MODELING A RFID AND AGENT-BASED SHOP FLOOR CONTROL SYSTEM FOR DRAM TESTING FIRM BY UML

HSIEN-PIN HSU

Department of Logistics Management
National Kaohsiung Marine University, Kaohsiung, Taiwan
E-mail: hpsu@webmail.nkmu.edu.tw

ABSTRACT

Due to centralized and hierarchical structure, many shop floor control systems (SFCSs) currently used in dynamic random access memory (DRAM) testing firms are facing with problems, such as with difficulties in data collection and to integrate with legacy systems. To deal with these problems, advanced technologies such as Radio Frequency Identification (RFID) and agent-based approach appear capable and applicable. In this study, a software process based on Unified Modeling Language (UML) is proposed, and some results such as use case diagram, class diagram and a RFID and agent-based system architecture are presented. These results can help DRAM testing firms improve their outdated SFCSs.

Keywords: Radio Frequency Identification (RFID), agent-based approach, shop floor control system (SFCS), Unified Modeling Language (UML)

1. INTRODUCTION

Dynamic Random Access Memory (DRAM) has been used in various electronic devices. Ensuring the functionality of DRAMs is thus critical. In a DRAM supply chain, DRAM testing is the final process to ensure the functionality and quality of DRAMs before shipments. A good control on the DRAM testing process is thus necessary. However, this is difficult to achieve, even a shop floor control system (SFCS) is used to help.

There are three difficulties hindering the achievement of a good control on DRAM testing. The first one comes from the dynamic nature of the DRAM testing process. During DRAM testing, the testing process of a lot needs to dynamically adapt to the immediate testing results of the lot. The second comes from the unexpected interrupts, such as machine breakdowns, arising from the shop floor. The third comes from the fact that a mountain of testing lots tend to appear in the shop floor. The huge amount testing lots raises the difficulty of performing relevant shop floor activities, including data collection, lot management and process control. This indicates that the shop floor control system (SFCS) used in the DRAM testing firm should be powerful enough. However, many of the SFCSs used in DRAM testing firms are found to have encountered problems, such as incapable for data collection and hard to integrate with legacy systems due to their centralized and hierarchical architectures [1]

To enhance the power of a SFCS, two advanced technologies, Radio Frequency Identification (RFID) technology and agent-based approach are believed to be useful and applicable. The advantage of RFID technology is that it can identify an object, including a testing lot, accurately and quickly. Due to this advantage, RFID technology has been employed in various areas, such as retailing, animal identification, hospital, logistics and supply chain management [2]. However, it is noted that RFID technology has not been widely used for SFC [3]. In contrast, the main advantage of the agent-based approach is that it can facilitate the development of a distributed system due to the fact that agents can be distributed on different computers so as to accomplish tasks autonomously or cooperatively. Another advantage of the agent-based approach is that it enables a system to be more adaptable to dynamic environment when agents are empowered by intelligence [4]. Moreover, agents can resolve legacy problem when they are cooperating with middleware. Though agent-based approach appears promising, however, there is still a lack of a formal tool and software process for the agent-based approach.
To address this issue, a software process based on a modeling tool, Unified Modeling Language (UML), is proposed in this study. UML is a set of object-oriented (OO) modeling notations that has been standardized by the Object Management Group (OMG) [5]. Based on the OO concept, UML can link system analysis (SA), system design (SD) and programming [6,7] together so that these tasks can be performed more efficiently. In addition, UML can provide a good basis for agent-based system modeling [8]; Especially, UML 2.0 offers a firmer foundation [4].

Our literature review shows that RFID technology and agent-based approach have never been simultaneously used to improve the SFCS for DRAM testing firms. This finding has prompted us to initiate a RFID and agent-based SFCS for DRAM testing firms. The rest of this study is organized as follows. Section 2 introduces a proposed software process useful for developing a RFID and agent-based SFCS. Section 3 presents some results derived from the proposed software process. Finally, Section 4 gives the conclusion.

2. A SOFTWARE PROCESS

Though UML is a useful tool for system modeling, it cannot be regarded as a software methodology due to the lack of a software process, which has to include details about development activities, dependencies of these activities and how they are applied [4]. To addresses this issue, a software process is outlined as follows.

Step 1. Process flow analysis: this step is to understand the application domain. User interview can be conducted and flow chart with descriptions can be used to detail the process flow.

Step 2. Requirement analysis: this step is to acquire the user requirements for the SFCS to be used in DRAM testing firms. User interview can be conducted to derive user requirements, which can be transformed into an event table to derive classes.

Step 3. Class analysis: this step is mainly to acquire classes for modeling a SFCS. Using noun-oriented approach, nouns that first appear in the requirement description are of interested and are collected. These nouns are candidates of classes/attributes. The class diagram of UML will be used in this step.

Step 4. Agent derivation: this step is mainly to derive agents to form an agent-based system architecture. Classes derived in Step 3 can be transformed into agents.

Step 5. Agent collaboration analysis: this step is to detail the collaboration and interactions among agents. Communication diagram can be used in this step.

Step 6. System architecture design: this step is mainly to design the architecture for the agent-based SFCS. System architectures built from agents can have three types: hierarchical, federated and distributed [9]. In the hierarchical architecture agents can only communicate with their direct supervisors. In the distributed architecture, agents are deployed on different computers and they can communicate to each other via network. In the federated architecture agents cooperate with others through a host agent. For better communication, middleware can be treated as one agent. In addition, in a system architecture, an agent can be attributed as a static or a mobile one.

3. THE DERIVED RESULTS

Following the software process proposed in Section 2, we present the derived results in the following subsections.

3.1 Process Flow Analysis

Figure 1 shows the testing process, which is clarified as follows.

For better management, DRAM dies in the shop floor are grouped into lots, each assigned with a unique lot ID. After the lead scan operation, a lot first goes through a Burn-In (B/I) for 24 hour, and then it moves to a low temperature test (FT2). However, before start FT2, the check on waiting time for each lot is required. If the waiting time of a lot for FT2 exceeds 80 hours then the lot needs a re-B/I. After FT2, a judgment is made for each lot. If a lot is judged “passed” then it will be directed to Quality Control 1 (QC1); otherwise, it will be held for further verification. QC1 is a retest operation conducted by QC operator to ensure the lot to behave normally under low temperature condition (-20°C). If the lot fails QC1 then it will be further verified by engineer, and then a decision of “hold”/”re-FT2” may be made. The lot passes QC1 will be moved to Variable Retention Time (VRT), a
test including two minor tests, VRT1 (150°C) and VRT2 (75°C). During VRT2 a speed test is simultaneously performed. After VRT2, a check is made and then the next step the lot to go is determined; if the lot passes the yield condition, VRT(Bin3)<1%, it will go to FT3 (a high temperature test); otherwise, it will to the VRT3 test. The term “Bin3” means the quality of level 3, which is an indicator based on the speed performance of the tested dies. After FT3, dies of a same bin will form a new sub-lot, which will be later moved to Lead Scan (L/S) for inspection. Following this, the sub-lot will go through Mark (M/K) operation, QC2 check, Visual Inspection (VI) and then the Bake (B/K) operation, after which the sub-lot will be moved to Packing (P/K) operation, where sub-lots of a same mother lot are merged into a shipping lot. When a shipping lot passes the Final Quality Assurance (FQA) operation, it will be moved to Finished Good Stock (FGS) for shipment.

3.2 Requirement Analysis

After the step 2 of the proposed software process, the requirements for the SFCS are listed as follows:

- **Operator** (OP) can write a **lot ID** into a **RFID tag**, which represents a lot in shop floor (write lot ID function). A lot includes a batch of **dies** (lot size) and these dies have a same **part ID**. Each part has a **Part types** specified by **orders** placed by **customers**.
- Each part is assigned with a testing **process**, which is composed of a series of **steps**. In the **Engineer system** it designates a testing process for each lot.
- An **OP** can track a lot through the RFID tag that represents the lot (track lot function).
- An **OP** can “check in” a lot (check in function) at each step. In addition, a **dispatcher** is required to assign resources (such as **tester**, **ovens**), etc. to the check-in lot. During checking in a lot, an **OP** can input data such as **badge no** and **machine ID** (entry lot data function)
- An **OP** can “check out” a lot (check out function) at each step. During checking out a lot, OP can input relevant data such as **defect codes** and **defect quantities** (entry lot data function). Especially, during checking out a lot from FT3, data collection for the quantity of dies in each **bin** is necessary. Dies of a same bin are split from the mother lot (split lot function) and forms a new **sub-lot**, with the **sub-lot number** being automatically assigned.
- When checking out a lot, the system must display the next step for the lot to go (routing function).
- The **system** should integrate with **order system** (OS), a sub-system of Enterprise Resource

---

**Diagram:**

- **VRT2(Bin3)<1%** → **Judge** → **VRT2(Bin3)>=0.5%**
- **Yield < 90% or VRT2(Bin3)<3%** → **Hold**
- **MBT3(VRT3)**
- **VRT3(Bin3)<1%** → **VRT3(Bin3)=0%**
- **VRT3(Bin3)>3%** → **Inform Engr/WB**
- **VT2(HT & Bin Sort)**
- **Pass** → **Fail**
- **QCC(01)**
- **Pass** → **Fail**
- **Mark**
- **No** → **Yes**
- **Inform Engr/WB**
- **DCY1(Bin5)=0% or FCY1(Bin5)=0%**
- **Inform Engr/WB**
- **VT1**
- **Fail** → **Pass**
- **Verify**
- **Pass** → **Fail**
- **Inform Engr/WB**
Planning system (ERP) (connect to ERP function), which keeps relevant order data, such as order number and part name, etc.

- The system can access the process data of a part stored in the Engineer system (connect Eng system function).
- Both OP and Quality Control Operator (QCO) can hold a lot if required (hold function).
- OP and QCO can release a lot (release lot function).
- OP and QCO can query the status of a lot (such as normal, hold, current step, location, etc) (query function).
- Before checking in a lot to the step FT2, the system has first to check for the waiting time of the lot. If the waiting time of the lot exceeds 80 hours then the lot needs a re-burn in. (routing lot function).
- The system can monitor the status of the machines on-line and real time. (machine monitoring function).

Table 1 shows the derived event table, in which the first column specifies the associated events; the column “triggered by” specifies the data required to trigger a specific event; the column “activity” specifies the activity to be performed after triggering a specific event; the column “response” specifies what message(s) to be prompted for a specific event; finally, the column “destination” specifies the destination(s) to where the message(s) will deliver.

<table>
<thead>
<tr>
<th>Events</th>
<th>Triggered by</th>
<th>Activity</th>
<th>Response</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiate lot ID</td>
<td>Store Lot ID</td>
<td>Lot ID stored into tag</td>
<td>OP/QCO</td>
<td></td>
</tr>
<tr>
<td>Track lot</td>
<td>Display lot status</td>
<td>Lot status</td>
<td>OP/QCO</td>
<td></td>
</tr>
<tr>
<td>Check in lot</td>
<td>Update lot data</td>
<td>Check in completed</td>
<td>OP</td>
<td></td>
</tr>
<tr>
<td>Check out lot</td>
<td>Update lot data</td>
<td>Check out completed</td>
<td>OP</td>
<td></td>
</tr>
<tr>
<td>Entry lot</td>
<td>Add lot data</td>
<td>Lot status</td>
<td>OP</td>
<td></td>
</tr>
<tr>
<td>Hold lot ID</td>
<td>Update lot data</td>
<td>Lot held</td>
<td>OP/QCO</td>
<td></td>
</tr>
<tr>
<td>Release lot ID</td>
<td>Update lot data</td>
<td>Lot released</td>
<td>OP/QCO</td>
<td></td>
</tr>
<tr>
<td>Query lot</td>
<td>Display lot data</td>
<td>Lot status</td>
<td>OP/QCO</td>
<td></td>
</tr>
<tr>
<td>Trace lot</td>
<td>Display lot data</td>
<td>Lot status</td>
<td>OP/QCO</td>
<td></td>
</tr>
<tr>
<td>link to ERP</td>
<td>Access ERP system</td>
<td>ERP system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link to Eng</td>
<td>Access Eng system</td>
<td>Eng system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stores data</td>
<td>Data stored transaction</td>
<td>DB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dispatch lot ID</td>
<td>Assign and show</td>
<td>Resource ID</td>
<td>OP</td>
<td></td>
</tr>
<tr>
<td>Monitor Resource</td>
<td>Update resource</td>
<td>Resource ID</td>
<td>System</td>
<td></td>
</tr>
</tbody>
</table>

3.3 Class Analysis

After the step 3 of the proposed software process, the nouns are derived and some of them are transformed into classes while others are transformed into attributes. Figure 3 shows the class diagram, which is clarified as follows.

A lot can be split into multiple sub-lots, each with a RFID tag. Once a RFID reader has detected a lot (or sub-lot), a lot-related event is initiated with the event ID being sent to the class lot management immediately. Through the message bus, class lot management can communicate with others, such as Order system, dispatcher, Eng system, resource and part. The communication between classes resource event and resource management is direct, bypassing the message bus.
3.4 Agent Derivation

After the step 4 of the proposed software process, agents are derived as follows.

- **Lot management agent (LMA):** a static agent who cooperates with LA to process lot-related events. Each lot-related event is assigned with an event ID. According to the event ID, the LMA can display corresponding operating menu to operator.

- **Part agent (PA):** a static agent who is responsible for maintaining part data, such as part ID, part type, customer ID, etc. PA cooperates with LMA to perform “check in” and “check out” transaction for a lot.

- **Process agent (PRA):** a static agent who controls testing process. PRA can determine the next step a lot to go, based on its testing process and current testing results.

- **Dispatching agent (DA):** a static agent who dispatches resources (such as testers and ovens) to a lot during checking in.

- **Device event agent (DEA):** a static agent who monitors the device-related events. When a device-related event, such as machine breakdown, is detected, the corresponding event ID will be passed to RA. The devices commonly used in a test firm include testers, ovens, etc.

- **Resource agent (RA):** a static agent who monitors and coordinates with DEAs so as to handle the device-related events reported by DEA. For example, when a machine breakdown event is detected, RA may reassign another machine.

- **Lot agent (LA):** a mobile agent who first resides on a RFID tag, but to be uploaded to computer when the TRID tag has been scanned by a RFID reader. An uploaded LA will cooperate with LMA to deal with lot-related events such as “check in”, “check out”, etc.

- **Database agent (DBA):** a static agent who manipulates the data stored in database. DBA also supplies the data needs for agents via the message bus.

- **Middleware agent (MA):** an agent who is responsible for communication and data delivery.
3.5 Agent Collaboration

3.5.1 Check in a lot

Figure 4 shows the communication diagram for agents/objects to “check in” a lot. Once a RFID reader has scanned a tag, the LA agent together with relevant lot data (including Part_ID, Lot_ID, Route_ID, Lot_Qty, etc.) will be uploaded to a computer, so that the LA can cooperate with the LMA to display the “check in” menu to an operator. After the operator entering lot data and confirming the “check in”, the LMA will call on the DA for resource dispatching. Following this, the “check in” data are delivered to DBA for storage. In this use case, the LA communicates to the LMA directly so as to alleviate the workload of message passing for the MA.

3.5.2 Check out a lot

Figure 5 shows the communication diagram for agents/objects to “check out” a lot. Once a RFID reader has scanned a tag, the acquired lot data (such as Part_ID, Lot_ID, Route_ID, Lot_Qty) together with the LA will be uploaded to a computer, so that the LA can cooperate with the LA to display the “check out” menu to an operator. After the operator entering data and confirming the “check out”, the LMA will call the PRA to route the lot to the next operation. Finally, “check out” data are delivered to DBA for storage. In this use case, the communications among agents are via the MA.
3.6 System Architecture

Usually, an enterprise owns numerous systems. These systems can be classified into three levels: enterprise, shop floor and device. Systems, such as the Enterprise Resource Planning (ERP) system, in the enterprise level provide general enterprise functions. Systems in the shop-floor level are mainly used to control shop floor activities. SFCS is the main system used in shop floor and is the focus of this study. Finally, physical devices, such as machines or workstations, are those belonging to the device level. For data exchange among systems in various levels, physical connections among these systems are necessary.

After the step 6, Figure 6 shows the architecture of the SFCS for DRAM testing firms. It is a hybrid system architecture, in which agents are organized into hierarchical and distributed relationships simultaneously. For instance, RA and DEA are in a hierarchical relationship while RA, LA, DA, DBA, PA and PRA are in a distributed relationship. In addition, we further attribute each agent to be a static or mobile one. In this architecture, LAs are characterized as mobile agents which initially reside on RFID tags but to be uploaded to servers when they are scanned by RFIF reader and a lot-related events occur.

Figure 6: The architecture of the SFCS

Figure 7 shows the deployment of these agents on various servers. As shown in Figure 7, DA resides on dispatching server while the DBA stays on DB server. LMA server is the platform where LMA can cooperate with LAs to process lot-related events. As a mobile agent, LA initially resides in a RFID tag, but to be uploaded to the LMA server once the RFID tag of the lot has been scanned by a RFID reader. To share the workload in the shop floor environment, multiple LMA servers can be simultaneously installed in a production environment. In another arrangement, RA and DEA are managed to reside on a same computer so as to communicate directly. This arrangement alleviates the network workload. RA and DEA play as static agents in this system architecture.

4. CONCLUSION

The dynamic nature of DRAM testing together with the unexpected interrupts from the shop floor raises the challenge to well control the DRAM testing lots in the shop floor. To help DRAM testing firms face this challenge, a powerful SFCS is definitely required. As RFID technology and agent-based approach appear as two promising technologies, we proposed using the two technologies to enhance the capability of the SFCS used in DRAM testing environments. The contributions of this study are listed as follows.

(1) A hybrid architecture is proposed. In this hybrid architecture agents can be organized into hierarchical and distributed relationships simultaneously. Agents with a hierarchical relationship can communicate to each other directly while agents with distributed relationship can communicate through middleware. This hybrid is expected to alleviate network traffic. In addition, middleware helps integrate SFCS with legacy systems.

(2) A software process based on UML for modeling agent-based systems is provided. As UML is an objected-oriented tool able to link
OOSA, OOSD and OOP together, it can facilitate system modelling and development.

(3). RFID technology and agent-based approach are employed in this study to initiate an advanced architecture for the SFCS used in DRAM test firms. As no such architecture has ever been proposed, it can be referred by DRAM test firms to improve their outdated SFCSs.

Since agents with intelligence can make an agent-based system more adaptable to the dynamic environments, empowering the agents derived in this study with intelligence can be treated as a further research direction.

REFERENCES


