Survey if supported on all antennas. The RF Survey is useful for two reasons. First, it allows the middleware to know the number of antennas that are operating, so it only remains to compare this number to the previous number of the operating reader's antennas. Second, the middleware will know with this survey the signal power levels supported by each antenna. This is helpful to verify that the reader did not use a signal whose power exceeds the capacity of the antenna. If the reader is not able to perform RF Survey and knowing that most readers have at least two antennas (e.g. one for transmission and one for reception), the middleware can transmit a signal by an antenna to receive it by the other one. If this operation succeeds, we assume that the antennas are not broken. To verify if one of them is damaged, the middleware will vary the receive sensitivity[7] or the signal power level of the antenna to process the scope of the antenna (a damaged antenna will cover smaller area), and so deduce if the antenna is damaged or not[8]. If there are more than two antennas, the middleware will perform the above technique on all pairs of antennas.

[7] Receive sensitivity indicates how faint a radio frequency signal can be successfully received by a given receiver. The lower the power level that the receiver can successfully process, the better the receive sensitivity [12].

[8] In this technique, the middleware must know in advance the distance between the antennas to perform a correct diagnosis.

- For the second cause of failure (i.e. "the tags are out of the scope of the antenna"), the middleware will locate the tags by varying the power level or the receive sensitivity of the antenna as explained above. Figure 13 shows an area containing four readers, and how to exactly locate a tag to verify if it is out of the scope of a reader.

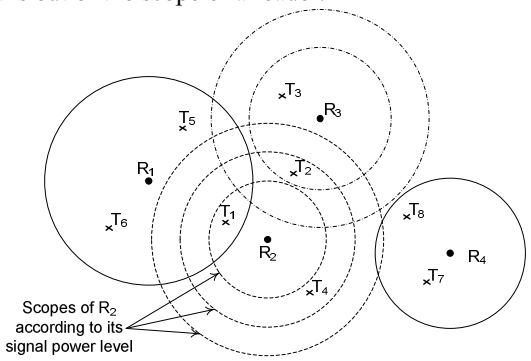

Figure 13. Tag localization technique.

For example, to verify that the tag T_2 is too far from reader R_1, the middleware will exploit the neighbours of R_1 to locate the tag T_2 by reducing the scope of these readers enough to deduce that T_2 is out of the scope of R_1.

- To deduce that the failure is due to tags that move too fast, the middleware can proceed as follows:
  - If the tags are carried by a conveyor belt, as it is shown in Figure 14, the middleware has just to know the speed of the conveyor belt and to compare it with the maximum speed supported by the reader.

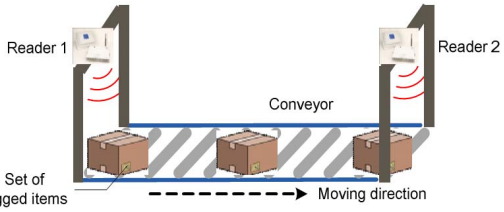

Figure 14. Moving items on a conveyor belt.

  - The second possibility is to calculate the speed of the tagged item using its two previous positions (or three, to calculate the tag speeding).

For example, if reader R_1 in Figure 15 reads the tagged item at 11:14:25, and R_2 reads it at 11:14:35, the middleware can calculate the item speed (0.5 m/s), so it knows when this item will probably appear in the scope of R_3 and with what speed will pass it. The middleware can also deduce the tag speeding (if the tag speed is changing) by knowing the spent times between (R_1 and R_2) and (R_2 and R_3).

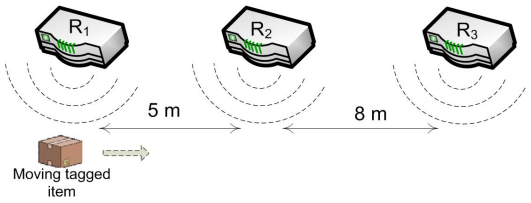

Figure 15. Tag speed calculation.

- To detect the failure of the communication link, we proceed as follows:
  - If erroneous data are received by the middleware, than we are sure that the problem comes from the communication link between the reader and the middleware otherwise the erroneous data will be filtered by the reader (for example, this will result in an incomplete tag inventory).
  - To verify that the problem is coming from the communication link between the reader and the tags, the middleware will ask the neighbours of the concerned reader which air protocol they used to successfully identify the tags. So, the middleware can deduce that the reader has used a non-supported protocol. Otherwise, the middleware will check if the reader does not support the concerned tags (e.g. UHF reader that try to read HF tags). If none of these causes are found, we assume that the failure is due to external disturbances such as the presence of metal objects between the reader and the tags (wave backscattering phenomenon).

We can also note that failures such as "impossible tag inventory or tag memory access" may come from the fact that the maximum number of ROSpec or AccessSpec supported by the reader is reached. So the middleware has just to delete the old specifications to add other ones. We have intentionally passed over such failures because we assume that before adding a specification, the middleware will check the maximum number of specifications that can be set in the reader. This is possible with a specific LLRP command "GET_READER_CAPABILITIES".

VII. CONCLUSION

RFID systems are increasingly used in areas where dependability is a key factor. Since we can never be sure that a piece of software is safe and reliable, it is necessary to define or apply methods in program design, as well as introduce test and fault tolerance features at runtime. In this paper we presented an ongoing work aiming at providing model-based testing facilities focusing on the RFID reader's failure diagnosis. We have proposed a formalization of the

LLRP protocol as a FSM and we have shown that usual test sequence generation techniques can deal with some failure detection. Then, we presented an extension of the LLRP protocol and the associated model and explained how it can be used to generate test sequences able to detect more failures and their causes.

The proposed approach is not yet mature. It deals only with some failures of RFID systems. So, in future work, we will improve the extended LLRP FSM to include more failure categories and we will fully implement automatic test generation. This will build a set of transition sequences for all causes of the identified failures; i.e. each sequence will identify a cause of the failure. So, if a transition sequence is entirely and successfully executed on the faulty system, we can deduce easily the cause of the failure according to the executed sequence.

REFERENCES

[1] P. Krishna and D. Husak, "RFID infrastructure", IEEE, Applications & Practice, Vol. 45, Issue 9. Canada. September 2007.

[2] T. Hassan and S. Chatterjee, "A Taxonomy for RFID", IEEE, the 39th Hawaii International Conference on System Sciences, 2006.

[3] S. Paul, T. Frank, H. Brad, K. M. F. Francesco, K. John, M. Das Anand, B. Hersh and C. Anita, "How to Cheat at Deploying and Securing RFID", Elsevier, Inc. 2007, ISBN 13: 978-1-59749-230-0.

[4] R. Derakhshan, M. E. Orlowska, and X. Li, "RFID data management: Challenges and opportunities", IEEE international Conference on RFID, USA, 2007.

[5] G. Fritz, V. Beroulle, O. Aktouf, M. D. Nguyen and D. Hély, "RFID System On-line Testing Based on the Evaluation of the Tags Read-Error-Rate", SpringerLink, Journal of Electronic Testing, 2010.

[6] M. L. Shooman, "Reliability of computer systems and networks : Fault Tolerance, Analysis, and Design", Wiley-Interscience Publication, 2002. ISBN: 0-471-29342-3.

[7] L. Cauffriez, J. Ciccotelli, B. Conrard and M. Bayart, "Design of intelligent distributed control systems: a dependability point of view", Elsevier, Inc. Journal of Reliability Engineering and System Safety, April 2004, pp. 19-32.

[8] D. K. Pradhan, "Fault-tolerant computer system design", Prentice Hall, February 1996. ISBN: 0-13-057887-8.

[9] EPCglobal, "Low Level Reader Protocol (LLRP) version 1.1", EPCglobal, Inc. 2010.

[10] F.C. Hennie, "Fault detecting experiments for sequential circuits", in Proc. FOCS, 1964, pp. 95-110.

[11] L. David and Y. Mihalis, "Principles and Methods of Testing Finite State Machines – A Survey", Proceedings of the IEEE, Vol. 84, Issue 8. August 1996, pp. 1090 - 1123.

[12] A. W. David, D. C. David, M. Ben and M.Peter, "CWAP - Certified Wireless Analysis Professional Official Study Guide: Exam PW0-270", Wiley publishing Inc., Indianapolis, Indiana, 2011, ISBN: 978-0-470-76903-4.

TABLE I. FMEA OF DATA READING COMPONENT

Failure modes	Possible causes	Effects on the system	Proposed Solutions
Mismatch between the received and the expected data.	- Triggered command by DR does not match the demand of the AAL or the action performed by the reader. - Improper command interpretation / adaptation by the HAL.	- Inconsistent product management.	- DR, HAL or reader driver code review (with regression testing).
Erroneous received data.	- Error in HAL code. - Error in DR code. - Recent addition of not supported hardware. - The communication link is failing (e.g. disturbance due to the environment).	- System crash.	- DR or HAL code review. - MW code review to support the unsupported reader. - Regression testing. - Interference suppression. - Replacement of the communication link.
No data capture.	- Capture devices, DR or HAL crash. - Unsupported reader protocol. - Application queries unrecognized (new application, commands that are not implemented, etc.).	- System crash.	- DR or HAL code review for error correction and to support the new hardware components, the new application or the reader protocol. - Regression testing. - Replacement of faulty equipment or the one not supported by the MW.
Slow execution / DR overload.	- DR source code not optimized. - Low performance machine. - Host machine slowed down by other processes. - Huge amount of data to process (too many applications or readers).	- System performances slowed down / System crash.	- DR Code review for optimization with regression testing. - Replacement of the host machine by a high performance one. - Reduction of the number of processes on the host machine. - Reduction of the number of applications or readers. - Load / Stress Testing.

TABLE II. FMEA OF DATA PROTECTION COMPONENT

Failure modes	Possible causes	Effects on the system	Proposed Solutions
No access to data and MW services by an application.	- DP misconfiguration. - Errors in DP source code.	- System performances slowed down - Loss of information.	- DP reconfiguration. - DP code review. - Regression testing.
Access control failure (application accesses prohibited data and services).	- DP misconfiguration. - Errors in DP source code.	- Problem of data privacy.	- DP reconfiguration. - DP code review. - Regression testing.
DP crash.	- Errors in DP source code.	- System performances slowed down / System crash. - Loss of information.	- DP source code review. - Regression testing.
Slow execution / DP overload.	- Low performance host machine. - DP source code not optimized. - Host machine slowed down by other processes. - Huge amount of data to process.	- System performances slowed down / System crash. - Loss of information.	- Replacement of the host machine by a high performance one. - DP Code review for optimization / Rregression testing. - Reduction of the number of processes on the host machine. - Reduction of the number of entities that request DP services. - Load / Stress Testing.

TABLE III. FMEA OF PHYSICAL LAYER

Failure modes	Possible causes	Effects on the system	Proposed Solutions
HAL indicates that the received data is erroneous (but the data is correct).	- Errors in HAL code. - HAL misconfiguration (a reader driver is not loaded when needed). - Communications link failure between MW and reader or disturbance of transmission on the medium. - Unsupported hardware or protocol.	- System performances slowed down / System crash. - Loss of information.	- HAL source code review. - Regression testing. - HAL Reconfiguration.
HAL crash.	- Error in the HAL source code. - Too much data to process.	- Impossible read/write operations. - System crash. - Loss of information.	- HAL code review for errors fixing and improving of its performances. - Regression testing. - Reduction of the number of applications / readers.
HAL receives data but does not provide any output.	- Unsupported received data from the DPL (components incompatibility). - Unrecognized communication protocol. - Erroneous translation of the DPL command for the reader.	- System crash.	- Integration and compatibility testing. - Improvement of the MW to support the communication protocol. - HAL code review. - Regression testing.
The performed action is not that requested.	- Erroneous translation of the DPL command for the reader.	- Inconsistent product management. - System crash.	- HAL code review. - Regression testing.
Slow execution / HAL overload.	- Low performance machine. - HAL source code not optimized. - Huge amount of data to process (too many applications or readers). - Host machine slowed down by other processes.	- System performances slowed down / System crash. - Loss of information.	- Replacement of the host machine by a high performance one. - HAL Code review for optimization. - Regression testing. - Reduction of the number of entities that request HAL services. - Reduction of the number of processes on the host machine.