
COMPUTER INTEGRATED MANUFACTURING IN SEMICONDUCTOR INDUSTRY



AUTOMATION, ELECTRONIC WAFER MAPPING,
DEFECT REDUCTION AND EQUIPMENT UTILIZATION
IMPROVEMENT IN PROBE AND FINAL TEST

MANUEL JOSÉ MORENO LIZARANZU

UNIVERSITY OF SEVILLE

DOCTORAL DISSERTATION
SUPERVISED BY DR. FEDERICO CUESTA



NOVEMBER, 2012

First published in November 2012 by
IAR Group
ETS Ingeniería
Camino de los descubrimientos s/n
Sevilla, 41092. SPAIN

Copyright © MMXII Manuel José Moreno Lizaranzu

In keeping with the traditional purpose of furthering science, education and research, it is the policy of the publisher, whenever possible, to permit non-commercial use and redistribution of the information contained in the documents whose copyright they own. You however are *not allowed* to take money for the distribution or use of these results except for a nominal charge for photocopying, sending copies, or whichever means you use redistribute them. The results in this document have been tested carefully, but they are not guaranteed for any particular purpose. The publisher or the holder of the copyright do not offer any warranties or representations, nor do they accept any liabilities with respect to them.

Support: This work has been supported by Freescale Semiconductor Inc.

University of Seville

The committee in charge of evaluating the dissertation presented by Manuel José Moreno Lizaranzu in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Software Engineering, hereby recommends _____ of this dissertation and awards the author the grade _____.

Ramón González Carvajal
Catedrático de Universidad
Univ. de Sevilla

Reinhard Brauningl
Ao. Univ. Professor
Technische Universität Graz

Rafael Corchuelo Gil
Profesor Titular de Universidad
Univ. de Sevilla

Fernando Gómez Bravo
Profesor Titular de Universidad
Univ. de Huelva

José Manuel Martín Ramos
Profesor Contratado Doctor
Univ. de Huelva

To put record where necessary, we sign minutes in _____,
_____.



Computer Integrated Manufacturing in Semiconductor Industry by Nicolás, aged five.

To my family.

Contents

Acknowledgements	xi
Abstract	xiii
Resumen	xv

I Preface and Background Information

1 Introduction	3
1.1 Introduction	4
1.2 Research rationale	4
1.2.1 Hypothesis	4
1.2.2 Thesis	6
1.3 Summary of contributions	7
1.3.1 Architecture and framework with Model Driven Architecture approach	8
1.3.2 Electronic wafer mapping and outlier detection algorithms in probe	12
1.3.3 Equipment utilization tracking and improvement in probe and final test	15
1.4 Structure of this dissertation	16
2 Semiconductor Industry	19
2.1 Introduction	20
2.2 Wafer manufacturing	21
2.3 Crystal growth and wafer slicing	21
2.4 Wafer fabrication process	21

2.4.1	Photolithography	22
2.4.2	Etching	22
2.4.3	Implant	22
2.4.4	Diffusion	22
2.4.5	Visual inspections	23
2.5	Probe	23
2.6	Assembly	25
2.7	Final test	25
2.8	Freescale Semiconductor Inc.	26

II Our Approach

3	Architecture and Framework for Probe and Final Test and Tester Controller Implementation	33
3.1	Introduction	34
3.2	Related work	35
3.3	Architecture and framework	36
3.3.1	Java as development language	37
3.3.2	Eclipse as IDE	37
3.3.3	Client/Server architecture	37
3.3.4	Model View Controller framework	40
3.3.5	Support libraries	40
3.3.6	XML screens definitions	42
3.4	Station Controller 2	43
3.4.1	SC2 architecture, framework and MVC division	44
3.4.2	SC2 XML screens definitions	46
3.5	Model Driven Architecture approach	48
3.6	Failover	53
3.7	Results	53
4	Electronic Wafer Mapping and Zero Defect Algorithms in Unit Probe	57
4.1	Introduction	58
4.2	Related work	60
4.3	Classic outlier detection methods	63

- 4.3.1 SBL, BMY 64
- 4.3.2 GDBC 65
- 4.3.3 SPAT, DPAT, AEC DPAT, NNR 65
- 4.4 New outlier detection methods 69
 - 4.4.1 GDBC SB 69
 - 4.4.2 BBBC 70
 - 4.4.3 DPAT ST 70
 - 4.4.4 R DPAT 71
 - 4.4.5 Multi-site test results 76
 - 4.4.6 Test aggregation 77
- 4.5 EWM functionality 78
 - 4.5.1 Inputs 79
 - 4.5.2 Processes 80
 - 4.5.3 Outputs 81
 - 4.5.4 Interfaces 82
- 4.6 SEMI E142 specification for substrate mapping 82
- 4.7 User interface 84
 - 4.7.1 Hierarchical view 84
 - 4.7.2 Device recipes (inspections) 84
 - 4.7.3 Lot screens 84
 - 4.7.4 Wafer screens 85
 - 4.7.5 Manual visual inspections 86
 - 4.7.6 Reports 86
- 4.8 Results 90
 - 4.8.1 Spatial algorithms comparison 93
 - 4.8.2 Part Average Testing (PAT) algorithms comparison 95
 - 4.8.3 Additional benefits 98

5 Equipment Utilization Tracking and Improvement in Probe and Final Test Areas 101

- 5.1 Introduction 103
- 5.2 Related work 103
- 5.3 Equipment utilization tracking 104
 - 5.3.1 Functionality 105
 - 5.3.2 System context diagram 106
 - 5.3.3 SEMI E10 standard 106

5.3.4	Freescale EPR++ standard for equipment utilization	107
5.4	Real time factory views	111
5.4.1	Utilization view	111
5.4.2	Production view	112
5.4.3	Maintenance view	113
5.4.4	Engineering view	113
5.4.5	Tester transition chart	114
5.4.6	Shop floor 3-dimensional view	116
5.5	Reports and data retention	116
5.5.1	Maintenance report	117
5.5.2	Utilization trend	117
5.5.3	Utilization by tester	117
5.5.4	Detailed losses	119
5.5.5	Average test time and yield by product line	120
5.5.6	Other report types	120
5.5.7	Data summarization and retention	122
5.6	Report scheduling	122
5.7	Results	123

III Final Remarks

6	Conclusions and Future Work	131
----------	--	------------

Bibliography	137
-------------------------------	------------

List of Figures

1.1	Timeline of our contributions	8
1.2	Probe applications	9
1.3	Final test applications	11
2.1	Silicon ingots and wafers	20
2.2	Wafer with integrated circuits	23
2.3	Probers	24
2.4	Prober heads	24
2.5	Packaged integrated circuit	26
2.6	Freescale as global leader in embedded processing solutions	27
2.7	Freescale presence around the world	27
3.1	Eclipse as IDE	38
3.2	Model View Controller division	38
3.3	Client/Server interface	39
3.4	Station Controller 2 MVC division	45
3.5	Client/Server communication	46
3.6	Handler setup screens textual definitions	47
3.7	Handler setups table	48
3.8	Handler setup form	49
3.9	Platform Independent Model	50
3.10	PIM to PSM transformation	50
3.11	Entity template	51
3.12	Database template	51
3.13	PSM and framework	52
3.14	Failover architecture	53
4.1	Freescale's quality initiatives	59
4.2	Statistical outlier detection algorithms	64

4.3	GDBC example	66
4.4	DPAT example	67
4.5	Nearest Neighbour Residual	69
4.6	BBBC example	71
4.7	Robust DPAT algorithm	73
4.8	Lognormal to normal distribution transformation	76
4.9	Exponential to normal distribution transformation	77
4.10	Multi-site example	78
4.11	EWM data flows	79
4.12	EWM deployment status	80
4.13	EWM system context diagram	81
4.14	SEMI E142 wafer map data	83
4.15	Hierarchical view	85
4.16	Device recipe (inspections)	86
4.17	Lot report. Wafer maps view	87
4.18	Lot report. Yield map	88
4.19	Hard bins wafer view	89
4.20	Parametric test results wafer view	90
4.21	STDF report	91
4.22	DPAT yield impact by wafer	91
4.23	DPAT yield impact by test	92
4.24	CQI report	93
4.25	Classic spatial algorithms comparison	94
4.26	New spatial algorithms comparison	95
4.27	Bin to CQI correlation	96
4.28	Part Average Testing algorithms comparison	97
4.29	New Part Average Testing algorithms comparison	98
5.1	TTT architecture	105
5.2	SEMI E10 standard	107
5.3	States and sub-states	108
5.4	EPR++ state transitions	109
5.5	EPR++ example	111
5.6	Utilization view	112
5.7	Production view	113
5.8	Maintenance view	114
5.9	Engineering view	115

5.10	Tester transition chart	115
5.11	3-D shop floor view	116
5.12	Utilization trend report	118
5.13	Utilization by tester report	118
5.14	Detailed losses report	119
5.15	Data summary	122
5.16	Data retention	123
5.17	List of scheduled reports	124
5.18	Report scheduling screen	125
5.19	Reports published on application web page	126
5.20	Utilization improvement. Factory 1	127
5.21	Utilization improvement. Factory 2	128

List of Tables

3.1	Applications statistics before and after using framework (LOC = Lines of Code)	55
3.2	Applications statistics (LOC = Lines of Code)	56
3.3	Average network traffic in bytes before and with framework	56
4.1	Percentage of wafers running each algorithm	92
5.1	EPR++ events	110
5.2	Database size	126

Acknowledgements

Forget injuries, never forget kindnesses.

Confucius

This has been a long journey that took me from one shore of the Atlantic to the other and back. Completing a circle, I have returned to my alma mater for my doctorate. And along this journey, I have met many inspiring individuals that I would like to thank.

Thanks to Dr. Eduardo Fernández Camacho for sparking my interest in Industrial Engineering at the University of Seville.

My gratitude to Dr. Richard Wysk for guiding and advising me for my master's thesis at The Pennsylvania State University and for teaching me the principles of Computer Integrated Manufacturing.

My deepest appreciation to Stephen Chelstrom, Dr. Donald Hartman, and Joju Quejada for their guidance and mentoring at Motorola and Freescale.

I would also like to recognize Thomas Kessler, Scott Hildreth, and Benjamin Yip at Freescale for their insights into outlier detection and manufacturing.

I am deeply thankful to Dr. Federico Cuesta at the University of Seville for mapping the course and guiding me through this dissertation and to Dr. Rafael Corchuelo as well.

Finally, I would like to convey my gratitude to my family for their help, love, and support. You are my northern star.

To all of you, thank you!

Abstract

If you can't explain it to a six year old, you don't understand it yourself.

Albert Einstein

The semiconductor industry poses challenging problems for Computer Integrated Manufacturing (CIM) teams. New requirements are constantly arising, development is addressed by multiple, small groups, and documentation is often poor. Applications must be developed quickly, be robust and reliable, and minimize the amount of resources used (CPU, memory, disk space, and network traffic).

Devices for the automotive and medical industries must perform with a very low defect rate. Electrical tests and visual inspections are applied to detect defective devices and avoid shipping them to customers, yet devices with latent defects are sent originating a Customer Quality Incident (CQI). Outlier detection algorithms are a key component in screening latent defects and decreasing number of CQIs.

Additionally, in semiconductor manufacturing, it is imperative to maximize equipment utilization, especially to survive a down economic cycle. Tuning maintenance, decreasing setup changes and idle time are key factors, although the first step is to measure equipment utilization to determine the dimension of the actions to implement and to be able to quantify the effect of such actions.

An architecture and framework have been created to tackle these problems and have been used to develop three applications that run in production in the probe and final test areas in Freescale Semiconductor Inc. in five factories in Asia, Europe and North America. Freescale is a global leader in embedded processing solutions for the automotive, consumer, industrial and networking markets. Freescale manufactures microprocessors, microcontrollers, sensors, and analog integrated circuits.

The following goals have been achieved:

Development time has been reduced considerably and the three applications are easy to maintain and upgrade. CPU, memory and network traffic have been reduced significantly. Moreover, the framework learning curve is small and developer efficiency has increased at least 25%.

Application 1 controls hundreds of testers in three test floors in Asia and North America. It has increased quality by providing a unified user interface as tester controller and incurred considerable savings by avoiding buying very expensive software while allowing complete customization for manufacturing requirements.

Application 2 systematically combines wafer visual and electric defects and applies outlier detection algorithms to decrease the number of Customer Quality Incidents (CQIs). This application has processed data for over 10 million wafers to date. It implements several outlier detection algorithms published in the technical literature, novel variations of those and new algorithms to increase defect detection performance. Experimental results on 289,080 dice on 495 wafers were analyzed to determine efficiency and effectiveness of these algorithms. The enhancements implemented and the new algorithms have significantly increased performance. Also, as part of this research, the SEMI E142 standard for wafer map data transfer has been developed in collaboration with other semiconductor companies.

Finally, application 3 combines equipment data (probers and testers) and Manufacturing Execution System (MES) events in real time to reflect the state of each tester at any moment. It implements a powerful equipment state model based on the SEMI E10 and SEMI E58 standards. Hundreds of probers and testers send equipment events in real-time to this application. The average number of events received daily is 250K across all the factories. This application allows engineers to quickly spot equipment whose performance is degrading. In addition, the detailed data collected allows performing data mining to identify causes for utilization losses. This has contributed to equipment productive time increase in two final test factories in Freescale Semiconductor Inc. 7.5% and 6.2% respectively during the 10 months following the deployment.

Resumen

*Si no lo puedes explicar a un niño de seis años,
entonces no lo entiendes ni tú mismo.*

Albert Einstein

La industria de semiconductores presenta arduos problemas en el campo de Manufactura Controlada por Ordenador (CIM en inglés). Nuevos requisitos surgen constantemente, el desarrollo de aplicaciones de software se aborda en múltiples grupos y la documentación suele ser deficiente. Las aplicaciones se tienen que desarrollar rápidamente, deben ser robustas y fiables, y reducir al mínimo la cantidad de recursos utilizados (CPU, memoria, espacio en disco y tráfico de red).

Dispositivos para la industria del automóvil y médica deben tener una tasa de defectos muy baja. Pruebas eléctricas e inspecciones visuales se aplican para detectar dispositivos defectuosos y sin embargo, algunos dispositivos fallan después de ser enviados al cliente. Algoritmos de detección de valores atípicos son un componente clave en la detección de defectos latentes y en la disminución del número de dispositivos defectuosos enviados al cliente.

Asimismo, en la fabricación de semiconductores es imperativo optimizar la utilización de la maquinaria de producción, especialmente para sobrevivir un ciclo económico de contracción. El mantenimiento, la disminución de cambios en la configuración y del tiempo en espera son factores clave. Sin embargo, el primer paso consiste en cuantificar la utilización de la maquinaria para determinar la dimensión de las acciones a implementar y para medir el efecto de tales acciones.

En esta tesis, hemos creado una arquitectura y un marco de software para hacer frente a estos problemas y los hemos utilizado para desarrollar tres

aplicaciones que se ejecutan en las áreas de pruebas eléctricas y pruebas finales en Freescale Semiconductor Inc. en cinco fábricas en Asia, Europa y América del Norte. Freescale es un líder mundial en aplicaciones de procesamiento empotrado para los mercados de automoción, consumo, industrial y de redes. Freescale fabrica microprocesadores, micro-controladores, sensores y circuitos digitales integrados.

Se han alcanzado los siguientes objetivos:

El tiempo de desarrollo de aplicaciones se ha reducido considerablemente y las aplicaciones son fáciles de mantener y actualizar. El uso de CPU, memoria y tráfico de la red se han reducido significativamente. La curva de aprendizaje es pequeña y la eficiencia del programador ha aumentado al menos un 25 %.

La aplicación 1 controla cientos de equipos en tres plantas de pruebas finales en Asia y América del Norte. Esta aplicación ha contribuido al aumento de la calidad, proporcionando una interfaz de usuario unificado y ha supuesto un ahorro considerable al evitar la compra de software muy costoso, permitiendo al mismo tiempo una adaptación completa a los requisitos de manufactura.

La aplicación 2 combina sistemáticamente defectos visuales y resultados de pruebas eléctricas y aplica algoritmos de detección de valores atípicos para reducir el número de dispositivos que fallan después de ser enviados al cliente. Esta aplicación ha procesado datos de más de 10 millones de obleas (wafers en inglés) hasta la fecha. Esta aplicación implementa varios algoritmos de detección de valores atípicos publicados en la literatura técnica, variantes novedosas de aquellos y nuevos algoritmos para aumentar el rendimiento de la detección de defectos. Los resultados experimentales de 289,080 circuitos en 495 obleas se han analizado para determinar la eficiencia y la eficacia de estos algoritmos. Las mejoras implementadas en estos algoritmos y los nuevos algoritmos han aumentado considerablemente el rendimiento. Así mismo, como parte de esta tesis, se ha desarrollado el estándar SEMI E142 para la transferencia de mapas de datos de oblea en colaboración con otras compañías de semiconductores.

Finalmente, la aplicación 3 combina datos de los controladores de maquinaria de prueba y eventos del Sistema de Ejecución de Manufactura (MES en inglés) en tiempo real para reflejar el estado de cada máquina en cualquier instante. Se ha implementado un modelo de estado basado en los estándares SEMI E10 y SEMI E58. Cientos de máquinas envían eventos en tiempo

real a esta aplicación. El número medio de eventos recibidos diariamente es 250K. Esta aplicación permite a los ingenieros detectar rápidamente equipos cuyo rendimiento se está degradando. Además, los datos recopilados permiten realizar un análisis para identificar las causas de las pérdidas de utilización. Esto ha contribuido a aumentar el tiempo productivo en dos fábricas en Freescale Semiconductor Inc. un 7,5% y 6,2%, respectivamente, durante los 10 meses posteriores a la implementación.

Part I

Preface and Background Information

Chapter 1

Introduction

The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency.

Bill Gates

Our goal in this dissertation is to present the results achieved automating and improving manufacturing processes in the probe and final test areas in the semiconductor industry, specifically in Freescale Semiconductor Inc.

1.1 Introduction

In this dissertation, we present a software architecture and framework that have been utilized to create three software applications that run in all the probe and final test floors in Freescale Semiconductor Inc. These three applications successfully and efficiently have increased automation and have significantly enhanced manufacturing processes, specifically in the fields of latent defects detection and equipment utilization improvement.

This chapter is organized as follows: section 1.2 present the hypothesis that has motivated this work and the thesis. Section 1.3 summarizes the contributions and lastly, section 1.4 describes the structure of this dissertation.

1.2 Research rationale

In this section we present the hypothesis that has motivated our research work in the context of Computer Integrated Manufacturing in Semiconductor Industry, and state our thesis, which we prove in the rest of the dissertation.

1.2.1 Hypothesis

There is a tremendous need in middle size semiconductor companies to quickly implement customized software applications to automate manufacturing processes [20]. In addition, in the field of wafer testing, it is paramount to detect as many defective devices as possible while maintaining high yield [41]. Moreover, increasing the productivity of the machines used for testing is economically indispensable [19].

Commercial applications are available to cover some of these spaces but they lack some of the requirements. Additionally, the cost associated to the product, the integration in the manufacturing process and the customization make them often economically prohibitive.

Internal development teams are multiple and small groups (normally less than five members) and new requirements are constantly arising. Consequently, quick prototyping is highly desirable as the complexity of these applications makes requirements gathering and validation a difficult task. In addition, software applications must be developed quickly and should be easily maintainable and upgradable, yet robust and efficient.

Traditional lack of design documentation makes enhancing any existing application an enormous challenge. Also, there is the need to train new developers to become productive in a short amount of time.

Moreover, the application landscape is heterogeneous, where legacy applications coexist with newly developed ones and there is an endemic lack of standardization. There are multiple software enablers, operating systems, and database versions rendering systems upgrade and maintenance a difficult task.

In addition, applications should minimize usage of hardware and network resources and run on multiple operating systems due to the need to decrease costs in manufacturing. Applications must be robust and reliable, and minimize the amount of resources used (CPU, memory, disk space, and network traffic). Specially when, very often, the client and server side of the same application are running on opposite sides of the world. Minimizing resource usage increases the competitiveness of the company.

From a quality point of view, devices for the automotive and medical industries must perform with a very low defect rate. Electrical tests and visual inspections are applied to detect potential failures and yet defective devices are shipped to customers originating Customer Quality Incidents (CQIs).

There is a need to systematically combine visual inspections, electrical probe results to more effectively and efficiently filter out defective devices [51, 77]. Statistical outlier detection methods also play a fundamental role in identifying devices that have a high probability to fail [7, 35, 37, 78].

Another challenge is that the processes that produce and consume these defects use different data formats. It became obvious during this research that there is a strong demand for a data format standardization in this area.

Additionally, addressing all these quality problems must be performed in such a way that is seamlessly integrated in the manufacturing process, minimizing disruptions and maximizing automation [77].

From a financial perspective, semiconductor manufacturing follows a cyclical business model in which demand varies considerably. Thus, the need to increase equipment utilization is imperative, especially to survive a down cycle [19]. Probe and final test equipment (testers) represent a vast portion of the cost of final manufacturing operations in the semiconductor industry. Maximizing their utilization is critical to increasing profit margin.

In order to increase utilization, it is essential to be able to measure equipment utilization accurately [13], in a standard manner across manufacturing

floors, and including enough detail, so that losses can be clearly identified, specially the conditions and factors that cause them. Although, companies normally have different equipment tracking applications across factories with overlapping functionality, making factory to factory equipment utilization comparisons not possible.

Equipment losses can be addressed with proactive and reactive maintenance to drive utilization up [12, 13, 29, 31, 68]. Too little maintenance and testers will experience down time more often. More maintenance than necessary will waste valuable engineering resources and equipment time [31]. The reasons for idle time need to be identified and addressed as well.

Although, due to the multitude of factors that contribute to losses, actions implemented to correct them may not have the anticipated effects. A careful analysis of utilization trend is also necessary to quantify the effect of such actions.

1.2.2 Thesis

It is feasible and cost effective to create a domain specific framework and architecture [17] that allows to generate software quickly [22, 36], providing a uniform user interface, encapsulating functionality, and promoting code reuse. This framework should be simple, with a small learning curve to allow developers to become productive in a small amount of time and to increase maintainability. Moreover, a common framework and architecture can be debugged once and reused multiple times, providing a very robust foundation. This framework and architecture would standardize software development, simplifying application maintenance and upgrades.

The Model Driven Architecture (MDA) approach could provide automatic code generation [36, 80] and quick prototyping. This approach would enforce the creation of design documents, increasing the level of documentation which is normally poor and would contribute to the maintainability of the applications developed.

Also, this framework and architecture could be designed is such a way that the amount of resources used (CPU, memory, disk space, and network traffic) would be minimized. By compressing any data sent between client and server and any information stored in the database, disk space and network traffic would be minimized. CPU and memory usage could be decreased by implementing and reusing libraries with common functionality needed in the probe and final test areas. These libraries would be optimized once and reused multiple times.

Creating such framework and reusing it to develop several applications would make them very cost effective, surpassing commercial solutions that have a prohibitive cost and require extensive effort integrating and customizing them to suit the specific needs of the factories.

In the defect detection field, visual inspections [51] and electrical probe results can be systematically combined to screen out defects more effectively and efficiently. Additionally, statistical outlier detection methods could be implemented to decrease the number of devices with latent defects shipped to customers [35, 41, 55, 58, 59, 60, 61, 79]. Some of these methods can be enhanced significantly by analyzing past CQIs to determine which tests that have a higher correlation to CQIs.

At the same time, by leveraging the mentioned framework and architecture, an application can be developed to address these quality problems while being seamlessly integrated in the manufacturing process, minimizing disruptions and maximizing automation.

Additionally, to avoid having to process information in many different data formats, a standard for wafer map data transfer could be created.

To increase equipment utilization, the first step is to measure it in a consistent and standard manner across factories. A common application should be used to collect equipment utilization data to accomplish that goal. Moreover, combining equipment data and Manufacturing Execution System (MES) data is critical to provide a realistic state of the shop floor [28].

Moreover, the data records stored should contain enough detail so that utilization losses can be traced to the ultimate cause. These losses can be addressed with proactive and reactive maintenance to drive utilization up [29, 68].

Finally, information should to be kept for long periods of time (one year or more) so that the effect of corrective measures can be determined by examining the utilization trend over time. To avoid storing huge amounts of data, a summary process could be implemented to keep detailed information for the immediate past weeks and decrease the amount of detail as the data ages.

1.3 Summary of contributions

To prove our thesis, an architecture and framework have been created and leveraged to develop three software applications (SC2, EWM, and TTT) to address the problems listed in section 1.2.

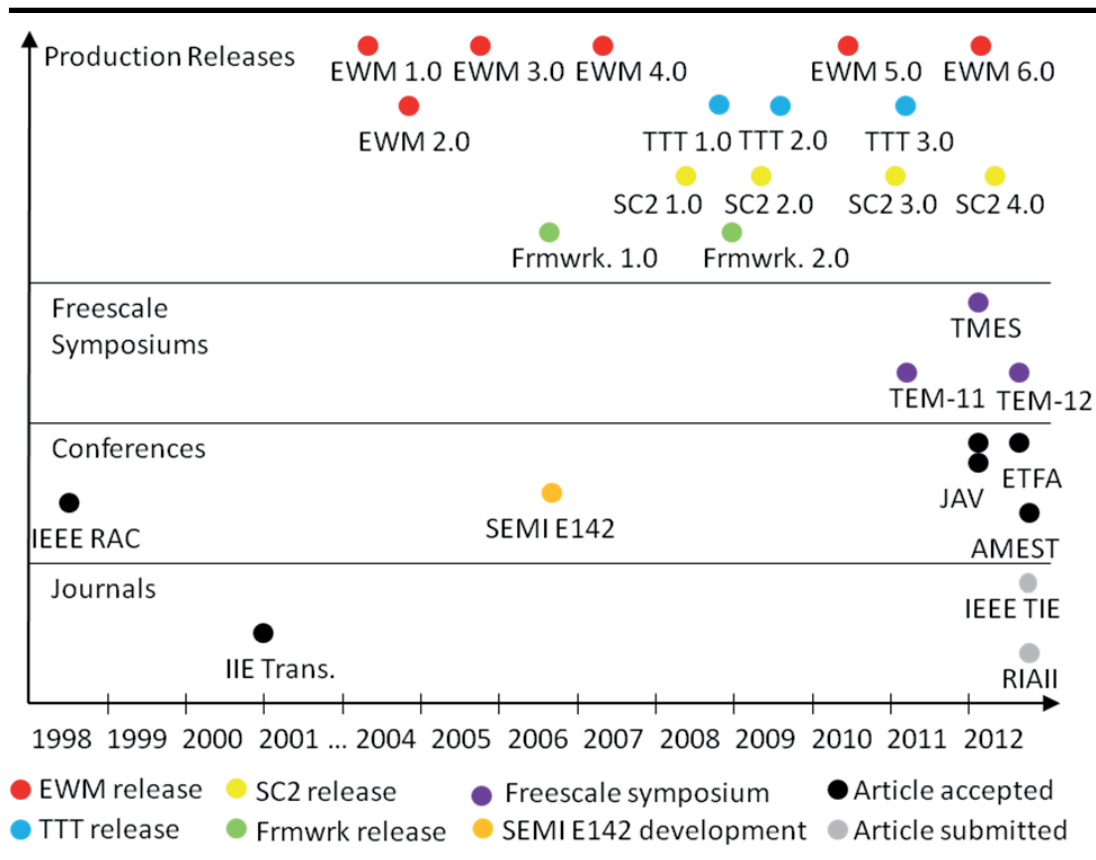


Figure 1.1: *Timeline of our contributions*

The following sections elaborate the results presented in this dissertation. Figure 1.1 shows chronologically the contributions with articles sent to conferences and journals [43, 44, 45, 46, 47, 48, 49, 82], presentations in Freescale internal symposiums (Technical Enrichment Matrix, Test Methodology and Efficiency Symposium), development of the SEMI E142 standard [75], and production releases in Freescale of SC2, EWM, TTT, and the cited framework.

1.3.1 Architecture and framework with Model Driven Architecture approach

This framework and architecture have been used successfully to develop three applications that run in production in all the probe and final test areas in Freescale Semiconductor Inc., in five factories in Asia, Europe and North America.

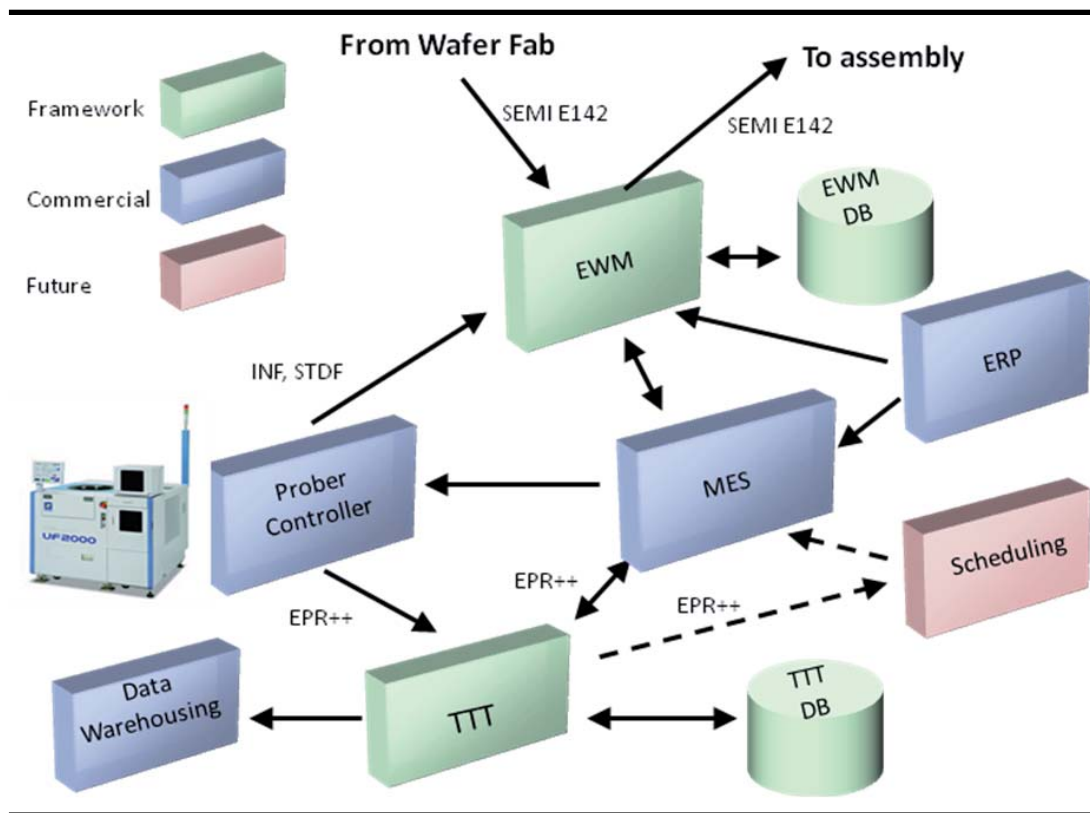


Figure 1.2: *Probe applications*

Figure 1.2 shows the landscape of applications in the probe area. The blocks in blue represent commercial applications, the ones in green have been entirely developed using this framework and architecture and the ones in red will be developed in the near future using the same framework. Similarly, figure 1.3 displays the main applications in the final test area with the systems developed using this framework and architecture highlighted in green.

There are commercial solutions that cover part of this spectrum but fail to consistently combine all defects in a robust, reliable manner and integrate seamlessly in the existing manufacturing process.

- The first application developed is known as Station Controller 2 (SC2). It controls hundreds of testers in three test floors in Asia and North America. It has increased quality by providing a unified user interface as tester controller and incurred considerable savings by avoiding buying very expensive software while allowing complete customization for manufacturing requirements.

- The second application is called Electronic Wafer Mapping (EWM). This application systematically combines wafer visual and electric defects and applies outlier detection algorithms to decrease the number of latent defects sent to customers known as Customer Quality Incidents (CQIs). This application has been successfully deployed to five factories in Asia, Europe, and North America and has processed data for over ten million wafers to date. This application implements several outlier detection algorithms published in the technical literatures, novel enhancements of those and new algorithms to increase defect detection performance which has been proven experimentally with production data.
- The third application is named Tool Time Tracker (TTT). It correlates equipment data (probers and testers) and Manufacturing Execution System (MES) events in real time to reflect the true state of each tester at any moment. This application runs in three probe floors and three final test floors in Freescale, located in Asia, Europe, and North America. Hundreds of probers and testers send equipment events in real-time to this application (the average number of events received daily is 250K). This application allows engineers to quickly identify equipment whose performance is degrading and to perform data mining with the information collected to identify the causes of utilization losses. This has contributed to an increase in equipment productive time in two final test factories in Freescale Semiconductor Inc. of 7.5% and 6.2% respectively, during the 10 months following the deployment.

In addition, the following goals have been achieved:

- Quick prototyping and development. Using a Model Driven Architecture (MDA) approach, a fully functional application is automatically generated. The model is composed of UML [81] class diagrams and XML screens definitions. A simple engine translates this model into source code to generate most of the code for these applications for the probe and final test areas in Freescale Semiconductor Inc. The generated code is multi-layered and complies with the domain specific framework mentioned earlier, thus increasing maintainability. Logic can easily be added to the code generated without having to implement data display and persistence functions. Consequently, development time has been reduced considerably. The three applications described in this dissertation were redeveloped using this architecture and framework. Table 3.1 compares the number of lines of code for the model and view portions

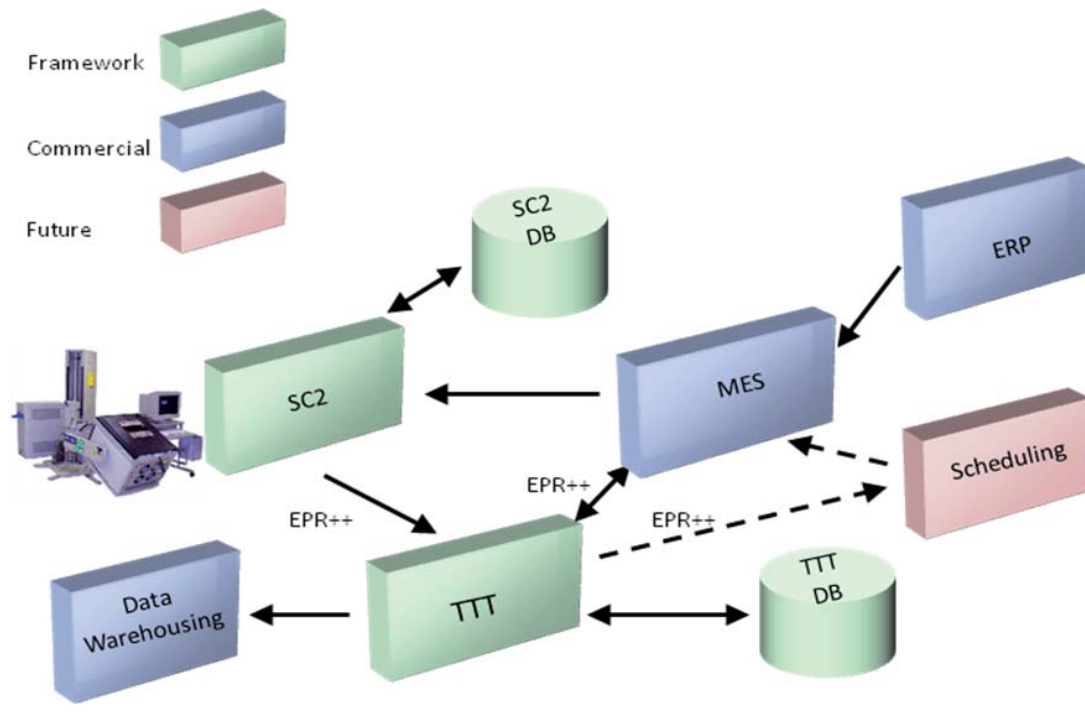


Figure 1.3: Final test applications

of applications 1 and 2 before using this framework and after: a 69% and 40% reduction in lines of code written. Application 3 implemented more requirements than its predecessor so the comparison is not meaningful.

- High maintainability and upgradability. The clean organization that the framework enforces makes maintaining and upgrading these applications a simple task. Figure 1.1 shows all the major releases that have been deployed over several years while maintaining a very low number of defects in the code and an uptime above 99.99%.
- Small learning curve. This framework is very conducive towards the agile development methodology after the first release and very easy to use, with a small learning curve. Junior developers are able to leverage this framework in less than 40 hours of training and developer efficiency is estimated to have increased at least 25%.
- Small footprint (CPU, memory, disk space, database, and network traffic). As detailed in section 3.7, these three applications use little resources. Traffic and transmission times are reduced 93% for medium

size objects and 48% for extra large ones. One of these applications has an average number of client requests of 30,000 per day and the average data transferred between server and client is just 125 Mbytes each day after compression.

- Increased documentation. By using a Model Driven Architecture (MDA) approach to generate the source code, application documentation has improved.
- Cost efficient. All tools and enablers used are open source. The Integrated Development Environment (IDE), all support libraries and the programming language are open source, contributing to the savings for these three applications.
- Leverage existing hardware resources. Since the programming language used is Java, the applications (client and server sides) are platform independent. The code runs on Unix, Windows and Macintosh.

1.3.2 Electronic wafer mapping and outlier detection algorithms in probe

A holistic approach has been developed for defect detection and CQI reduction by combining visual and electrical defects plus statistical outlier detection methods. Visual inspections highlight some latent defects that electrical probe cannot screen out [51]. Additionally, outlier detection methods are proven to further reduce the number of devices with latent defects [35, 41, 55, 58, 59, 60, 61, 79].

A software application called Electronic Wafer Mapping (EWM) has been created to cover this space and is completely and seamlessly integrated in the Freescale probe manufacturing area, running on all the probe floors in five factories in Asia, Europe, and North America. In addition, EWM can be easily set up and tailored for the needs of each device.

Several algorithms described in the technical literature have been implemented: SBL, BMY, GDBC, SPAT, DPAT, AEC DPAT and NNR. Novel modifications to these outlier detection algorithms and new methods have been developed and implemented which significantly have increased the efficiency and effectiveness of these methods, decreasing the number of CQIs.

The percentage of wafers running these algorithms is very high, significantly contributing to device quality: Part Average Testing (PAT) algorithms

are applied to 48% of the wafers, Spatial algorithms (GDBC) to 71%, SBL and BMY to 66%, and visual inspections to 20% (17% automated, 3% manual). Not all devices need to run these outlier detection algorithms. Devices for non-critical applications do not require them. Most devices run SBL and GDBC, and the ones for critical applications run DPAT and have visual inspections.

Algorithm processing time is less than 20 seconds per wafer for the most complex algorithm and about 2 seconds on average. These short processing times minimize CPU usage, keeping hardware cost low.

A novel variation of GDBC and a new spatial algorithm have been created which have increased the performance of this algorithm significantly:

- GDBC with specific bins(GDBC SB). This modification only considers bad dice the ones that have failed due to specific causes (bins). This list of causes is determined by analyzing past CQIs and calculating the probability that a given defect cause (bin number) has an adjacent CQI. All the known CQIs should be included in this analysis. In the absence of CQIs, the classic GDBC algorithm is used.
- Bad Bin in a Bad Cluster (BBBC). This algorithm identifies clusters of bad dice and then marks all good dice surrounding the cluster as bad. The rationale behind this method is the fact that certain clusters of defects have a high correlation to latent defects in neighboring dice.

Several modifications of the DPAT algorithm have been developed as well:

- DPAT with specific tests (DPAT ST). Traditionally, most tests performed are included in DPAT. Although, not all tests have a strong correlation to latent defects. Consequently, using all tests indiscriminately renders difficult to identify future CQIs. One of the enhancements designed in this thesis only includes tests that have a correlation with past CQIs which increases DPAT's performance significantly. All known CQIs should be considered. In the absence of CQIs, a list of tests defined by the division is used.
- Robust DPAT (R DPAT). This enhancement uses robust statistics to better estimate sample data dispersion. It also transforms non-normal data to normal to factor in skewness. These two procedures increase the method's performance.

- Multi-site DPAT. Most often, test results are collected by several probe heads and, often, these probe heads are not calibrated correctly and record results that are shifted with respect to one another. Outlier detection is more effective and efficient if test results are divided by probe site.

An analysis has been conducted on 289,080 dice on 495 wafers with 205,671 good dice, 83,409 defective dice, and 26 CQIs to determine the efficiency and effectiveness of these algorithms which reveals the significant increase in performance of the new modifications and methods as detailed in section 4.8. Such studies are extremely rare in the technical literature.

Additionally, the SEMI E142 standard [75] for wafer map data transfer was elaborated in collaboration with other semiconductor companies and leveraged in this research which has significantly decreased application development time. Freescale was one of the first companies to implement this standard and many other companies are using it today: Infineon, NXP (former Phillips), STMicroelectronics, TSMC and many more.

Besides the quality benefits of outlier detection algorithms, the following benefits have been realized as well:

- Algorithms can be easily set up in EWM and tailored to the needs of each device. According to the device's application, outlier detection will be more or less aggressive.
- One of the traditional outlier detection algorithms is Static Part Average Testing (SPAT). This algorithm is normally executed inside the test program. The performance of this algorithm is poor, but it is run to avoid shipping parts outside the specification limits. The limits for SPAT need to be updated every six months to adjust to changing conditions and the test program needs to be re-qualified which is time consuming. This algorithm can be executed in EWM, thus the test program revalidation step is avoided. In addition, SPAT limits in EWM can be easily relaxed to allow more efficient algorithms (DPAT, NNR) to be the front line of outlier detection.
- There is a complete control over algorithms and roadmap. New algorithms can be easily and quickly implemented, developing an advantage over competitors with a similar product.

- Defective devices are normally marked with ink after the fabrication process to indicate they must be discarded. An alternative is called inkless assembly in which an electronic file contains the location of defective devices on the wafer. EWM supports inkless assembly by implementing the SEMI E142 format [75] for wafer map data transfer. Savings are incurred by not having to ink wafers to mark faulty dice before sending them to assembly.
- Deployment of this application to all probe floors in Freescale was carried out in conjunction with process standardization across all probe floors, decreasing variation (and development time) and, consequently, increasing quality. Requirements from all sites were collected throughout several years and prioritized to implement first the ones that had the greatest impact.
- EWM has also increased automation by integrating into the unit probe operation, interfacing with the Manufacturing Execution System (MES) for lot split, merge, hold, release, and automatic die count updates and with the Enterprise Resource Planning (ERP) system for automatic wafer map transfer to assembly. By increasing automation, labor costs have been reduced.
- Finally, hardware resources used by EWM are optimized. Database footprint for wafer maps and recipes stored is minimized. This is achieved by compressing wafer map information before storing it in the database. This compression schema is also used when transferring information for display to the user interface, decreasing network traffic.

1.3.3 Equipment utilization tracking and improvement in probe and final test

A software application called Tool Time Tracker (TTT) has been developed entirely to address the equipment utilization improvement points listed in section 1.2 and has been successfully deployed in Freescale to three probe floors and three final test floors in Asia and North America. Hundreds of probers and testers send equipment events in real time to TTT. On average, the number of events received daily is 250K across all the floors.

TTT correlates equipment utilization data and Manufacturing Execution Systems (MES) events in real time to reflect the true state of each tester at any moment. It has successfully become a single source for equipment performance data that implements a simple, yet powerful, equipment

state model based on the SEMI E10 standard [74] while recording detailed information about utilization losses according to the SEMI E58 standard [76].

A key contribution introduced in this dissertation is the concept of context which is an extension of the equipment state and includes sub-state, device, lot, operator, test program, probe card ID, load-board, etc. This allows intricate data mining that makes possible to discover specific factor(s) contributing to equipment utilization losses. Any numerical variable (devices processed, yield, test time, alarms, etc.) can be graphed versus any context variable and analyzed that way.

Additionally, data records are summarized by merging the ones with the same context. By using this schema, information can be kept for months while minimizing the amount to database space needed. This functionality sets TTT apart from other commercially available software.

TTT allows accurate factory to factory comparison of tester performance and losses. Moreover, a real time status display of testers and real time reports are available in TTT, allowing maintenance engineers to react immediately to deteriorating conditions. Historical performance reports are also available to analyze the impact of corrective measures implemented.

Additionally, TTT is aware of equipment and resources usage, notifying engineers about preventive maintenance when usage reaches the maximum allowed.

Finally, the equipment data collected is also used for accurate capacity planning.

As a result, productive time has increased partly due to the leverage of TTT. During the 10 months following TTT deployment, equipment productive time has increased by 7.5% and 6.2% respectively in two final test Freescale factories as detailed in section 5.7

1.4 Structure of this dissertation

This dissertation is organized as follows:

Chapter 1 is the introduction to this dissertation.

Chapter 2 provides an insight into semiconductor manufacturing and Freescale Semiconductor Inc.

Chapter 3 describes a framework, architecture and a Model Driven Architecture (MDA) approach for code generation used to implement three applications with very demanding requirements. Additionally, an application to control testers named SC2 is presented as an example of this approach.

Chapter 4 introduces a holistic approach to defect reduction by combining visual and electrical defects plus statistical outlier detection methods. Novel modifications to existing algorithms and new ones are introduced which increase performance. Additionally, an analysis has been conducted on production data to prove it. Finally, a software application named EWM that was developed at Freescale Semiconductor Inc. to materialize these concepts is presented in detail.

Chapter 5 describes a global application (TTT) developed at Freescale to track and increase equipment utilization.

Chapter 6 concludes this dissertation highlighting the main accomplishments and future work.

Chapter 2

Semiconductor Industry

Technology has advanced more in the last thirty years than in the previous two thousand. The exponential increase in advancement will only continue

Niels Bohr

S*emiconductor manufacturing is divided into four steps: wafer fabrication, probe, assembly, and final test. The first step is wafer fabrication, in which integrated circuits (ICs) are fabricated layer by layer on silicon wafers. The next step is probe where electrical tests are performed on each IC on the wafer to determine whether or not they are defective. Assembly follows next, in which non-defective ICs are enclosed into a package. And finally, in final test, the packaged ICs go through an additional series of tests to filter any possible defects added during the packaging operation.*

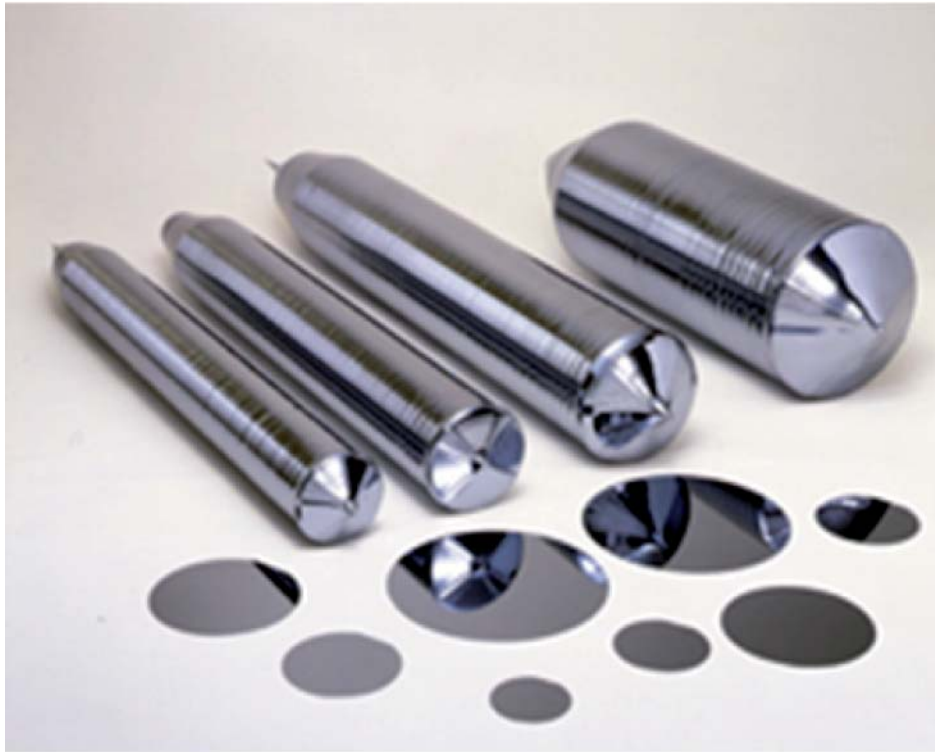


Figure 2.1: *Silicon ingots and wafers*

2.1 Introduction

Semiconductor manufacturing transforms silicon wafers into integrated circuits. Starting with wafers of pure, crystallized silicon (figure 2.1), the processes described here build up a succession of layers of materials and geometries to produce thousands of electronic devices at microscopic sizes, which together function as integrated circuits (ICs). These processes require incredible precision and control.

Semiconductor manufacturing is divided into four major steps: wafer fabrication, probe, assembly, and final test. This chapter is organized as follows: Section 2.2 describes the wafer fabrication process. Sections 2.3, 2.4, and 2.5 address probe, assembly, and final test respectively. Finally, section 2.6 provides information about Freescale Semiconductor Inc.

2.2 Wafer manufacturing

Integrated Circuit (IC) designs are developed with Computer-Aided Design (CAD) systems. Designs are tested by simulation, and perfected on computer systems before they are actually built.

ICs contain billions of components: transistors, resistors, and capacitors which are built on multiple layers, one on top of another. A glass photo mask is developed for each layer of the circuit which will be used during photolithography (detailed later).

Silicon is the basic material of ICs. Turning silicon into ICs requires numerous steps and a lot of precision. The first step is to create the silicon wafers themselves. Then multiple layers are built on the wafers to create the ICs, known as wafer fabrication process [8]. Finally, a visual inspection is performed to detect particle contamination.

2.3 Crystal growth and wafer slicing

The first step is the formation of a large silicon crystal (see [8] for in-depth description). The silicon starts as granular powder that is melted. Then, a crystallized seed is dipped into molten silicon and then removed slowly as it rotates (Czochralski method). The result is a pure silicon cylinder called ingot. The diameter is either six (150 mm) or eight inches (200 mm).

Then, the silicon ingot is sliced into very thin wafers, which is done with a diamond saw. Each wafer is given a flat edge that will be used to orient the wafer correctly during later procedures. Finally, the wafers are polished until they are smooth and have the right thickness.

2.4 Wafer fabrication process

Semiconductor devices are fabricated in clean rooms to avoid particle contamination that will damage the devices. Class 1 clean rooms are typical environments which restrict to no more than 1 particle of dust in a cubic foot of air. The air inside a clean room is filtered continuously, and operators wear special gowns and masks to keep the air particle-free.

Each single wafer will go through multiple steps to achieve the complex layers of conductor, semiconductor, and insulating material needed.

These steps are repeated dozens of times (once for each mask required by the circuit) to create the various layers necessary to build the circuitry.

The first layers deposited on the wafer create all the components and the last layers connect these components. The following sections describe these steps.

2.4.1 Photolithography

In this step, wafers are coated with photoresist which is a light-sensitive substance. Then, a mask is used to expose portions of the wafer. This mask is carefully aligned and ultraviolet light is applied. This light passes through the transparent sections of the mask and chemically modifies the photoresist on those areas. Lastly, a developer solution is applied to the entire wafer to remove the exposed photoresist. The non-exposed photoresist is left on the wafer which will not react to etchants used in successive steps.

2.4.2 Etching

The etching process follows photolithography to remove unwanted material from the wafer. This process removes oxide not protected by photoresist. This leaves a pattern on the wafer in the exact design of the mask. There are two main methods of etching, wet etching (using acids) and dry etching (using gas).

2.4.3 Implant

The next step in the process consists on implanting ions (known as dopants) onto areas of the wafer that are not covered by the photoresist. These dopants are implanted just below the surface of the top layer and will modify the electrical characteristics of these selected areas which encourage or discourage the flow of electrical current. Typical dopants are: boron, arsenic, and phosphorous. After this step, wafers are heated in a process called annealing to reduce any possible damage incurred by implant.

2.4.4 Diffusion

Diffusion is performed in furnaces where an oxidation process occurs. Areas of the wafer not covered by the photoresist will be oxidized.

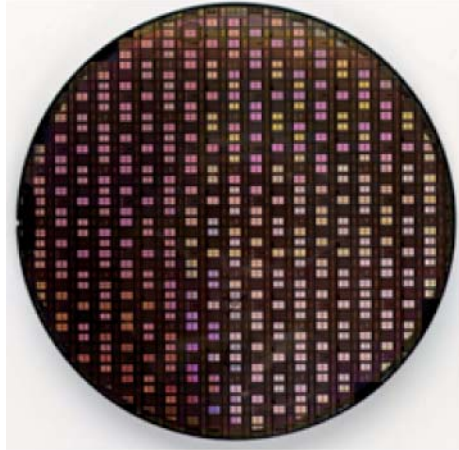


Figure 2.2: *Wafer with integrated circuits*

2.4.5 Visual inspections

The final step in wafer manufacturing is a visual inspection where wafers are placed under a microscope and automatically scanned for particle contamination and structural defects.

2.5 Probe

At this point, all individual integrated circuits (also known as dice) are still on the wafer (figure 2.2). During this step, these dice are tested for functional defects by applying special test patterns to them and reading the results.

Depending on the application for the device, testing is more or less aggressive. Devices for critical applications in the automotive and medical industries must comply with extremely low defect rates. Devices for those markets are tested more rigorously to ensure a small number of latent defects.

These electrical tests are conveyed on a piece of equipment called prober (figure 2.3). A set of microscopic contacts or probes called probe card is positioned while the wafer is moved to make electrical contact with the probe heads (figure 2.4). The software that determines what tests to apply and records the results is referred to as test program.



Figure 2.3: Probers

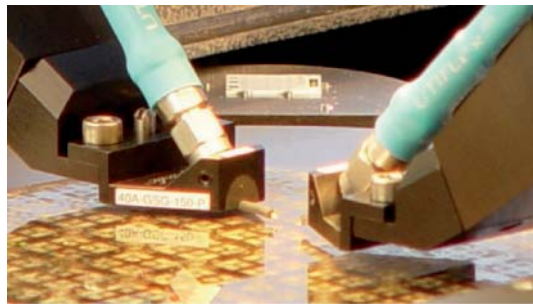


Figure 2.4: Prober heads

There are two steps in probe: class probe and unit probe. In class probe, an entire reticle of dice is tested. In unit probe, individual dice are tested instead.

When a die (or array of dice) have been electrically tested, the prober moves to the next die (or array) and the next test is performed. The wafer prober is usually responsible for loading and unloading the wafers from their carrier (or cassette) and is equipped with automatic pattern recognition optics capable of aligning the wafer with sufficient accuracy to ensure correct registration between the contact pads on the wafer and the tips of the probes.

The results of these tests are measurements like voltage, current, time delay, etc. and are real numbers. These results are interpreted by the test program and if any of the measurements are outside the specification lim-

its, the die is considered defective. A bin number (integer) is assigned to the die to indicate if it passed (1) or if it failed (number between 2 and 255 where each number indicates a different failure cause).

The proportion of dice on the wafer that have passed all the tests is referred to as wafer yield.

In addition to these electrical tests, standard outlier detection algorithms are applied to devices for critical applications to further screen defective devices as requested by clients in those demanding markets.

Finally, non passing dice will be typically marked with a small dot of ink in the middle of the die (referred to as inking) before the next manufacturing step.

Alternatively, the information of passing/non-passing dice is stored in an electronic file, named wafer map. In this case, the process is referred to as inkless assembly. This map categorizes the passing and non-passing dice by making use of bins (1 for passing, 2 through 255 for failing). This wafer map is then sent to the assembly process which only picks up the passing dice by selecting the bin number for good dice.

2.6 Assembly

At this manufacturing step, a diamond saw cuts the wafer into individual dice. The ones marked as defective during the probe step are discarded.

Then, the die bonding process takes place which connects the pads on the chip to the frames with gold wires to create the electrical path between the chip and the package legs.

Finally, dice are encapsulated into plastic packages. Molten plastic is pressed around each die to form its individual package (figure 2.5).

2.7 Final test

Finally, additional tests are performed which push chips to their extreme limits of performance to ensure a high quality, reliable die and to assist engineering with product and process improvements.

During this final step in the manufacturing process, each chip is tested at various conditions to make sure the chip is still performing according to specifications. These conditions include cold, room and hot temperatures and for



Figure 2.5: *Packaged integrated circuit*

some devices a rigorous test called burn-in where the chips are placed in ovens at high temperature while electrical tests are applied to ensure reliability. Tests are performed by equipment known as testers. A device called handler picks up devices to be tested, feeds them to the tester and discards the ones deemed as defective.

Finally, chips are inspected, sealed, labeled, and shipped to customers.

2.8 Freescale Semiconductor Inc.

Freescale Semiconductor Inc. is a global leader in embedded processing solutions for the automotive, consumer, industrial and networking markets [18]. Freescale manufactures microprocessors, microcontrollers, sensors, and analog integrated circuits (figure 2.6).

Freescale was one of the first semiconductor companies in the world, starting as a division of Motorola in Phoenix, Arizona in 1949 [50]. Later, it became the Semiconductor Products Sector of Motorola. And finally, it span off as an independent company in 2004.

Freescale has a headcount of over 18,000 employees and revenues of \$4.57 billion USD in 2011. The headquarters are in Austin, Texas [18]. Design, research and development, manufacturing and sales operations are present in more than 20 countries (figure 2.7).

The company has a long history of technology innovation and has developed some of the most world changing inventions while embracing the sustainable use of the earth's resources, and executing with impeccable ethics.

Automotive

- No. 1 in auto independent MEMS units¹
- No. 1 in auto airbag accelerometers¹
- No. 1 in North American auto semiconductors²
- No. 2 in auto MCUs²
- No. 2 in auto infotainment applications processors²

Consumer

- No. 1 in e-reader processors³
- No. 2 in applications processors for tablets and e-readers³

Industrial

- No. 2 in MCUs¹
- No. 2 in China MCUs⁴
- No. 3 in industrial microcomponents¹

Networking

- No. 1 in communications processors⁵
 - No. 1 in embedded MPUs (ex data proc)⁵
 - No. 1 in high power RF transistors for wireless infrastructure⁶
 - No. 2 in programmable DSPs⁷
-

Figure 2.6: Freescale as global leader in embedded processing solutions



Figure 2.7: Freescale presence around the world

Additionally, Freescale is fully committed to the highest levels of product quality, manufacturing excellence and flawless new products.

There are about 17 billion semiconductor chips in use around the world manufactured by this company. In addition, the company offers software and development tools to support product development [50].

The following is a list of key milestones in Motorola and Freescale [18]:

- 1949:** the Motorola semiconductor development group was created and the first semiconductor plant was built in Arizona.
- 1955:** Motorola commercialized the world first high-power transistor for car radios.
- 1969:** a Motorola radio transponder aboard the Apollo 11 sent the first words from the moon to Earth in July 1969.
- 1974:** Motorola manufactured its first microprocessor, the MC6800 8-bit model used in automotive and computing applications
- 1979:** Motorola and its automotive customers developed the world's first microprocessor-based engine control.
- 1984:** Motorola introduced the MC68020, the world's first true 32-bit micro-processor. It contained 200,000 transistors and powered devices such as Apple Macintosh computers, and Sun 3 workstations.
- 1989:** Motorola pioneered the communications processor market by delivering the industry's first multi-protocol microprocessor — the precursor to the market-leading PowerQUICC line.
- 1990:** General Motors chose Freescale's SMARTMOS technology to create the first automotive smart power IC for a brake antilock system.
- 1991:** Motorola demonstrated the world's first global system for mobile communications (GSM).
- 1995:** the PowerQUICC communications processor was introduced as a revolutionary device for networking and telecommunications.
- 1996:** Motorola was one of few suppliers to deliver the first micro-electro mechanical systems (MEMS) inertial sensors for automotive airbags.

- 2002:** Motorola microprocessors were designed into the world's first tubing-free wireless insulin pump for diabetes patients.
- 2003:** the micro-electro-mechanical sensor (MEMS) is one of the first pressure sensors to address the U.S. TREAD act requirement for tire pressure monitoring.
- 2005:** the first 90 nm multi-core digital signal processor (DSP) for networking is released.
- 2008:** Freescale debuts the world's most powerful automotive microcontroller for efficient engine design based on Power Architecture technology, designed to deliver 10 times the performance of typical engine controllers time while reducing system cost by nearly 30 percent.
- 2008:** Freescale's motion-sensing accelerometer enables the interactivity of Guitar Hero series of video games.
- 2008:** Freescale introduces the QorIQ series of communications platforms — the evolution of the industry-leading PowerQUICC line.
- 2010:** Freescale is a leading supplier of semiconductors for the global automotive industry and it is estimated that 7 out of 10 mobile phone calls are carried by Freescale silicon.
- 2010:** Freescale launches Xtrinsic, the first smart sensor in the market for smart phones and medical devices.
- 2010:** Freescale is #1 in powering the world's e-readers, including market leaders Amazon Kindle™ and Sony eReader™.
- 2011:** Freescale introduces industry's first multimode wireless base station processor family that scales from small to large cells.

Part II

Our Approach

Chapter 3

Architecture and Framework for Probe and Final Test and Tester Controller Implementation

Simplicity is the ultimate sophistication.

Leonardo da Vinci

Computer Integrated Manufacturing (CIM) in the semiconductor industry faces a plethora of challenges. Companies must adapt continuously to the changing economic environment while providing new products and solutions, consequently new requirements are constantly being added to CIM applications. Development is normally tackled by multiple, small groups without sharing a common architecture.

Software documentation is frequently poor. Prototyping, development as well as upgrading existing applications must be done in a short time. Moreover, applications must be robust and efficient to minimize CPU, memory, disk space, and network costs, increasing the competitiveness of the company.

To address all these problems, a framework and architecture have been created to quickly develop shop floor applications for the semiconductor industry. A Model Driven Architecture (MDA) is also utilized to automatically generate portions of the code.

Finally, an application to control testers named Station Controller 2 (SC2) is presented as an example of this approach.

3.1 Introduction

The semiconductor industry faces specific demands due to the cyclical nature of the business. It is also critical to adapt to changing markets and customers' needs. In addition to the heterogeneous environment where legacy applications coexist with newly developed ones and lack of standardization, new requirements are constantly surfacing.

Software applications must be developed quickly and should be easily maintainable and upgradable, yet robust and efficient. Traditional lack of design documentation makes enhancing any existing application an enormous challenge.

Another problem found in computer integrated manufacturing in the semiconductor industry is the need to train new developers to become productive in a short amount of time. Development teams are normally less than five members.

Quick prototyping is also highly desirable as the complexity of these applications makes requirements gathering and validation a difficult task. In addition, applications should minimize usage of hardware and network resources and run on multiple operating systems due to the need to decrease costs in manufacturing.

To address all these challenges, a framework and architecture have been created to quickly develop shop floor applications for the semiconductor industry.

This architecture and framework have successfully been used to develop three software applications in the probe and final test areas in Freescale Semiconductor Inc. These applications operate in five factories in Asia, Europe and North America. Application 1 controls hundreds of testers. Application 2 has processed data for millions of wafers to date and application 3 receives about 250K messages daily from application 1 and prober controllers. Additionally, these applications have an uptime of more than 99.99%.

Application 1 is known as Station Controller 2 (SC2). SC2 is an advanced operator interface that controls both, the tester and handler hardware in final test and it is presented in this chapter as an example of the utilization of this framework, architecture and MDA approach.

This chapter is organized as follows: Section 3.2 describes the related work. Section 3.3 introduces the architecture and framework. Section 3.4

presents one of the applications developed with this approach named Station Controller 2 (SC2) and how the architecture and framework have been materialized. In section 3.5, the Model Driven Architecture paradigm that has been used is presented. Section 3.6 describes a failover schema for added robustness and finally, section 3.7 discusses the main results achieved.

3.2 Related work

Most of the problems mentioned earlier are also found in CIM in other industries besides semiconductors. In the last decade multiple approaches have been published.

Years ago, we proposed an engine called RapidCIM [43, 82] to generate software automatically to control and coordinate a manufacturing system, which substantially reduces the cost of developing and integrating such systems. The benefit of keeping definition and implementation separate is evident [34]. By defining a model, the application functionality can be reviewed and perfected before any code is written. This also increases the quality of the generated code. In addition, it is conducive towards automatic code generation, decreasing development time.

More recently, Marcos et al. [36] proposed a model driven approach for designing industrial control systems based on different views of the system. To do so, they define a description language with information about the different domain views and relationships. Moreover, Estevez et al. [15] suggest the usage of a markup language (XML) for factory automation at the different strata of the process which allows specifying applications from different views. Whereas markup languages lack the visual advantages of UML [81] as modeling tool.

To bypass that disadvantage, Jim [27] proposes to generate source code from structural and behavioral models, focusing on the transformation from the Platform Specific Model (PSM) to source code. Although, it is difficult to capture semantics with UML [81] and it is impractical to fully model the behavior of a system with UML [81].

As the size and complexity of applications in industry has augmented over the last years, a more modular approach is needed. To address this point, Hargassner et al. [22] describe how they have developed a middleware based on OSGi for connecting real-time control software with process visualization applications. The authors propose to divide industrial applications software in three layers. The bottom layer encapsulates the real-time control software,

the middle layer provides interfaces to databases and file systems and the top layer is the process visualization which feeds data to remote clients. Nevertheless, this solution is very generic and creates a huge executable (44 Mb). A simplified and specialized framework for the probe and final test areas would decrease executable size, reducing memory and disk usage. Moreover, this simplification would make it easier to learn and leverage, with efficient CPU usage and minimizing network traffic as key factors in its design.

Finally, with the goal of reducing maintainability, Frantz et al. [17] propose a domain specific language and a set of tools to develop solutions within the context of Model Driven Engineering. Nonetheless, this framework is geared towards enterprise integration using a messaging system that does not fit the specific requirements for the probe and final test areas in Freescale.

In summary, a new, specific framework is required in order to fulfill the strict demands of the semiconductor manufacturing.

In this chapter, we present a domain specific framework and architecture for the probe and final test areas in the semiconductor industry and we propose combining UML [81] diagrams with XML screens definitions to automatically generate most of the source code for applications that are fully compliant with that framework and architecture.

3.3 Architecture and framework

An architecture and framework have been specifically created for probe and final test areas in the semiconductor industry. The goals are:

- Quick prototyping and development.
- High maintainability and upgradability.
- Applications must be robust and efficient: small footprint (CPU, memory, disk space, database, and network traffic).
- Simple, with small learning curve.
- Support complex and interactive Graphical User Interface (GUI).
- Low cost development environment and enablers.
- Platform independent to allow reusing existing hardware.
- Promote application documentation, especially design documents.

3.3.1 Java as development language

Java is one of the most popular languages for enterprise application development [65]. There is a vibrant community of developers behind this language, it is open source and it runs on most of the widely used operating systems: Windows, Macintosh and most Unix systems which fulfills the platform independent requirement for this architecture and framework.

Java is simple yet powerful. The object oriented features make Java code easy to maintain and upgrade contributing to the framework high maintainability objective.

Additionally, Java combines the best features of interpreted and compiled languages. It can be easily debugged and profiled while the latest Java compilers generate machine code that is highly efficient.

Moreover, Java allows building very complex and interactive user interfaces, fulfilling that demand in the probe and final test areas where huge amounts of data need to be displayed in a fast and intuitive way. Solutions based on client HTML rendering do not meet this requirement.

3.3.2 Eclipse as IDE

Eclipse [14] was chosen as the Integrated Development Environment (IDE) for all the applications described in this dissertation. Eclipse (Figure 3.1) has been used for modeling, designing, coding, testing, profiling and team collaboration (repository).

The enormous support behind Eclipse, its Model Development Tools (MDT) module used for the Model Driven Architecture (MDA) approach, its wide use in the Java community plus the open source concept and the long list of available plug-ins make it a clear choice, fulfilling the requirement of having a low cost development environment.

3.3.3 Client/Server architecture

Distribution of information is paramount in a global company. Applications must be readily accessible anywhere in the world while minimizing downtime. The client/server architecture is the best suited to accomplish this goal.

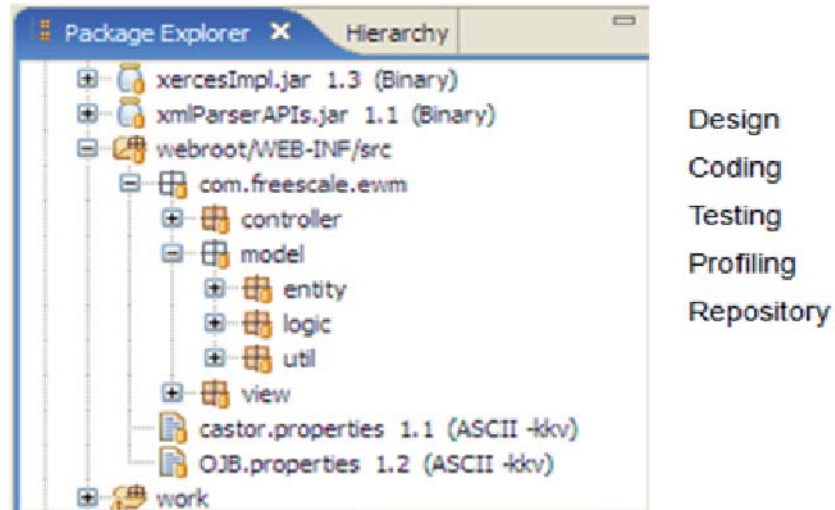


Figure 3.1: Eclipse as IDE

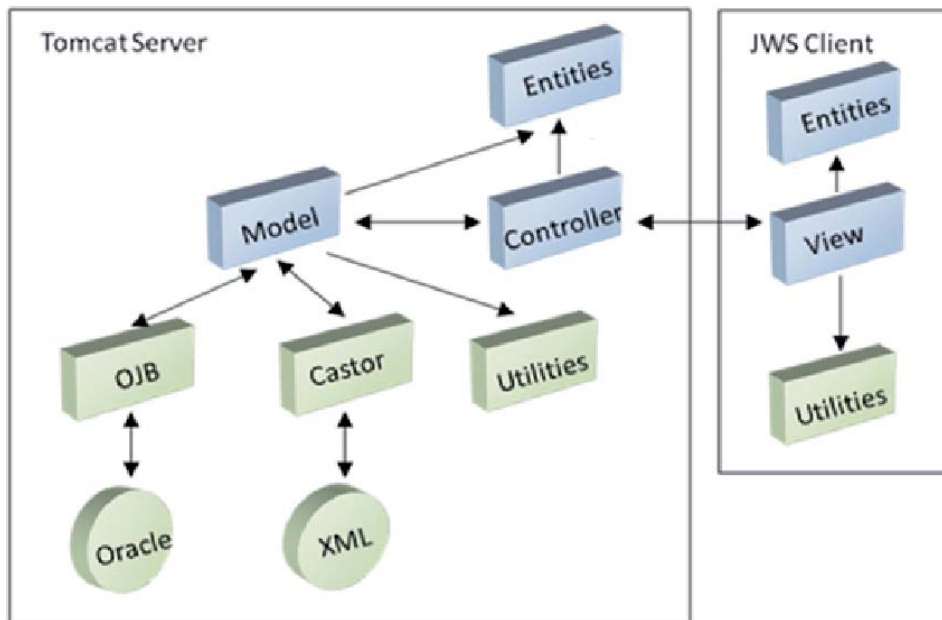


Figure 3.2: Model View Controller division

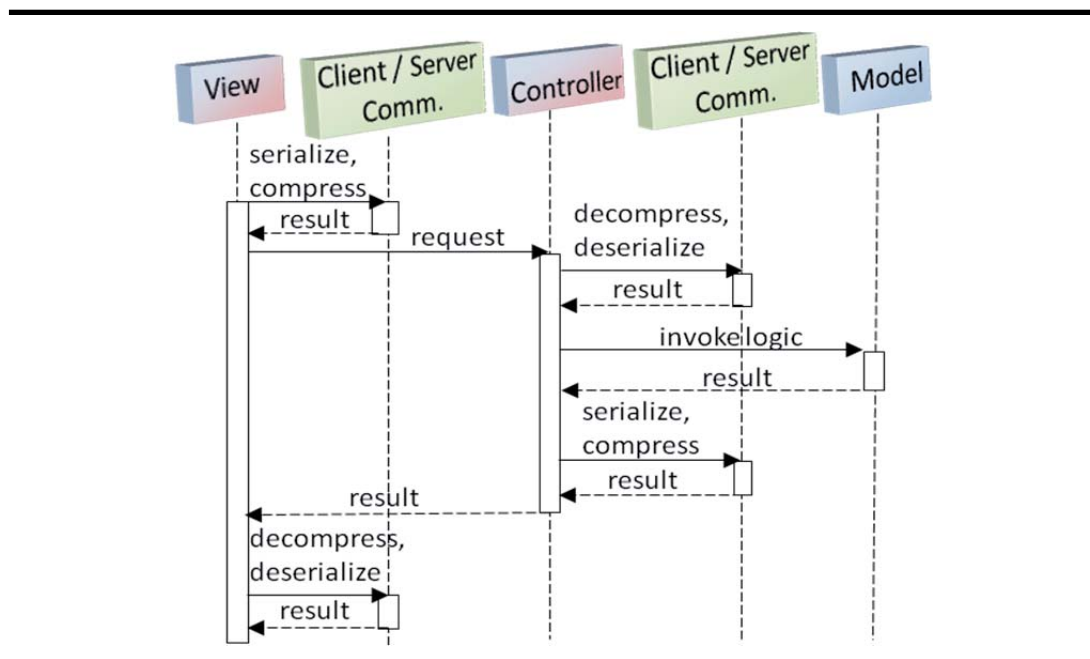


Figure 3.3: *Client/Server interface*

In this framework, both, client and server sides are developed in Java. The server runs in Apache Tomcat [5]. The client side is executed using Java Web Start (JWS) [64] as shown in figure 3.2. Consequently, client and server sides can run on any operating system that supports Java.

This architecture is POJO centered. POJO stands for Plain Old Java Object. Each entity is implemented as a POJO and only contains the attributes related to the entity and no logic. All the logic is kept separate from the entities definition.

The communication between the client and server is performed transferring these POJOs. The client sends a request to the server over a TCP/IP socket by serializing and compressing a POJO with the data needed to perform the request. The server decompresses and deserializes the object and executes the desired operation. The result of the operation is another POJO that is in turn serialized and compressed and sent back to the client (figure 3.3).

This schema provides a simple and very efficient interface between server and client. Entities are seamlessly transferred between client and server with a simple method invocation provided by the framework support libraries.

Additionally, the compression approach minimizes the network traffic which is a critical factor since often the client and the server are running on opposite sides of the world and the amount of data to be transferred is considerable and should be available at the client side in less than 10 seconds, preferably in less than 5 seconds. Thus, the requirement to minimize network traffic is fulfilled.

Moreover, the server side of the application runs on high availability servers that are constantly monitored. The client portion runs locally on the user's computer, mostly formatting and displaying the information requested while decentralizing some of the processing, contributing to the reliability requirement.

3.3.4 Model View Controller framework

The framework follows the Model View Controller (MVC) division thus increasing maintainability (figures 3.2). The model and controller portions run on the server side on Apache Tomcat [5].

The model portion implements the application logic which executes requests from the client side retrieving, storing, deleting and updating entities in the database. It also transforms the entities according to those requests and runs background processes as well. Entities are defined in the model portion but kept separate from the logic.

The controller runs on the server as well and it is the interface between the client and server as depicted on figure 3.2 The controller receives requests from the client (containing an entity), invokes the corresponding logic in the model module on that entity and passes the resulting entity back to the client.

The view module (figure 3.2) runs on the client side using Java Web Start (JWS) [64]. Additionally, this framework provides a support library that implements many functions needed by the model, view and controller portions. Most of the widgets needed by the view are included in this library: forms, tables, trees, wafer map data displays, charts, histograms, etc.

Finally, the application entities (POJOs) are defined in the model portion and the same definitions are included in the view part. This way, model, view and controller share the same entities on the client and server sides.

3.3.5 Support libraries

Support libraries have been developed to encapsulate and reuse the common functionality that is needed for the client and server side operations [17]

in the probe and final test areas. A few open source libraries are also used. This approach standardizes the look and feel of the user interface for these applications, decreasing user training time.

By using these libraries to support the most common functions, application development time is also greatly reduced. The libraries are optimized as well to decrease client footprint, allowing fast installation and upgrade.

Moreover, most of the screens for these applications are coded as textual definitions avoiding the need to write code. The client side library transforms the screen textual definitions into forms, tables, trees, grids, and calendars during execution. The client side library also provides charts, histograms, wafer display, and other utilities.

Another function provided by the support libraries is to encapsulate and compress the communication between client and server, minimizing network traffic.

In addition, open source libraries have been used to perform data storage, user authentication, and XML marshaling.

3.3.5.1 Apache OJB

Apache OJB [3] is an object relational bridge which provides data persistence and encapsulates the complexity of retrieving, updating, storing and deleting records in the database. This library takes a configuration file that links each entity to a table in the database and each entity attribute to the corresponding column in the table. A simple call to the OJB library will store or update a Java object in the database. Queries are performed by calling an OJB method with the conditions for the query. This method returns the information as Java entities (POJOs). Similarly, a call to the OJB library is used to delete Java objects from the database. There is no need to write code to create SQL statements or transform the SQL results into Java objects. Development time has been reduced significantly by using this library for data persistence.

In addition, Oracle JDBC library [62] is used as a driver for the Oracle database. Database drivers can be easily replaced to interface with other database systems besides Oracle.

3.3.5.2 ExoLab Castor

ExoLab Castor [16] is a library used for XML marshalling and unmarshalling. Similar to Apache OJB [3], Castor takes a configuration file that

maps every attribute in the entities to the corresponding attribute in an XML document. With a simple call to the marshal and unmarshalling methods in the Castor library, POJOs are easily converted to an XML document and vice versa.

Applications often need to exchange information in XML format. This library greatly reduces the amount of code needed to parse and generate XML documents.

3.3.5.3 Miscellaneous libraries

A few more open source libraries are included in this framework.

- Oracle JavaMail [66] is used to send emails with notifications and attachments.
- Java Excel API [26] to create Excel documents with tabular data.
- Oracle Java JNDI [63] is a useful library that allows user authentication with the company LDAP server.

3.3.6 XML screens definitions

Most of the screens and menus for the applications presented in this dissertation are specified as textual definitions in XML format. Forms, tables, trees, grids, and menus are generated during execution time from these descriptions. The XML definition for a screen makes a reference to the entity that will be used to hold the data items for that screen. It also specifies the attributes that will be displayed, field length, whether each item is required or not, a list of allowed values for the attributes, and a set of actions available from each screen. Introspection is used to access and manipulate POJO attributes, according to the XML screen definitions.

The following standard actions are automatically implemented by the support libraries for the client and server sides. No coding is required to perform these actions:

- Save. Save form data in the database.
- Cancel. Aborts operation.
- Add. From the table view, it creates a blank form to enter data for the entity.

- Delete. From the table view, it deletes the record selected.
- Export. From the table view, it exports all the records to Excel.
- Help. It opens another window with the help corresponding to that screen.
- From the table view, double-clicking on a record will invoke the update function which displays a form with the data items to be modified.

3.4 Station Controller 2

As mentioned in chapter 1, three applications have been developed using this architecture and framework. Station Controller 2 (SC2) is one of these applications. SC2 is an advanced operator interface that controls the operation of both the tester and handler hardware in final test. SC2 main functionality is listed below:

- SC2 setups the tester by loading the test program.
- SC2 also sets up the handler. If supported, a handler recipe is loaded otherwise the temperature and other items are set.
- It configures sort bin trays and bin categories.
- It monitors lot yield, site delta yield, and consecutive failure limits and it will disable problematic sites. It also monitors handler recipe or temperature settings. And it will set them again if they have been changed in the middle of a lot.
- Additionally, if testing at cold temperature for a long period of time, SC2 will auto thaw the handler if previous lot was testing at cold and next lot is being tested at room temperature.

SC2 also provides centralized configuration for tester and handler settings:

- Tester and handler setups are stored in a database.
- Setups can be edited or viewed from any computer within the Freescale network.

- A history of tester setups is kept, including what was changed and by whom. This allows traceability and rolling back to a previous release if needed.
- Setups can be exported to an XML file and sent to other factories for uploading to their SC2 database.
- Additionally, SC2 implements a production control feature that only allows loading approved tester setups.

Other SC2 features:

- SC2 is a Java based application that can run on any tester that supports Java. Moreover, automatic upgrades are performed seamless using Java WebStart. There is no downtime required.
- The core of the SC2 application and the tester control software are decoupled which provides a reduction in development time and testing for adding a new tester type.
- SC2 sends EPR++ events to TTT with detailed utilization data as described in chapter 5.
- SC2 has reduced operator training since all testers use the same operator interface.
- SC2 integrates with the Manufacturing Execution System (MES) to download lot attributes and processing parameters. Additionally, at the end of lot, a record of the setups that were used is uploaded to the MES system for traceability.
- SC2 easily supports new handler types by adding drivers for them. These drivers are decoupled and isolated from the main logic.

3.4.1 SC2 architecture, framework and MVC division

The architecture and framework described in section 3.3 were used to develop SC2. The client/server architecture and the Model Controller View (MVC) division can be appreciated on figure 3.4. There is a Java packages for each one of the model, view and controller portions. Additionally, the application entities are implemented as POJOs in the entity package under the model. This entity package is also shared by the view portion.

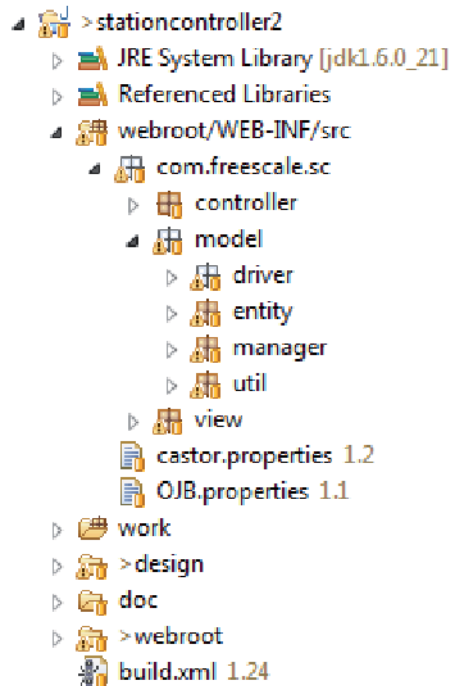


Figure 3.4: Station Controller 2 MVC division

The support libraries developed for this framework and the open source libraries used are accessible to all the components of this application.

An example of the client/server communication is depicted on figure 3.5. The first three lines are part of the client side (view portion) and the rest corresponds to the controller on the server side. For instance, to retrieve all the handler setups whose input/output type is GPIB, only those few lines of code are needed.

The client instantiates a HandlerSetup object and its attribute ioType is set to the string GPIB. Then the client sends a request to the server side passing the reason for the request (getHandlerSetups.do) and the object just created. The client blocks until the server sends the results or the request times out.

The server side overrides the process method in the controller to carry out all the different requests. The framework passes the action (getHandlerSetups in this case) and the object (handlerSetup object created by the client) to the process method. In this example, the search method is invoked on the input entity and the result is sent back to the client. The framework takes care of all the client/server communication, database query and error handling.

```
// Client side
HandlerSetup handlerSetup = new HandlerSetup();
handlerSetup.ioType = "GPIB";
Vector handlerSetups = (Vector)request("getHandlerSetups.do",handlerSetup);

// Server side
protected Object process(String action, Object input) throws Throwable
{
    if (action.equals("getHandlerSetups")) return search(input);
    :
    else return null;
}
```

Figure 3.5: Client/Server communication

In a previous version of SC2 without this framework, hundreds of lines of code had to be implemented to perform this functionality.

3.4.2 SC2 XML screens definitions

As described in section 3.3.6, most of the screens can be defined as textual descriptions in XML. SC2 is no exception. For instance, figure 3.6 shows the textual definitions and the few lines of code needed to create the table and form used to perform create, read, update, and delete (CRUD) operations on handler setups.

The table and form generated by these definitions are shown on figures 3.7 and 3.8 respectively. The first line of code on figure 3.6 creates an instance of HandlerSetups with extends the class Table provided by the framework. The parameter "HandlerSetups" in the constructor indicates the screen definition for that table which has the class that holds the data items for each record in the table (HandlerSetup), the attributes to display (name, create-Date, and user) and the list of actions for the pop-up menu (Add, Delete, Copy, Paste, Download, and Upload) as seen on figure 3.7. The framework invokes the add method when the Add option is selected from the pop-up menu and invokes the edit method when a record is double-clicked on the table. Add and edit methods have been overridden to display a form with an empty HandlerSetup object (add) or the HandlerSetup object selected (edit). The parameter "HandlerSetup" passed to the Form constructor

```

<screen name="HandlerSetups">
  <object>com.fsl.sc2.model.entity.HandlerSetup</object>
  <properties>handlerSetup,createDate,user</properties>
  <popup>Add,Delete,|,Copy,Paste,|,Download,Upload</popup>
</screen>

<screen name="HandlerSetup">
  <properties>handlerSetup, handlerType, iOType, GPIBTermination, GPIBBoardNumber,
  GPIBInstrNumber, GPIBIP, serialPort, handlerToTesterSite</properties>
  <required>Y,Y,Y,N,N,N,N,N</required>
  <length>12,16,10,10,4,4,12,4</length>
  <values field="handlerType">Deltacastle,Deltaflex</values>
  <values field="iOType">GPIB,LAN-GPIB,Serial,TCP,Parallel</values>
  <values field="GPIBTermination">CR+LF,CR,LF,NONE</values>
  <values field="GPIBBoardNumbe">0,1,2,3</values>
  <values field="GPIBInstrNumber">0,1,2,3</values>
  <values field="serialPort">0,1,2,3,4,5,6,7</values>
  <buttons>Save,Cancel</buttons>
</screen>

new HandlerSetups("HandlerSetups", true);
.
.
public class HandlerSetups extends Table
{
  public void add() throws Throwable {new Form("HandlerSetup",new HandlerSetup());}

  public void edit() throws Throwable {new Form("HandlerSetup", (HandlerSetup)getSelection());}
}

```

Figure 3.6: *Handler setup screens textual definitions*

indicates the screen definition for that form. This definition (figure 3.6) contains the fields to display, the length for each field, the list of possible values for some of the fields and the actions for that form as seen on figure 3.8.

The framework automatically creates the screens with the table and form as seen on figures 3.7 and 3.8. Additionally, all the following functions are automatically implemented by the framework without the need to write extra code: retrieving all the HandlerSetup objects from the database and displaying them, showing the pop-up menu, executing the add and edit functions when selected from the pop-up menu, saving in the database the new data added or updated when the button Save is clicked, and deleting selected records from the table.

With this approach, screens can be quickly and easily generated with textual descriptions and minimum code, contributing to the requirements of quick prototyping and development as well as high maintainability and upgradability.

Handler Setup	Create Date	User
HandlerSetup1	2010-08-05 15:05:57	Moreno Manuel
HandlerSetup2	2010-11-14 15:47:15	Moreno Manuel
HandlerSetup3	2010-11-14 15:48:13	Moreno Manuel
HandlerSetup4	2010-11-14 15:49:11	nuel
HandlerSetup5	2010-11-14 15:51:49	nuel
HandlerSetup6	2010-11-14 15:51:58	nuel
HandlerSetup7	2010-11-14 15:52:11	nuel

Figure 3.7: Handler setups table

3.5 Model Driven Architecture approach

Continuing the research on RapidCIM [43, 82] to automatically generate software to control a manufacturing system, we have devised a Model Driven Architecture (MDA) approach that has successfully been used to decrease development time and generate design documents, increasing application maintainability [17, 27]. Models are written as a UML [81] class diagram, representing the entities that make up the application, their attributes and their relationships plus XML screens definitions for forms, tables and menus that will be used to maintain those entities.

Figure 3.9 shows a small part of the Platform Independent Model (PIM) for SC2. This model is composed of the class diagram (TesterSetup, HandlerSetup and User classes) and the screen definitions for those classes. The Obeo Acceleo [54] generation engine is used to transform the PIM into the Platform Specific Model (PSM) as figure 3.10 depicts. Acceleo [54] implements the OMG’s model-to-text specification [57] and it is part of the Eclipse Model Development Tools (MDT) project. As mentioned in section 3.3.2, Eclipse was chosen as the Integrated Development Environment (IDE). Using Acceleo is preferred to commercial solutions such as IBM Rhapsody or open source such as Topcased due to the simplicity of the Eclipse MDT tools and to avoid a steep learning curve.

The model (class diagrams plus XML text definitions) is fed to the code generation engine (Acceleo) plus templates that define how to gener-

Setup

Handler Setup

Handler Setup: HandlerSetup1

Handler Type: Simulator

IO Type: GPIB

GPIB Termination: CR+LF

GPIB Board Number: 0

GPIB Instr Number: 0

GPIB IP Address:

Serial Port:

Handler to Tester Site:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Reset

Save Cancel

Figure 3.8: *Handler setup form*

ate the code. The result is Java code plus the database schema needed to store the entities defined in the class diagram

These templates [38] are written in MOF Model to Text Transformation Language (MOFM2T) [57]. They were created to generate code according to the architecture and framework defined in section 3.3. The templates are used by the code generation engine (Acceleo) to extract the information from the UML class diagram and the XML screens definitions and generate Java code plus configuration files and the database schema. Figure 3.11 shows an example of these templates, in particular the one used to generate the Java entities. The template loops through the classes in the class

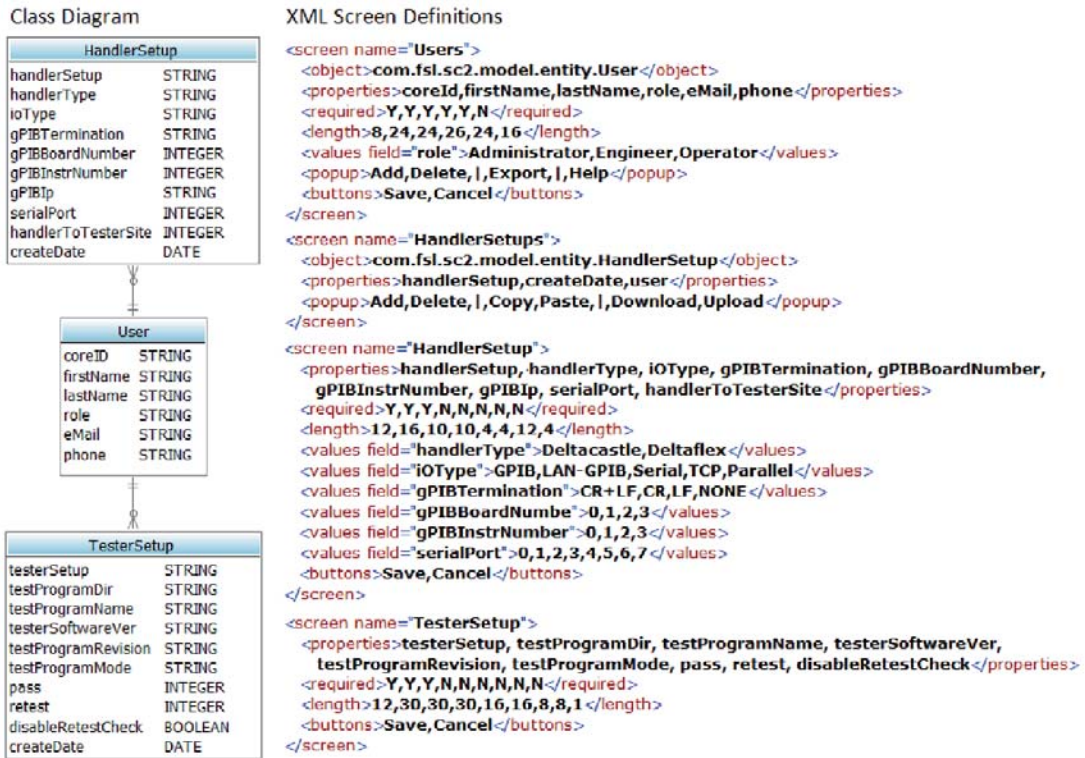


Figure 3.9: Platform Independent Model

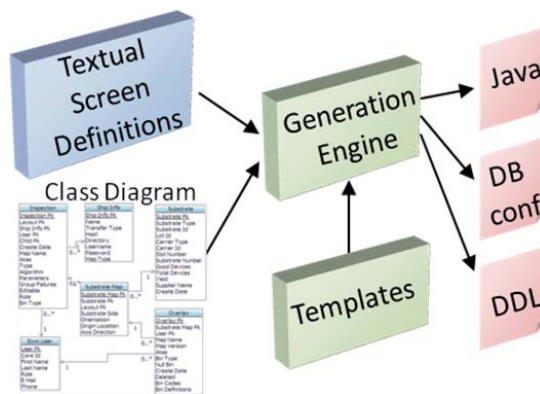


Figure 3.10: PIM to PSM transformation

```

[module entity('http://www.eclipse.org/uml2/2.1.0/UML'//)]
[template public generateEntities(p : Package)]
[for (c : Class | p.packageElement)]
[file ('WEB-INF/src/'.concat(p.name.substituteAll('.', '/')).concat
    ('/model/entity/').concat(c.name).concat('.java'), false)]
package [p.name/].model.entity;
import java.io.Serializable;

public class [c.name.toupperFirst()] implements Serializable
    {
        public Integer [c.name.toLowerFirst().concat('Pk')/];
        [for (po : Property | c.getAllAttributes())]
        public [po.type.name/] [po.name/];
        [/for]
    }
[/file]
[/for]
[/template]

```

Figure 3.11: Entity template

```

[module schema('http://www.eclipse.org/uml2/2.1.0/UML'//)]
[template public generateSchema(p : Package)]
[file ('../design/schema.sql', false)]
[for (c : Class | p.packageElement)]
    CREATE TABLE [c.name.toupper()] (
        [c.name.toLowerFirst().concat('Pk').toupper()] INTEGER NOT NULL
        [for (po : Property | c.getAllAttributes())]
        . [po.name.toupper()] VARCHAR(32)
        [/for]
    );

    CREATE SEQUENCE [c.name.toupper()/_SEQ] NOCACHE;
[/for]
[/file]
[/template]

```

Figure 3.12: Database template

model, generating a Java class (POJO) for each one with the attributes defined in each class. Similarly, Figure 3.12 shows the template used to generate the database Data Definition Language (DDL) artifacts.

The output of this code generator is a simple, fully functional application that implements the basic functionality for entity maintenance, persistence,

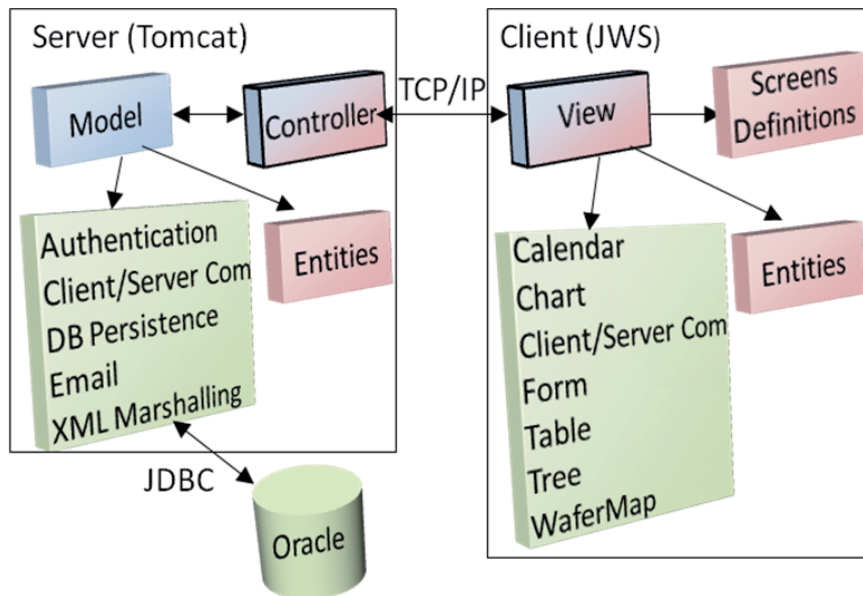


Figure 3.13: PSM and framework

marshalling, and client/server communication for the entities specified in the UML class diagram, according to the XML screens definitions. Additional logic can easily be added to this skeleton to implement the specific functionality required by the application. This code fully complies with the architecture and framework defined in section 3.3.

Figure 3.13 shows the Platform Specific Model (PSM) or code generated. The blocks in red are automatically generated from the model. The ones in green are the support libraries. The blocks in blue must be coded. As figure 3.13 shows, most of the controller and view portions are automatically generated. Entities are also automatically generated from each class in the class diagram.

Finally, the XML screen definitions are included with the code generated. The support libraries are also added to the code and will transform the XML screens definitions into forms, tables, trees, grids, and calendars during execution time. The client side support library also provides charts, histograms, wafer display, and other utilities decreasing significantly the amount of code that needs to be written.

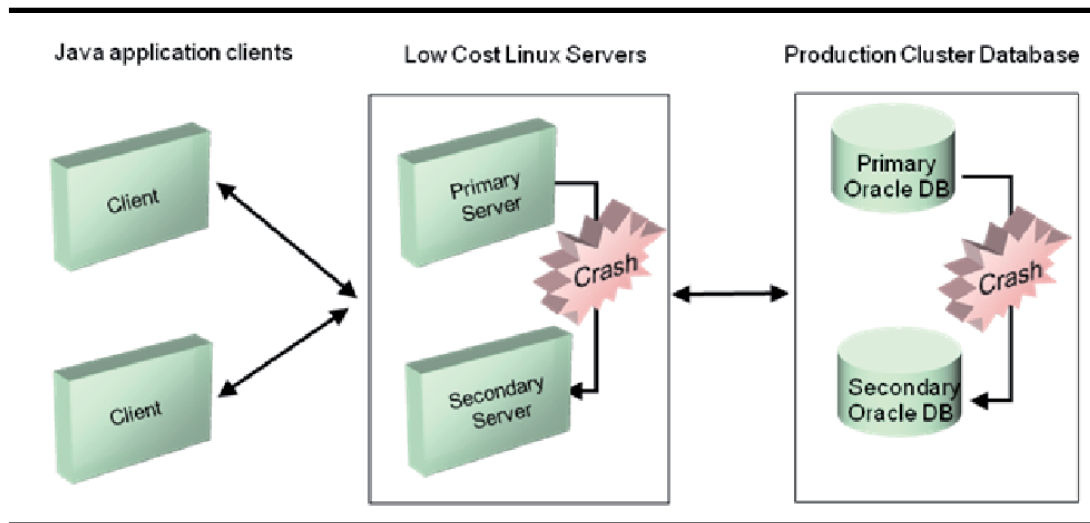


Figure 3.14: Failover architecture

3.6 Failover

The three applications generated with this framework are critical for manufacturing and must be running at all times. To avoid downtime a redundant system has been put in place as shown in figure 3.14.

The applications are setup on a computer cluster to provide failover capability in case of hardware failure. They run on the primary server. In case of hardware failure the secondary server takes over.

Similarly, the database is an Oracle cluster database which is mirrored. In case of hardware malfunction, the secondary server takes over without disruption to the system.

3.7 Results

The architecture and framework described in this chapter have been used to develop three applications running in production in the probe and final test areas in five factories in Freescale Semiconductor Inc. in Asia, Europe and North America:

- Application 1 (SC2) controls hundreds of testers.

- Application 2 (EWM) has processed data for over 10 millions of wafers to date and
- Application 3 (TTT) receives about 250K messages daily from application 1 and prober controllers.
- Although this library is tailored for probe and final test, it can be reused or easily extended to work in other semiconductor areas and other industries. In fact, this framework and architecture are currently being used for a new application in development in Freescale Semiconductor Inc. This application is a scheduler for the fabrication area [42,32] and it is expected to be deployed to three factories in North America by the end of 2013

The following goals have been achieved:

- Quick prototyping and development. With the MDA approach, a basic, fully functional application is automatically generated from the model (UML class diagram plus XML screens definitions). Logic can easily be added to the code generated without having to implement data display and persistence functions. Thus, development time has been reduced considerably.
- The three applications mentioned in this chapter were redeveloped using this architecture and framework. Table 3.1 compares the number of lines of code for the model and view portions of applications 1 and 2 before using this framework and after: a 69% and 40% reduction in lines of code written. Application 3 implemented more requirements than its predecessor so the comparison is not meaningful. The Apache Struts framework [4] was initially used to build application 2. Struts typically relies on JSP [67] and HTML [23] formatted response for the client side. As requirements for data transfer and visualization became more complex, the limitations of the HTML [23] client became apparent.
- High maintainability and upgradability. The clean organization that the framework enforces makes maintaining these applications a simple task. In addition, this framework is very conducive towards the agile development methodology. Developer efficiency is estimated to have increased at least 25%. Additionally, keeping model and implementation separate forced to review the model entirely before any code was written which has increased the quality and reliability of the final application.

	App1 before Framework	App1 with Framework	App2 before Framework	App2 with Framework
View LOC	29,271	3,754	6,305	2,633
Model LOC	17,406	10,564	8,026	5,947

Table 3.1: Applications statistics before and after using framework (LOC = Lines of Code)

- Small application footprint (CPU, memory, disk space, database, and network traffic). Table 3.2 shows statistics about the three applications implemented. This table lists the lines of code for each application, number of classes, screens, menus, and executable sizes. Moreover, the compression schema minimizes the database space needed and the network traffic. Table 3.3 displays the average data transfer sizes before and after this framework for small, medium, large and extra large objects sent between client and server. These figures can be extrapolated to database space used as well. Traffic and transmission times are reduced 93% for medium objects and 48% for extra large ones. Average data transferred between server and clients for one of these applications is 125 Mbytes each day (compressed) and the average number of requests is 30,000 per day.
- Increased application documentation. It is achieved by the use of class diagrams and XML screens descriptions in the PIM.
- Small learning curve. The framework is very easy to use. Junior developers are able to leverage this framework in less than 40 hours of training.
- Low cost development environment and enablers. All tools used are open source.
- Platform independent. The code generated is Java which runs on UNIX, Windows and Macintosh, allowing leveraging existing hardware.

	Application 1	Application 2	Application 3
Controller LOC	56	86	122
View LOC	3,754	2,633	1,270
Model LOC	10,564	5,947	2,592
Entities LOC	1,186	1,098	71
Classes	96	46	24
Screens	87	52	35
Menus	29	20	8
Executable size	16.4 Mb	8.2 Mb	6.2 Mb

Table 3.2: Applications statistics (LOC = Lines of Code)

	Before Framework	With Framework
Small object	6,618	2,100
Medium object	35,935	2,611
Large object	1,023,297	294,710
Extra large object	5,740,890	2,968,775

Table 3.3: Average network traffic in bytes before and with framework

Chapter 4

Electronic Wafer Mapping and Zero Defect Algorithms in Unit Probe

Quality means doing it right when no one is looking.

Henry Ford

Electronic devices for automotive and other critical applications must perform with an extremely low defect rate. Traditional unit probe test programs check each device individually and visual inspections can detect defects that electrical testing cannot. Statistical outlier detection algorithms can be applied to these individual results to identify devices that have high probability of failing. All defects (electrical, visual and outliers) can be combined systematically to more efficiently decrease number of Customer Quality Incidents (CQIs).

Outlier detection algorithms are a key component in screening latent defects and decreasing number of CQIs. New algorithms and enhancements to existing ones have been created during this research to improve the performance of latent defect detection. Additionally, experimental results were analyzed to determine efficiency and effectiveness of these algorithms.

4.1 Introduction

Achieving a low defect rate for semiconductor devices for the automotive and other critical applications is paramount. After the fabrication process, wafers are subject to extensive electrical (probe) and visual testing to screen out devices that do not meet the specifications. After probe, devices are sawn from the wafer, packaged (also known as assembly) and tested again (final test) to further filter out failing devices. Yet some devices contain latent defects that will manifest later in their life and will originate a Customer Quality Incident (CQI). This chapter describes the implementation of a software application in Freescale Semiconductor Inc. that systematically combines wafer maps from visual inspections, electrical probe results and applies outlier detection algorithms to decrease the number of CQIs. This application is known as Electronic Wafer Mapping (EWM) and is based on the architecture and framework presented in chapter 3.

EWM has been running in all the Freescale probe factories for several years and it is fully integrated into the manufacturing process. It collects wafer maps with visual inspection defects, probe hard bin and parametric test results. Visual inspections will detect some latent defects that electrical probing cannot. EWM also applies statistical outlier detection algorithms to those results to identify abnormal dice (outliers which have a high probability of failing). These algorithms are: SBL, BMY, GDBC, SPAT, DPAT, AEC DPAT and NNR. Also, four novel variations of these algorithms and one new algorithm have been created and implemented which have significantly increased their performance. In addition, a study was conducted with production data to determine and compare the efficiency and effectiveness of all these algorithms in identifying CQIs. This type of studies is very rare in the technical literature.

After probe, EWM combines all the defects (visual, electrical and statistical outlier defects) into a single wafer map with good and bad dice to specify which devices need to be discarded during assembly. As part of this research, the SEMI E142 industry specification [75] for wafer map data transfer was elaborated in collaboration with other semiconductor industries which has significantly decreased application development time. Freescale was one of the first companies to implement this standard and many other companies are using it today.

There exist commercial applications that address some of these areas but do not consistently and robustly combine all defects. Moreover, adapting these applications to integrate into the existing manufacturing process is very costly.

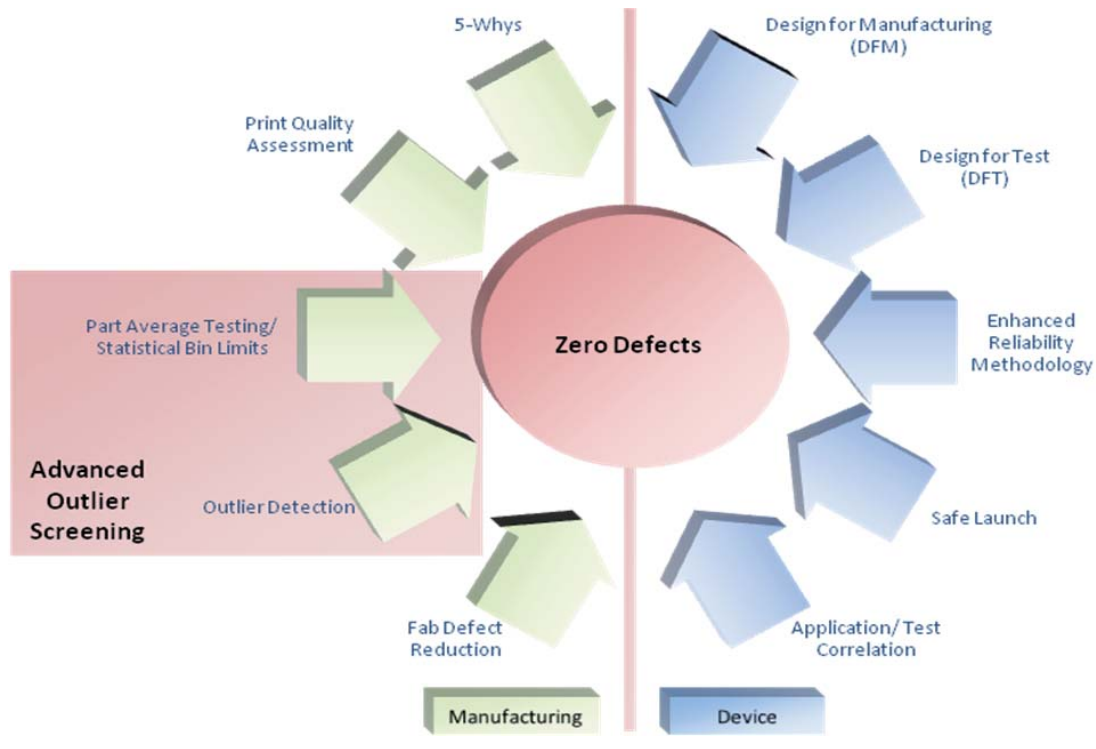


Figure 4.1: Freescale's quality initiatives

Throughout several years, requirements in this area were collected from all probe floors and prioritized to address the most important ones first. Additionally, alongside the deployment of EWM, processes across all probe floors were standardized as much as possible, which minimized variation and development time. As a consequence, quality improved since variability and quality are inversely related.

The work described in this chapter is part of the global initiatives implemented in Freescale to ensure that the number of defective devices shipped to customers is as low as possible. These initiatives have been implemented during the design and manufacturing phases. As depicted on figure 4.1, the research presented in this chapter covers the Advanced Outlier Screening initiative, contributing to the corporate wide Zero Defect efforts.

This chapter is organized as follows: section 4.2 reviews the related work, section 4.3 addresses existing outlier detection methods. Section 4.4 introduces new enhancements to these existing methods. Section 4.5 describes the EWM functionality, including automation and integration with current man-

ufacturing processes. Section 4.6 addresses development of the SEMI E142 specification for substrate mapping [75]. Section 4.7 details the user interface for this application. Finally, section 4.8 details the achieved results.

4.2 Related work

Automotive electrical modules must perform below 10 Defects per Million (DPM) [41]. Since each module comprises hundreds of components, each one must have a DPM below 1 which is virtually zero defects.

A key factor for achieving zero defects is outlier detection methods. According to Migl [41], these methods have varying degrees of efficiency but can significantly improve quality. The benefit varies by device, technology and implementation method.

Other authors (Mann [35], Marinissen et al. [37]) have surveyed different techniques applied to decreasing device defects. Most of these test methods improve device quality and reliability at a yield loss. Ultimately, they attempt to optimize the balance between yield and product quality.

Defects tend to cluster as Ooi et al. [60, 61] show. They propose an approach based on a die-level neighborhood predictive model to successfully screen latent defects. Their solution combines data mining with a defect-cluster extraction schema. They have observed from production data that failing dice with traceable causes tend to form clusters at the wafer level or hot spots at the wafer-lot level. For semiconductor wafers, defect clusters appear as areas of random defect patterns at the wafer level that are the result of random perturbations in the manufacturing process (particle related). Local defect patterns at the wafer-lot level are mostly located at a specific location and are process related.

Similarly, Barnett et al. [6] arrived at the conclusion that some defects on wafers are not randomly distributed, but tend to cluster, based on burn-in data from 77,000 microprocessors manufactured at IBM. Also, latent defects (CQIs) are normally found to cluster with killer defects detected during probe. A later paper from the same authors [7] analyzed the same set of data results and introduced a yield–reliability model based on the fact that defects on a wafer have a tendency to cluster. This study also shows that conclusion can be utilized to group devices into sets of different reliability based on how many of their neighbors failed. Dice that pass all the probe tests but belong to a region with many defective dice have a higher probability of latent defects. Similarly, Ooi et al. [58] show a strong correlation between

probe data and device reliability. In another paper, Ooi et al. [59] investigate the use of defect characteristic models as yield models to screen latent defects for wafers with defect clusters. They propose and utilize a yield mining framework that guides manufacturers in determining if their device has a spatial relationship between the probe defects and latent defects.

Riordan et al. [69] present a die-level, predictive defect model. This model is shown to be twice as efficient in identifying unreliable material as traditional wafer or lot level methods. Their model includes die location yield and die local yield. Optimum nearest neighbor effect is at most two die in each direction (24 nearest neighbors) regardless of die size. Also, they prove that die location yield is useful if the fabrication process introduces a defect in a particular position on the wafer.

Likewise, Sing et al. [78] present experimental results showing that by using information about defect clusters, testing costs can be reduced by separating high quality dice that have a lower likelihood of failing future tests.

Anderson [1] proposes the Negative Binomial distribution to model yield distribution. The clustering parameter needs to be defined and he recommends a value of 2 where 1 represents a high degree of clustering to infinity which represents no clustering at all.

A different family of outlier detection methods is Part Averaging Testing (PAT). These are mostly applied to automotive devices that need to be qualified and tested to meet very demanding requirements. These specifications are described in the Automotive Electronics Council standard AEC Q-100 [56].

Ohletz et al. [55, 56] discuss two methods included in the AEC Q-100 specifications which are DPAT and Statistical Bin Analysis (SBA) which comprises the statistical evaluation of failed bins. Similarly, Solanki et al. [79] apply DPAT in testing Inertial Micro-Electro Mechanical System (MEMS) devices which are used as sensors. They claim that fabrication variations and defects can be caught using DPAT. These parts are referred to as outliers and it is thought that, even though they passed the test program specification limits, they represent a portion of the population that usually will fail in the future.

Manufacturers must use innovative statistical methods to comply with zero-defect product quality requirements. Marinissen et al. [37] address test methodologies that target those requirements. Outlier detection, part average testing, and neighborhood screening are a few examples of those statistical methods. It is accepted that if all devices are tested the same way, dice

from low-yield wafers, and low-yield regions within wafers, will likely be defective devices. Also, they show that the Nearest Neighbor Residual (NNR) method is an effective predictor of device failures [37], specially when testing the supply current (I_{dd}) in the quiescent state (I_{ddq}).

In addition to these statistical methods, semiconductor manufacturing often employs visual inspection during the wafer fabrication process in order to control particle contamination and structural defects. Mullenix et al [51] developed a formula to estimate the effect of various types of defects on the wafer yield.

Shanka et al. [77] built an online system that performed complete visual inspections of 8-inch wafers. The system had to screen flaws as small as $1\mu\text{m}$ that are unacceptable in the manufacturing process. Defective regions were identified automatically with the aid of a scanning microscope or manually by operators visually inspecting wafers and marking defective regions. Although manual detection may not be thorough and may ignore certain defects while incurring personnel costs.

Chang et al. [10] propose a self-organizing neural network to perform unsupervised visual defect detection. They showed that the proposed method successfully identifies defective regions efficiently using manufacturing data. Other authors propose using linear regression as described by Roehr et al. [70] and Schuermyer, et al. [71]. They select pairs of tests, calculate a linear regression and discard devices that are far from the regression line.

This chapter describes a holistic approach to defect detection and CQI reduction by systematically combining visual and electrical defects plus statistical outlier detection methods. Visual inspections identify some latent defects that electrical probe cannot screen out. A software application (EWM) has been created to cover this space and is completely integrated in the Freescale probe manufacturing area, running on all the probe floors. This application was built using the framework, architecture and MDA approach described in chapter 3.

Some novel modifications to these outlier detection algorithms and a new algorithm have been developed and implemented. Additionally, an analysis has been conducted on 289,080 dices with 26 CQIs to determine the efficiency and effectiveness of these algorithms which reveals the significant increase in performance of the new modifications. Such studies are extremely rare in the technical literature. Finally, the SEMI E142 standard [75] for wafer map data transfer was elaborated in collaboration with other semiconductor companies and leveraged in this research.

There are commercial solutions that cover part of this spectrum but fail to consistently combine all defects in a robust, reliable manner and integrate seamlessly in the existing manufacturing process.

4.3 Classic outlier detection methods

Several outlier detection methods described in the technical literature have been implemented in EWM. They identify dice that seem abnormal by using statistical methods. The dice are marked as defective and not picked during assembly. Processing time is less than 20 seconds per wafer for the most complex algorithm and about 2 seconds on average.

These algorithms can be divided into three categories:

- Statistical bin limits: This category comprises Statistical Bin Limits (SBL) and Below Minimum Yield (BMY).
- Geographical defects: Good Die in a Bad Cluster (GDBC)
- Part average defects: Static Part Averaging Testing (SPAT), Dynamic Part Averaging Testing (DPAT), Automotive Electronics Council Dynamic Part Averaging Testing (AEC DPAT), and Nearest Neighbor Residual (NNR).

Some of these methods are in general more effective and efficient than others. A method is more efficient if it identifies fewer good devices as bad. An algorithm is more effective if more bad dice are marked as bad.

Figure 4.2 depicts the efficiency and effectiveness of these methods according to the technical literature. SBL and BMY are low efficiency and effectiveness algorithms. GDBC is more efficient. PAT algorithms (SPAT, DPAT) increase in both, efficiency and effectiveness, in that order. Finally NNR is the most effective and efficient of all the algorithms.

The enhancements to these algorithms described in section 4.4 increase the performance of these classic methods.

These classic algorithms are described in detail below:

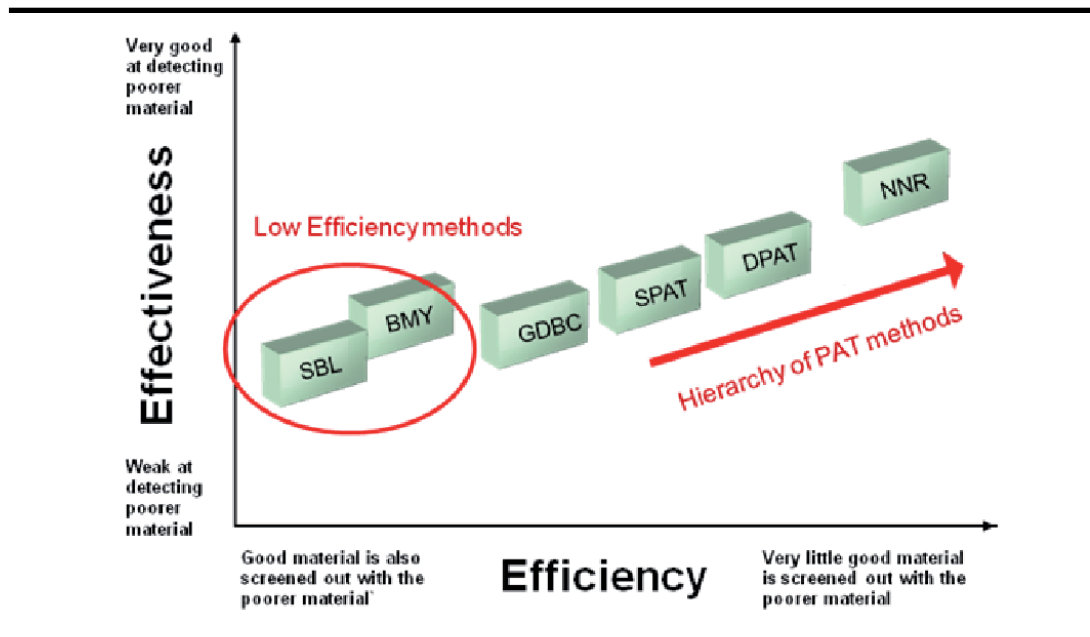


Figure 4.2: Statistical outlier detection algorithms

4.3.1 SBL, BMY

SBL stands for Statistical Bin Limits. Wafers that have a percentage of bins higher than the historical limit should be put on hold for analysis to determine if this unusually high percentage is acceptable. These limits are calculated from historical data [25] outside of EWM and are entered in EWM. EWM automatically puts a lot on hold in the Manufacturing Executing System (MES) if limits are exceeded. An engineer will analyze the wafer and determine if it can continue testing or if it needs to be scrapped.

BMY stands for Below Minimum Yield. It is similar to SBL. In this case, a lot is put on hold if any wafer has a yield lower than the historical yield (plus a buffer). The historical yield per wafer is calculated for each device outside of EWM and a buffer is added [25]. This yield is entered for each device. EWM automatically puts a lot on hold in the MES if the yield for a wafer is below the minimum entered.

The parameters for these two algorithms are:

- List of bin codes and upper percentage limits. The entire lot will be put on hold if any of these upper limits is violated.

4.3.2 GDBC

GDBC stands for Good Die in a Bad Cluster. Electric test results mark dice as good (all tests were passed) or bad (one or more test failed). This algorithm marks a good die as bad when it is surrounded by many bad dice.

The parameter for this algorithm is a threshold that indicates the percentage of bad neighbors needed to mark a good die as defective. If the device is for a critical application, this parameter will be set lower (more aggressive setting).

The rationale behind this method is that defects are normally clustered: a good dice surrounded by many defective ones has a high probability to contain latent defects. The higher the percentage of defective neighbors, the higher the probability the device will fail in the future [7].

A GDBC example is shown on figure 4.3 with 87.5% threshold, meaning that a good die is marked as defective if 87.5% of its immediate neighbors or more are defective. Good dice are represented in green and bad in yellow. The GDBC algorithm identified 5 dice (in red) as bad dice which were almost completely surrounded by bad dice.

4.3.3 SPAT, DPAT, AEC DPAT, NNR

Parametric test results need to be collected for each die to be able to run SPAT, DPAT, AEC DPAT, and NNR. Hundreds of tests are typically executed.

SPAT stands for Static Part Average Testing. For each parametric test, a lower an upper limits are determined to screen out dice that have a test result outside these specification limits. A series of electrical tests are performed and fed to EWM which will determine if the test result for each die is within the limits for that test. Any die with one or more test results outside the test limits is marked as defective.

The parameters for SPAT are a list of tests and associated lower and upper limits.

DPAT stands for Dynamic Part Average Testing. Unlike SPAT, limits are calculated for each wafer and test dynamically. Outliers are determined according to equation 4.1 that is applied to each test individually, where the mean and standard deviation are computed for the values for that test on the

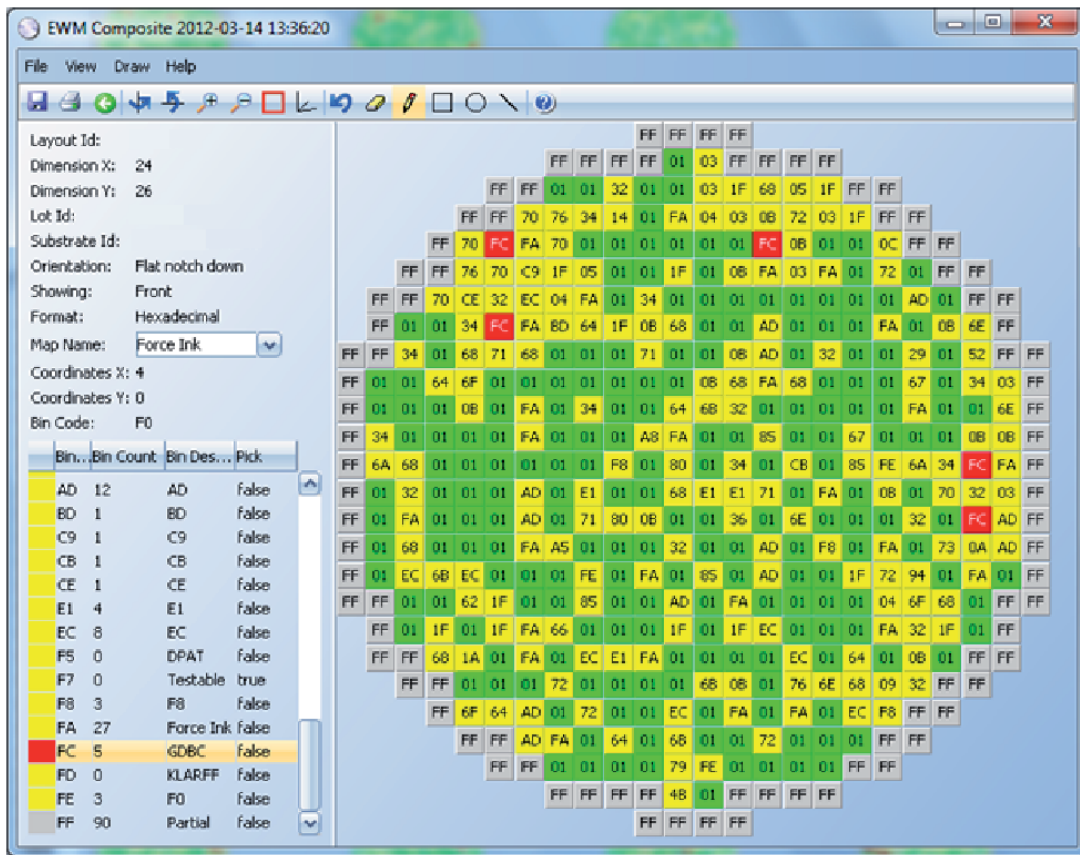


Figure 4.3: GDBC example

entire wafer (excluding results for defective devices), and k (sigma multiplier) is the parameter given for each test. Values outside the limits are considered outliers. The k multiplier is normally set to 6 (outliers are considered if outside 6 times the standard deviation from the mean). The smaller parameter k is, the smaller the upper and lower limits and the more dice that will be marked as defective.

$$\begin{aligned} \text{Upper Limit} &= \text{mean} + k \times \text{std deviation} \\ \text{Lower Limit} &= \text{mean} - k \times \text{std deviation} \end{aligned} \tag{4.1}$$

Figure 4.4 shows an example of a parametric test for a wafer. Each die has a value assigned which is a real number and it is the result of the electrical test for that die. There are hundreds of test performed on a wafer.

The rationale behind DPAT is that test results too far away from the mean

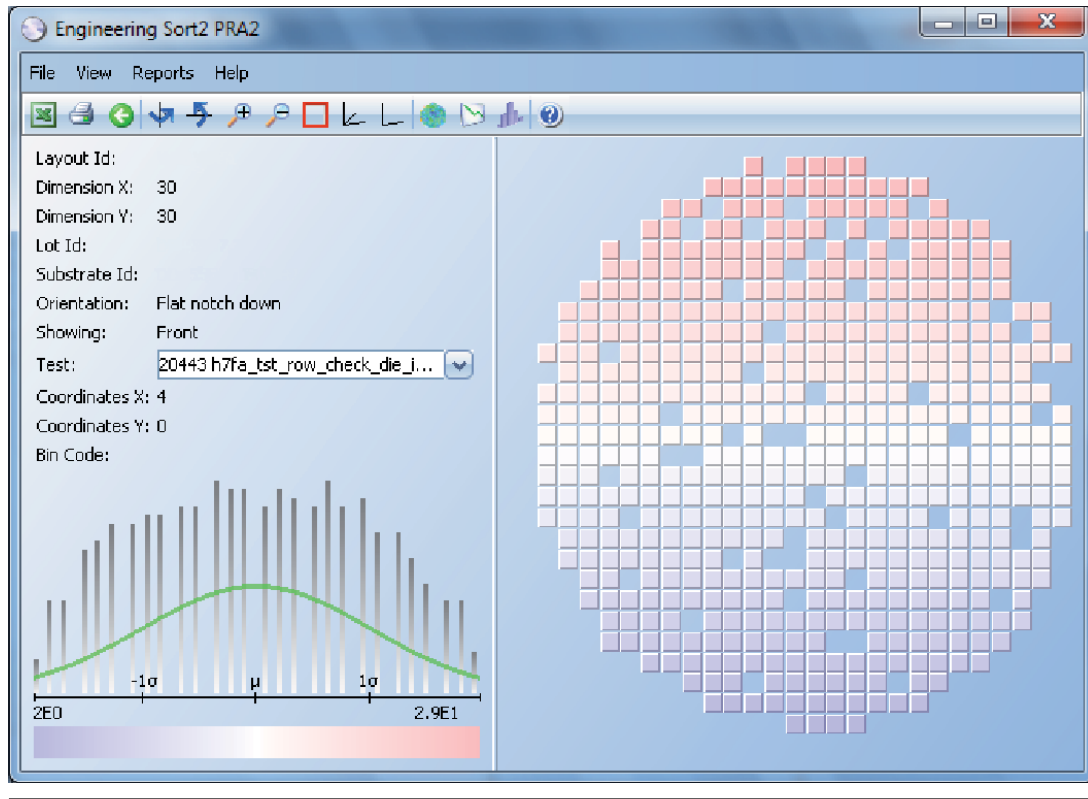


Figure 4.4: *DPAT example*

are suspicious. The further away from the mean, the higher the probability of that die to fail in the future [55, 56].

The parameters for DPAT are a list of tests and associated sigma multiplier (k).

Another version of DPAT that has been implemented in EWM is known as AEC DPAT which stands for Automotive Electronics Council Dynamic Part Average Test [56]. For each test, upper and lower limits are calculated including all the values for non-defective dice for a wafer as shown on equation 4.2, where p_1 and p_{99} are the 1st and 99th percentiles.

$$\begin{aligned} \text{Upper Limit} &= \text{median} + k \times (p_{99} - \text{median}) \times 0.43 \\ \text{Lower Limit} &= \text{median} - k \times (\text{median} - p_1) \times 0.43 \end{aligned} \quad (4.2)$$

This algorithm is in general more robust than DPAT since it discards possible outliers from the dispersion calculation and includes certain correction

for skewness in the distribution, increasing the efficiency. By avoiding the first and 99th percentiles, possible outliers are discarded from the limits calculation increasing statistical robustness. And by using a lower and upper limit, distribution asymmetry is factored in.

The parameters for AEC DPAT are a list of tests and associated sigma multiplier (k). As mentioned above, the k multiplier is used to be more or less aggressive with the test in particular.

Another algorithm implemented is Nearest Neighbor Residual (NNR) [83]. The average test result of a die neighborhood (expected value) is subtracted from the die test result (measured value) as shown on figure 4.5 and equation 4.3. That residual is a real number and is then used to apply DPAT. This algorithm is effective when test results follow a gradient on the wafer as depicted on figure 4.4. Results at the top of the wafer are greater (red color) than results at the bottom (blue). The darker the red, the further to the right of the mean and the darker the blue, the further to the left of the mean. This algorithm highlights values that are significantly above or below the average values of their neighborhood. Engineers analyze test results and apply NNR to tests that display that type of gradient instead of the classic DPAT algorithm.

$$\begin{aligned} \text{Residual}_j &= \text{Measured}_j - \text{Expected}_j \\ \text{Expected}_j &= \frac{\sum_{j \neq i} w_i X_i}{\sum_{j \neq i} w_i} \\ w_i &= \exp \left[-\frac{(\text{die}X_i - \text{die}X_j)^2 + (\text{die}Y_i - \text{die}Y_j)^2}{2\lambda^2} \right] \end{aligned} \quad (4.3)$$

The parameters for NNR are:

- Lambda (λ), determines the number of adjacent dice to be included in the neighborhood value calculation as shown on equation 4.3. The higher λ is, the more dice will be included in the neighborhood. Typical λ values are in the 1.5 to 2.0 range.
- List of tests and associated sigma multiplier as defined in DPAT.

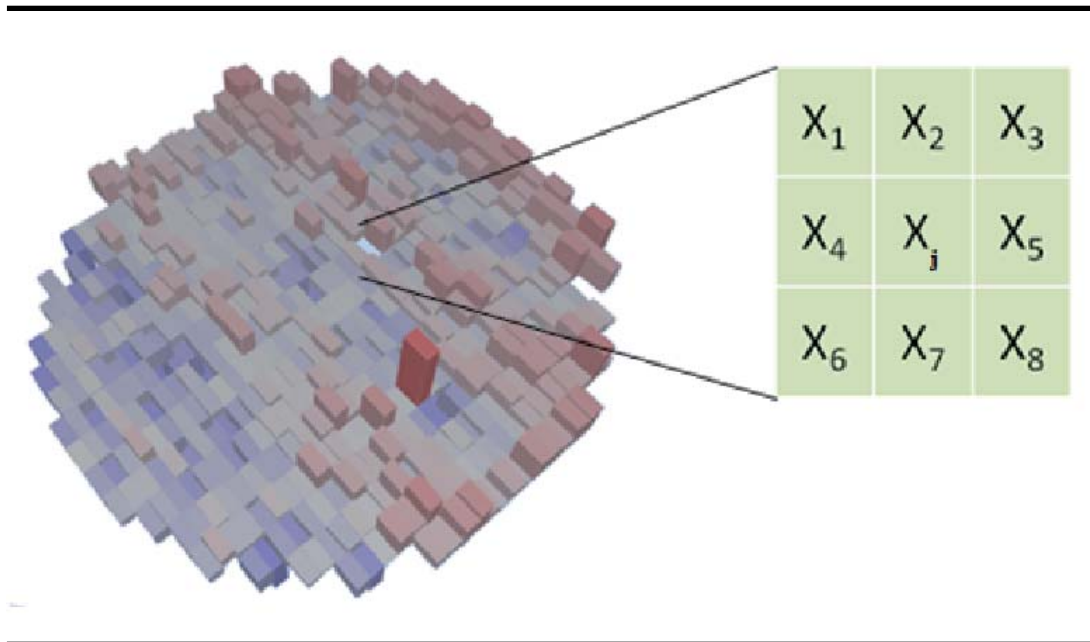


Figure 4.5: *Nearest Neighbour Residual*

4.4 New outlier detection methods

4.4.1 GDBC SB

A novel variation of GDBC named GDBC SB (Good Die in a Bad Cluster with Specific Bins) has been created which only considers bad bins a specific set of defective bins. The parameters for this new algorithm are the threshold of bad neighbors and the list of bins that are consider defective. The rationale behind this method is that some defects (defined by the bin number) are more likely than others to affect adjacent dice.

The list of defective bins included in this algorithm are determined by analyzing past CQIs and calculating the probably that a given bin number has an adjacent CQI. Bins with the highest probability will be included in the list. All the known CQIs should be included in this analysis. In the absence of CQIs, all defective bins are included which defaults to the classic GDBC method. The performance of this method increases significantly with respect to the classic GDBC as detailed in section 4.8.

4.4.2 BBBC

BBBC stands for Bad Bin in a Bad Cluster. This algorithm identifies clusters of bad dice and then marks all good dice surrounding the cluster as bad. The rationale behind this method is the fact that certain clusters of defects have a high correlation to latent defects in neighboring dice. Similar to GDBC SB, the set of bins that make up the clusters are chosen based on past CQIs. If there are no CQIs, defect causes (bin numbers) that are suspected to spread to neighboring dice are used.

A BBBC example is displayed on figure 4.6. Good dice are represented in green and bad in yellow. Clusters of bins 25 and 41 are identified first (orange color). Only bins 25 and 41 that are surrounded by enough bins 25 or 41 are considered part of a cluster. Then, all good dice surrounding the cluster are marked as bad (red color). The parameters for this algorithm are:

- List of bins that will be considered to form a cluster. In the example above, bins 25 and 41 only are used to determine clusters of bad dice.
- Threshold that indicates the percentage of bad neighbors that any of those bins need to have in order to form part of a cluster.
- Minimum Cluster. If specified, only clusters that have the minimum number of dice specified by this parameter will be considered. If not specified, the minimum cluster size is one.

4.4.3 DPAT ST

DPAT ST stands for Dynamic Part Average Testing with Specific Tests. In the classic DPAT algorithm, using tests with no correlation to CQIs deteriorates the efficiency of this method. A novel variation of this method has been developed which only includes tests that have a strong correlation with past CQIs. All known CQIs should be considered. In order to determine which tests will be used, device engineers utilize EWM by running the CQI and test correlation reports described in section 4.7.6. In the absence of CQIs, a list of key tests recommended by the division is used. The performance of DPAT improves considerably with this enhancement as shown in section 4.8. The reason behind this improvement is that some electrical measurements are more related to latent defects than others. By avoiding (or by

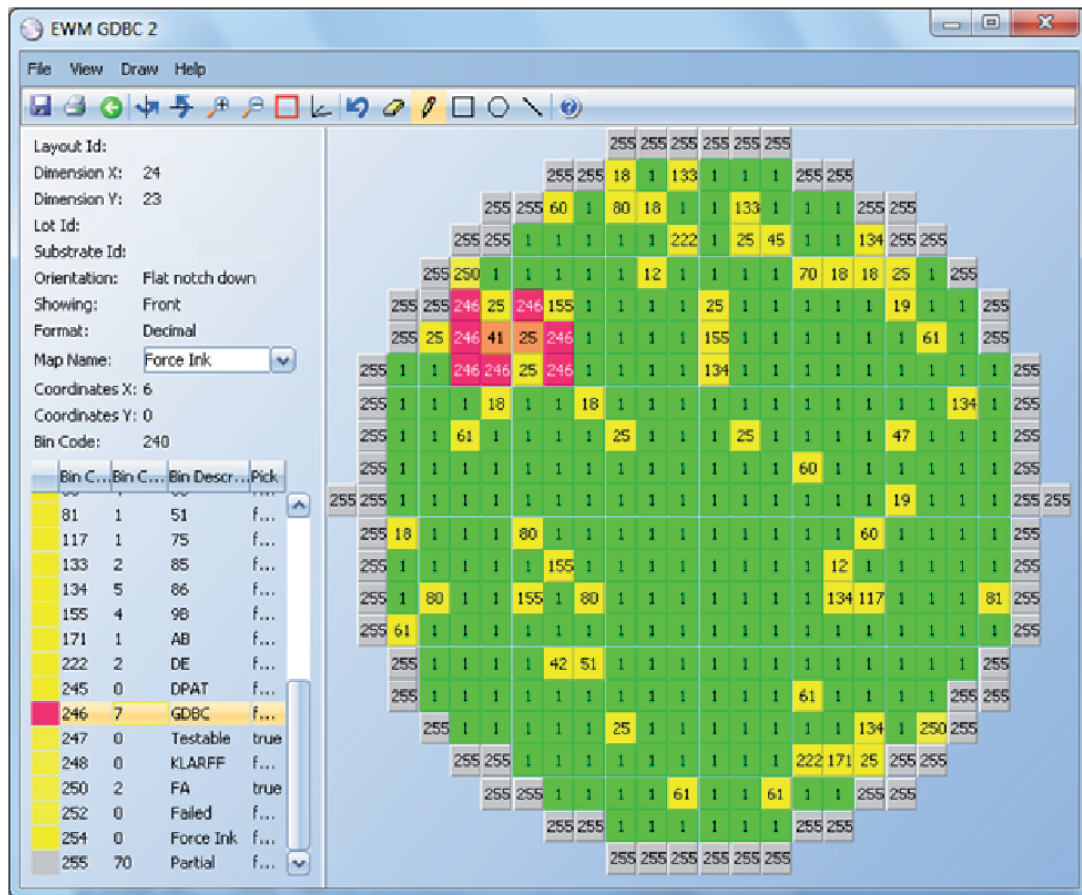


Figure 4.6: BBBC example

being less rigorous) with tests that have a lower correlation with latent defects, the efficiency increases.

This contribution can also be applied to AEC DPAT and NNR as shown in Section 4.8.

4.4.4 R DPAT

A variation of DPAT called R DPAT (Robust DPAT) has been developed and implemented to increase the efficiency and effectiveness of the DPAT algorithm. By using robust statistics (Grubbs algorithm), possible outliers do not bias the dispersion estimation. And by transforming non-normal data to normal, skewness is factored in (better than the AEC DPAT method).

Additionally, as in the DPAT ST algorithm (DPAT with specific tests), engineers can select which tests will be used with robust DPAT, further increasing the performance of this new algorithm.

Robust DPAT is described in figure 4.7. The main steps are:

- i. First, if the distribution has less than 8 categories, an interpolation is performed to increase the granularity and raise the chances of passing the normality test and the Johnson transformation [11] being successful (if needed). This value of 8 has been determined experimentally.
- ii. The next step is to test the distribution for normality (applying the Anderson-Darling test [2]).
- iii. If the distribution is normal, Grubbs algorithm [21] is applied which recursively removes outliers and the algorithm ends. Grubbs is only valid if the underlying distribution is normal.
- iv. If the distribution is not normal, Grubbs is still applied to eliminate outliers. If the resulting distribution without the outliers is normal, then Grubbs was rightfully applied since the underlying population is truly normal and the process ends.
- v. If the resulting distribution without the outliers is not normal, Grubbs is discarded and a Johnson transformation is applied. The Johnson transformation is computationally intensive.
- vi. If the transformed distribution is normal. Grubbs is applied and the algorithm ends
- vii. If the transformed distribution is not normal, the AEC DPAT version (described in section 4.3.3) is applied to the original data instead.

The Anderson-Darling normality test, Grubbs algorithm and Johnson transformation are described in the following subsections:

4.4.4.1 Anderson-Darling normality test

The Anderson-Darling normality test [2] is used as part of the robust DPAT algorithm. This test applies the cumulative density function to sorted

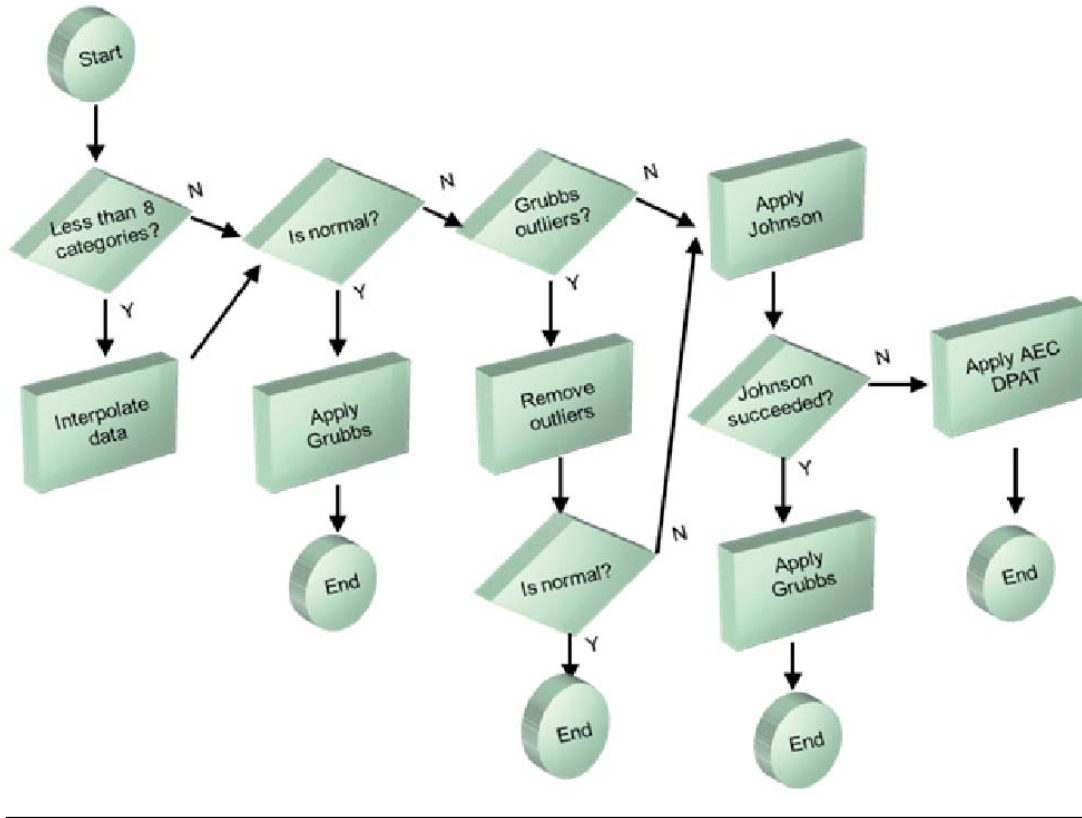


Figure 4.7: Robust DPAT algorithm

values with y_1 being the lowest, n the sample size, and F being the theoretical cumulative value of the normal distribution, as shown in equation 4.4.

$$A^2 = -n - \frac{1}{n} \sum_{i=1}^n (2i - 1) (\ln F(y_i) + \ln(1 - F(y_{n+1-i}))) \quad (4.4)$$

The Anderson-Darling Test statistic A^2 [2] is adjusted for low sample sizes and the p value can be approximated using equation 4.5. If the p

value is less than 0.05, the normality hypothesis is rejected.

$$\hat{A}^2 = A^2 \left(1 + \frac{0.75}{n} + \frac{2.25}{n^2}\right)$$

$$\begin{aligned} [\hat{A}^2 < 0.2] & \quad p = 1 - e^{(-13.463+101.14\hat{A}^2-223.73(\hat{A}^2)^2)} \\ [0.2 \leq \hat{A}^2 < 0.34] & \quad p = 1 - e^{(-8.318+42.796\hat{A}^2-59.938(\hat{A}^2)^2)} \\ [0.34 \leq \hat{A}^2 < 0.6] & \quad p = e^{(0.9177-4.279\hat{A}^2+1.38(\hat{A}^2)^2)} \\ [0.6 \leq \hat{A}^2 < 13] & \quad p = e^{(1.2937-5.709\hat{A}^2+0.0186(\hat{A}^2)^2)} \\ [13 \leq \hat{A}^2] & \quad p = 0 \end{aligned} \quad (4.5)$$

4.4.4.2 Grubbs algorithm

The robust DPAT algorithm uses the Grubbs method [21] to identify outliers but is only applicable if the underlying distribution follows a normal distribution. Grubbs does not use the standard deviation to identify outliers. It is instead an incremental algorithm that uses a distance from the average to detect outliers. The steps for the Grubbs algorithm as defined in [21] are:

- i. Compute the G_{critical} value as shown on equation 4.6 where N is the data sample size, t is the inverse Student's t cumulative distribution function and α is level of significance or type I error which is set at 0.05.

$$G_{\text{critical}} = \frac{N-1}{\sqrt{N}} \sqrt{\frac{t^2((\alpha/2N), N-2)}{N-2 + t^2((\alpha/2N), N-2)}} \quad (4.6)$$

- ii. Loop through all the values in the sample and compute their G value as (value - mean)/standard deviation and get the maximum G value.
- iii. If the maximum G value is greater than G_{critical} , mark that value as an outlier, remove it from the sample, go back to the first step to recalculate the G_{critical} value and continue removing outliers.
- iv. If the maximum G value is not greater than G_{critical} , end the process.
- v. Finally, Grubbs [21] only removes outliers temporarily so that a robust mean and a robust standard deviation can be computed from the data values without outliers. Then, equation 4.7 is applied to compute the upper and lower limits where k is the sigma multiplier given for this

algorithm. Data values outside these limits will be marked as outliers. By specifying the sigma multiplier (k), the algorithm can be tuned to be more or less aggressive.

$$\begin{aligned} \text{Upper Limit} &= \text{robust mean} + k \times \text{robust std deviation} \\ \text{Lower Limit} &= \text{robust mean} - k \times \text{robust std deviation} \end{aligned} \quad (4.7)$$

4.4.4.3 Johnson transformation

The Johnson transformation [11] attempts to transform a non-normal data sample to normality. As described in [11], the first step is to generate 200 transformations and test them for normality. The one that yields the best results is chosen.

First, for each $z \in \{0.2, 0.21, 0.22, \dots, 1.2\}$, the quantile ratio QR is calculated according to equation 4.8 where x_i is the q_i^{th} quantile for $i = 1, 2, 3, 4$, and q_1, q_2, q_3, q_4 are the areas on a standard normal curve below $-3z, -z, z$, and $3z$. Thus $q_1 = \Phi(-3z)$, $q_2 = \Phi(-z)$, $q_3 = \Phi(z)$, $q_4 = \Phi(3z)$, where Φ is the distribution function of a standard normal variable.

$$\text{QR} = \frac{(x_4 - x_3)(x_2 - x_1)}{(x_3 - x_2)^2} \quad (4.8)$$

If $\text{QR} < 1$ data values are transformed using functions S_L and S_B otherwise functions S_U and S_L are used. Functions S_L, S_B, S_U are given in equation 4.9. Parameters η, γ, λ , and ε are estimated with equation 4.10. These parameters must meet the conditions specified in equation 4.9 or the transformation for that z value is discarded.

Johnson Family	Transformation	Parameter Conditions	X Condition
S_B bound	$Z = \gamma + \eta \ln\left(\frac{X - \varepsilon}{\lambda + \varepsilon - X}\right)$	$\eta, \lambda > 0, -\infty < \gamma < \infty,$ $-\infty < \varepsilon < \infty$	$\varepsilon < X < \varepsilon + \lambda$
S_L lower bound or lognormal	$Z = \gamma + \eta \ln(X - \varepsilon)$	$\eta > 0, -\infty < \gamma < \infty,$ $-\infty < \varepsilon < \infty$	$X > \varepsilon$
S_U unbound	$Z = \gamma + \eta \sinh^{-1}\left(\frac{X - \varepsilon}{\lambda}\right)$	$\eta, \lambda > 0, -\infty < \gamma < \infty,$ $-\infty < \varepsilon < \infty$	$-\infty < X < \infty$

(4.9)

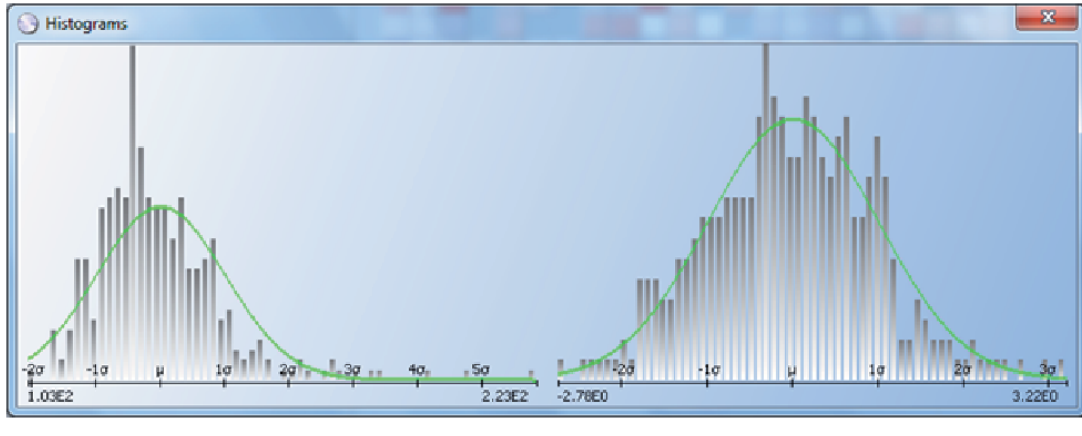


Figure 4.8: Lognormal to normal distribution transformation

S_B bound	S_L lower bound or lognormal	S_U unbound	
$\hat{\eta} = \frac{z}{\cosh^{-1}\left[\frac{1}{2}\left[\left(1 + \hat{x}_M/\hat{x}_U\right)\left(1 + \hat{x}_M/\hat{x}_L\right)\right]^{1/2}\right]}$	$\hat{\eta} = \frac{2z}{\ln(\hat{x}_U/\hat{x}_M)}$	$\hat{\eta} = \frac{2z}{\cosh^{-1}\left[\frac{1}{2}\left(\hat{x}_U/\hat{x}_M + \hat{x}_L/\hat{x}_M\right)\right]}$	(4.10)
$\hat{\rho} = \hat{\eta} \cdot \sinh^{-1}\left[\frac{(\hat{x}_M/\hat{x}_L - \hat{x}_M/\hat{x}_U) \cdot \left[\left(1 + \hat{x}_M/\hat{x}_U\right)\left(1 + \hat{x}_M/\hat{x}_L\right) - 4\right]^{1/2}}{2(\hat{x}_M^2/(\hat{x}_L\hat{x}_U) - 1)}\right]$	$\hat{\rho} = \hat{\eta} \cdot \ln\left[\frac{\hat{x}_U/\hat{x}_M - 1}{(\hat{x}_U\hat{x}_M)^{1/2}}\right]$	$\hat{\rho} = \hat{\eta} \sinh^{-1}\left[\frac{(\hat{x}_L/\hat{x}_M - \hat{x}_U/\hat{x}_M)}{2(\hat{x}_L\hat{x}_U/\hat{x}_M^2 - 1)^{1/2}}\right]$	
$\hat{\lambda} = \frac{\hat{x}_M \left[\left(1 + \hat{x}_M/\hat{x}_U\right) \cdot \left(1 + \hat{x}_M/\hat{x}_L\right) - 2 \right]^2 - 4}{(\hat{x}_M^2/\hat{x}_L\hat{x}_U - 1)^{1/2}}$	$\hat{\lambda} = \frac{1}{2} \left[\hat{x}_L + \hat{x}_U + \hat{x}_M \left(\frac{\hat{x}_U/\hat{x}_M + 1}{\hat{x}_U/\hat{x}_M - 1} \right) \right]$	$\hat{\lambda} = \frac{2\hat{x}_M (\hat{x}_L/\hat{x}_U - \hat{x}_U/\hat{x}_M)^2}{(\hat{x}_U/\hat{x}_M + \hat{x}_L/\hat{x}_M - 2)(\hat{x}_U/\hat{x}_M + \hat{x}_L/\hat{x}_M + 2)^{1/2}}$	
$\hat{\epsilon} = \frac{1}{2} \left(\hat{x}_L + \hat{x}_U - \hat{\lambda} + \frac{\hat{x}_M (\hat{x}_M/\hat{x}_L - \hat{x}_M/\hat{x}_U)}{(\hat{x}_M^2/(\hat{x}_L\hat{x}_U) - 1)} \right)$	$\hat{\epsilon} = \frac{1}{2} \left(\hat{x}_L + \hat{x}_U + \hat{x}_M \left(\frac{\hat{x}_U/\hat{x}_M + 1}{\hat{x}_U/\hat{x}_M - 1} \right) \right)$	$\hat{\epsilon} = \frac{1}{2} \left(\hat{x}_L + \hat{x}_U + \frac{\hat{x}_M (\hat{x}_L/\hat{x}_M - \hat{x}_U/\hat{x}_M)}{(\hat{x}_U/\hat{x}_M + \hat{x}_L/\hat{x}_M - 2)} \right)$	

Examples of Johnson transformations are shown on figures 4.8, and 4.9. Figure 4.8 shows a lognormal distribution (left) transformed to normal (right). Exponential distributions can also be transformed to normal as shown on figure 4.9. The data sample on the left follows an exponential distribution and it is transformed to normal on the right. In cases like these, R DPAT algorithm is more efficient than the classic and AEC DPAT versions.

It should be noticed that the robust DPAT algorithm will also interpolate data results with low granularity. This will increase the chances of the Anderson-Darling normality test and Johnson transformation succeeding.

4.4.5 Multi-site test results

Some devices are probed with probe cards that have multiple probe heads (or sites): 1, 2, 4, 8, up to 16. In some cases the probe heads are not cali-

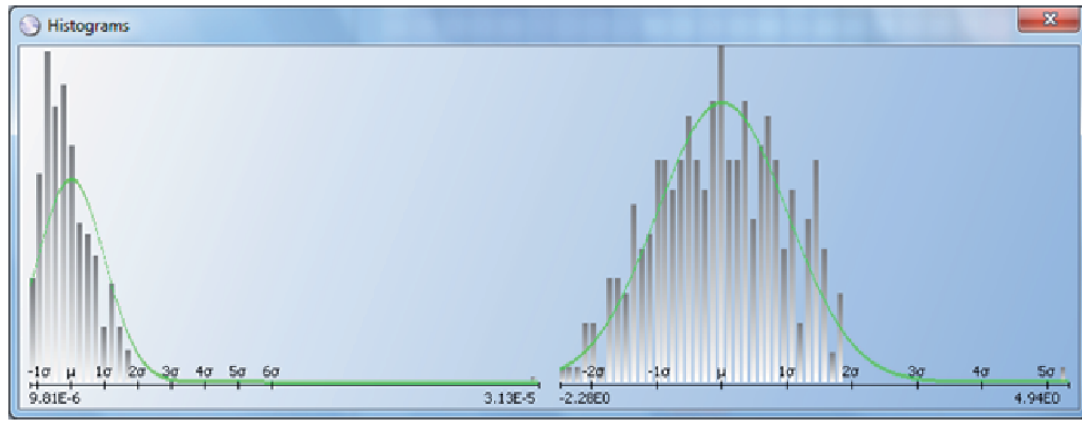


Figure 4.9: *Exponential to normal distribution transformation*

brated correctly and record results that are shifted with respect to one another. If all the results for a given wafer are included in the same histogram, the distribution will appear multi-modal and the classic DPAT algorithm will be less effective and efficient.

The charts on figure 4.10 show such case. To avoid this problem, a multi-site option is added to all the different DPAT versions (DPAT, AEC DPAT, and robust DPAT). If this multi site option is selected, test results will be separated by site and will be treated separately before applying the desired DPAT version.

The histogram at the top on figure 4.10 includes all the test results which is clearly bimodal. The histogram at the center and bottom are the test results divided by site.

4.4.6 Test aggregation

By default, each parametric test is processed independently by the DPAT algorithm and each one will be used to identify outliers separately. If this aggregation option is selected, instead of applying DPAT to each individual test, a new value is calculated for each die (equation 4.11) where α_i is this new value for die i , X_{ti} is the test result for test t and die i , μ_t is the mean for test t , σ_t is the standard deviation for test t , and T is the number of tests. Then, DPAT is applied to these new calculated values. The rationale behind this method is to decrease the influence of each test in favor of a

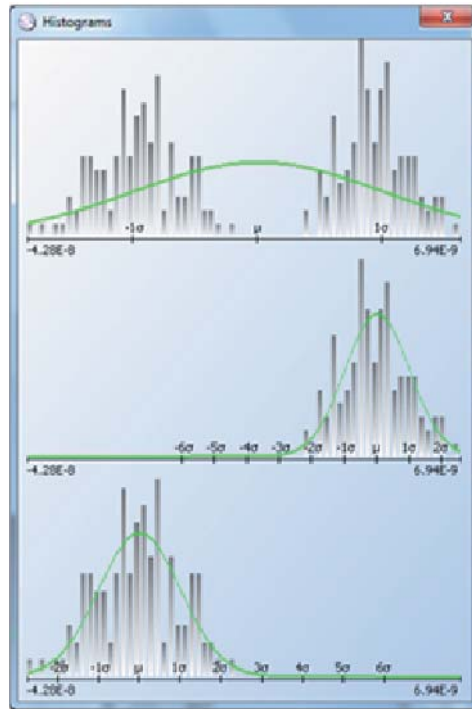


Figure 4.10: Multi-site example

combined measurement.

$$\alpha_i = \frac{\sum_{t \geq 1}^T \left| \frac{X_{ti} - \mu_t}{\sigma_t} \right|}{T} \quad (4.11)$$

4.5 EWM functionality

EWM collects wafer maps with visual inspection defects, probe electrical pass/fail results and parametric test results. Statistical outlier detection algorithms are applied to wafer maps to identify abnormal dice that have a high probability of failing. EWM finally combines all the defects (visual, electrical and statistical outliers) into a single wafer map with good and bad dice for assembly. Figure 4.11 shows these data flows. EWM uses the industry standard SEMI E142 [75] for wafer map data transfer with external probe and internal and external assembly. This SEMI E142 standard [75] was developed as part of this research in collaboration with other semiconductor industries as described in section 4.6.

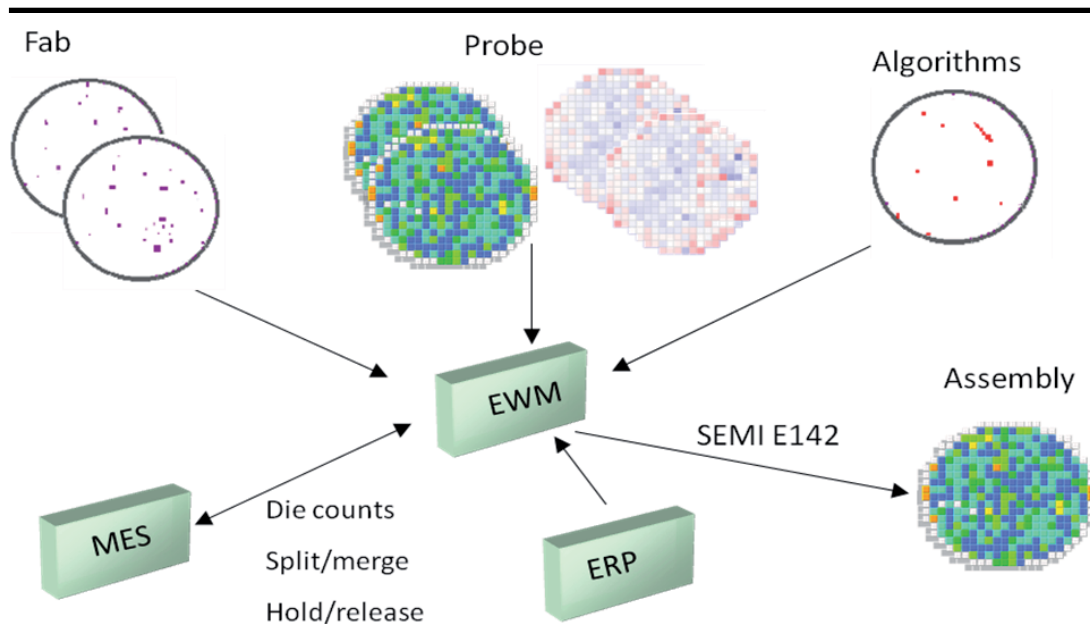


Figure 4.11: EWM data flows

EWM also integrates with the Manufacturing Execution System (MES) for automatic die count updates and lot splits/merges and hold/release operations and with the Enterprise Resource Planning system (ERP) for automatic wafer map transfer to assembly.

EWM started 8 years ago with very basic functionality. Multiple releases have been produced over these years to incrementally add more functionality. This gradual introduction of new functionality avoids including many errors in one release while providing value as soon as it is developed.

Currently, EWM runs on every probe floor in Freescale in Asia, Europe and North America (figure 4.12).

The inputs, outputs, interfaces with other key applications and internal processing are described in detail below and depicted in figure 4.13.

4.5.1 Inputs

- Unit probe electrical test results are generated in two file formats: Integrator Nested Format (INF) and Standard Test Data Format (STDF). These files are generated during unit probe and are sent to EWM. INF

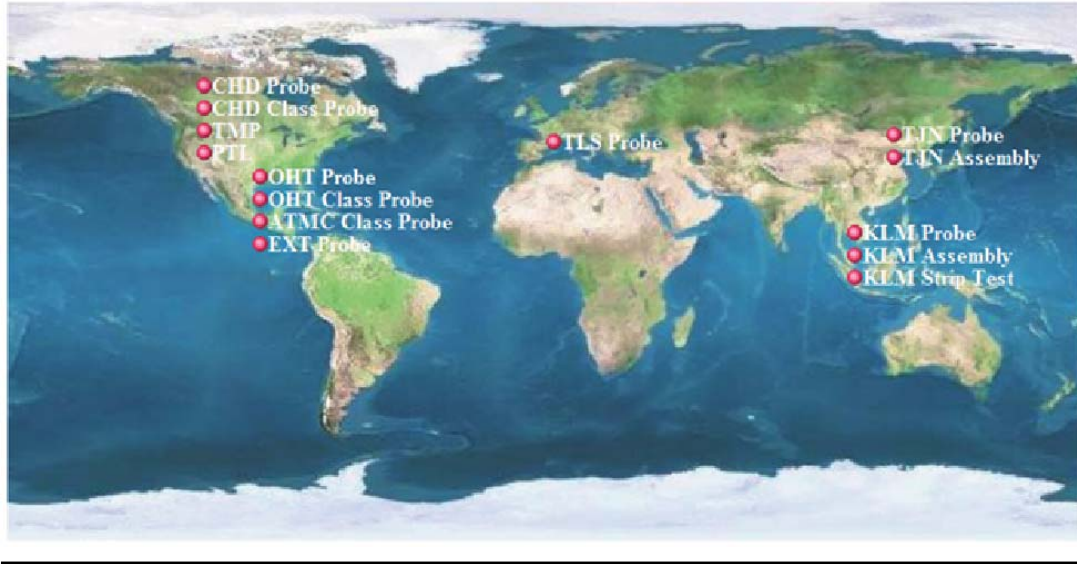


Figure 4.12: *EWM deployment status*

files contain hard bin results. They represent the output of the test program (pass or fail) for each die. A number from 1 to 255 indicates if the die passed all tests or if it failed and the reason. Typically 1 means pass and 2 through 255 denote failing dice. The failing reason is specific to each device and the bin number. STDF files contain parametric tests results. Each test program performs hundreds of tests and the electrical output for each test and each die is recorded in the STDF file.

- External probe subcontractors send SEMI E142 [75] files with hard bin results and STDF files with parametric tests.
- EWM also accepts wafer maps with fab defects from automated visual inspection tools. Supported file formats are KLARFF and SEMI E142 [75]. These two formats contain a wafer map with the dice that have failed the visual inspection.
- Additionally, operators can edit maps in EWM and mark defective dice after performing a manual visual inspection.

4.5.2 Processes

- When EWM receives wafer maps with hard bins, parametric tests or visual defects, it stores them in a database and applies zero, one or more

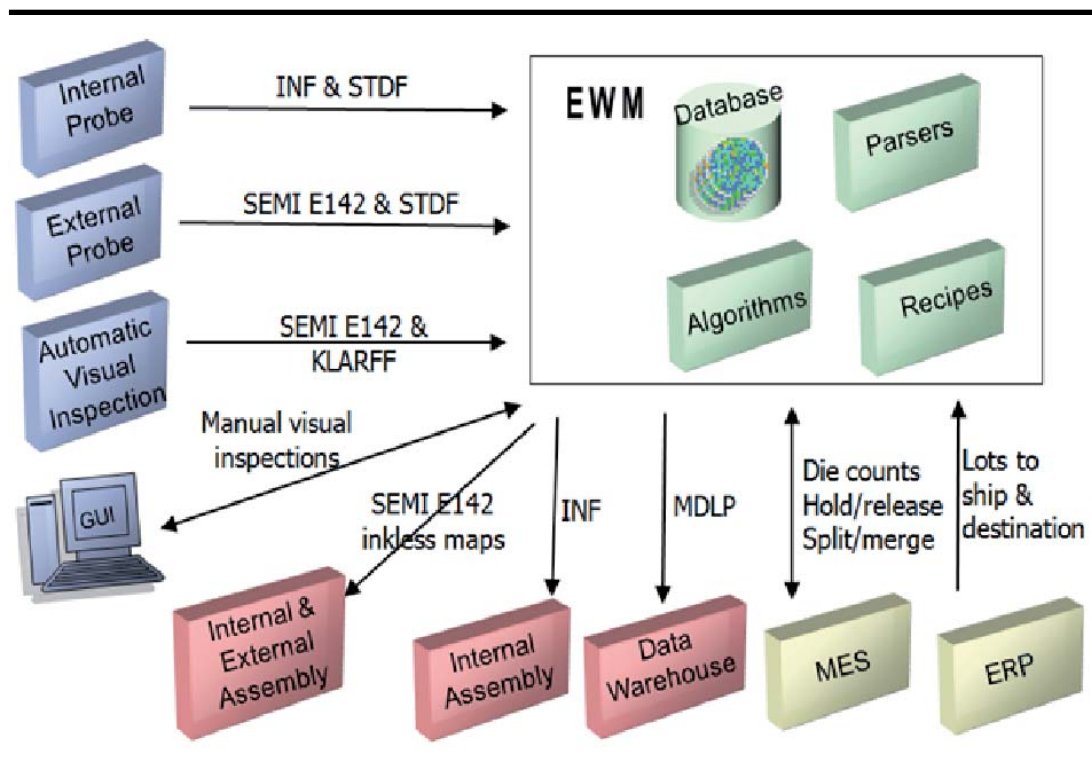


Figure 4.13: EWM system context diagram

algorithms (described in sections 4.3 and 4.4) to identify additional bad dice according to the recipe that each device has configured. As result of some of these algorithms, lots may be automatically put on hold (or released) in the MES.

- When a lot finishes probe, all the wafer maps for each wafer in the lot are merged, adding all the defects: visual, electrical and algorithmic. If any map or any algorithm has identified a die as defective, the final composite will mark that die as defective. In other words, defects are additive.

4.5.3 Outputs

- When a lot moves from probe to assembly, EWM generates a final wafer map with the dice that have passed all tests and can be picked in assembly. EWM supports ink and inkless assembly. For inkless assembly, an

electronic SEMI E142 file [75] is generated. This format is used for internal assembly and external assembly subcontractors. For inking, an INF file is generated instead.

- When EWM sends electronic wafer maps for a lot to assembly, a copy is also sent to the Data Warehousing System where engineers will analyze yield trends.

4.5.4 Interfaces

- EWM interfaces with the MES to put lots on hold or release them if needed when SBL and BMY algorithms are executed.
- Operators split and merge lots through the EWM GUI and they are automatically split and merged in the MES.
- Additionally, EWM automatically updates die counts in the MES. Also, the MES triggers EWM to generate INF files for inking (assembly). Finally, the Enterprise Resource Planning system (ERP) sends a file several times a day to EWM to indicate where to send inkless maps for assembly.

4.6 SEMI E142 specification for substrate mapping

As part of this research, the opportunity arose to participate in the elaboration of the SEMI E142 industry specification [75] in 2006 in collaboration with other semiconductor industries. Freescale has been one of the first companies to implement this standard.

The purpose of the SEMI E142 standard [75] is to define the data items that are required to report, store and transmit map data for substrates such as wafers, frames, strips, and arrays.

An example of the final composite map for a wafer generated by EWM is listed on figure 4.14. A data file containing that information is sent to assembly to be used during the pick and place operation to grab devices that have passed all tests.

The SEMI E142 specification [75] has also been used as a guide for implementing the database schema that holds the wafer map information in the database.

The implementation of this standard has decreased development time considerably by allowing a common wafer map data format to external probe and assembly subcontractors. Without this standard, a dozen different parsers would have been implemented and tested to interface with these subcontractors, unnecessarily supporting multiple wafer map data formats.

4.7 User interface

The EWM Graphical User Interface (GUI) is critical to probe operation, allowing probe and device engineers to review wafer maps to identify defects and patterns as well as the effects of outlier detection algorithms. The following sections describe some of the main screens.

4.7.1 Hierarchical view

The data stored in EWM is displayed in a hierarchical manner, in a tree-like structure (figure 4.15). The top node in the tree contains all the devices. Each device expands to show all the lots that belong to the lot. Each lot further expands into several wafers and finally, each wafer lists all the wafer maps generated in the fab and probe floors for that wafer, as well as results of outlier detection algorithms.

4.7.2 Device recipes (inspections)

Each device has a recipe (also called inspection) associated that determines which maps are required, which algorithms will be applied and the settings for those algorithms. Figure 4.16 shows an example of a recipe.

4.7.3 Lot screens

A lot report can be generated anytime during the probe process to follow the evolution of each lot and to analyze problems with them.

This lot report contains several pages. One of the pages shows the yield for each wafer on the lot. Another page shows a view with all the wafer maps where the geographical distribution of defects can be quickly identified (figure 4.17). Green represents passing dice while other colors represent different defects.

Finally, another page on the lot report shows a combined view of the yield of all the wafers in the lot (figure 4.18). This is also known as heat map where green signifies no errors for the given die location across all wafers for the given lot. The darker the red color, the lower the yield for that location.

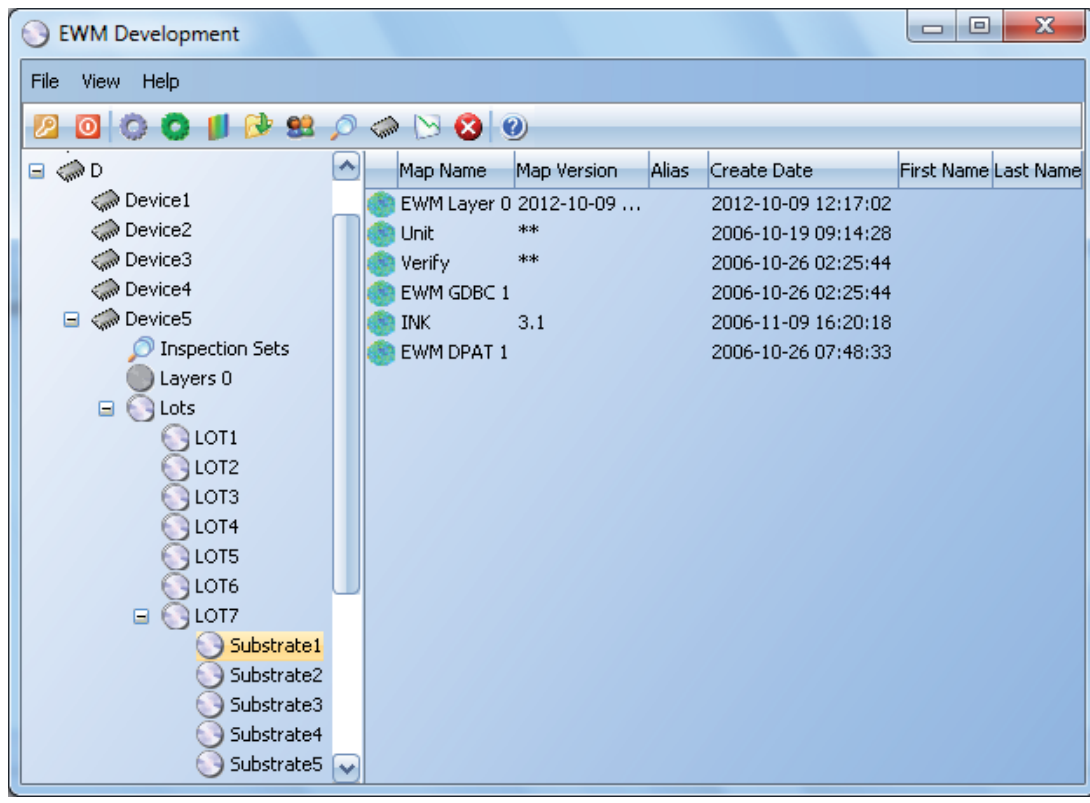


Figure 4.15: Hierarchical view

4.7.4 Wafer screens

Two and three-dimensional views are available for each wafer map (figures 4.19 and 4.20). Individual layers can be viewed or a combination of multiple layers can be generated.

Wafer maps with hard bin results (values between 1 and 255 for each die) normally use green to represent passing dice (figure 4.19).

Wafer maps with parametric test results (real numbers) use a gradient from dark blue to dark red to show each value in relation to the mean of the test results for that wafer. The further the value is to the left of the mean, the darker the blue color, the further to the right, the darker the red color (figure 4.20).

Type	Map Name	Alias	Algorithm	Parameters	Editable	Role	Ship To	Include Previous Layers	Group Failures	Comments
Optional	Force Ink				Y			Y	N	
Required	Sort1	Pass1			N			N	N	
Engineering	STDF S1	PRB	DPAT	...	N			N	N	
Required	EWM DPAT S1		SBL	...	N			N	N	
Not Used	EWM SBL 5				N			N	N	
Required	Sort2	Pass2			N			N	N	
Engineering	STDF S2	PRA2	DPAT	...	N			N	N	
Required	EWM DPAT S2		SBL	...	N			N	N	
Not Used	EWM SBL 6				N			N	N	
Required	Sort3	Pass3	GDBC	...	N			N	N	
Optional	EWM GDBC 1		GDBC	...	N			N	N	
Engineering	LOGIC STDF	PRA	DPAT	...	N			N	N	
Required	EWM DPAT S3		SBL	...	N			N	N	
Not Used	EWM SBL 4				N			N	N	

Figure 4.16: Device recipe (inspections)

4.7.5 Manual visual inspections

Probe operators sometimes visually inspect wafers for defects. These defects are entered manually in EWM.

4.7.6 Reports

EWM provides several reports for device engineers to analyze the yield impact incurred by the statistical outlier detection algorithms and to tune these algorithms. These reports are:

- STDF report. This report can be run on any STDF file loaded into EWM with parametric test results. Each test is identified by a test number and a test name. For each test the report lists the test number, test name, the number of different values or categories, the minimum value, maximum, mean, standard deviation, and whether the values could be transformed to normality or not. These tests are used for the DPAT algorithm. Figure 4.21 shows an example of this report.
- Algorithm reports. There are multiple algorithm reports that show yield losses due to the outlier detection algorithms by different categories.

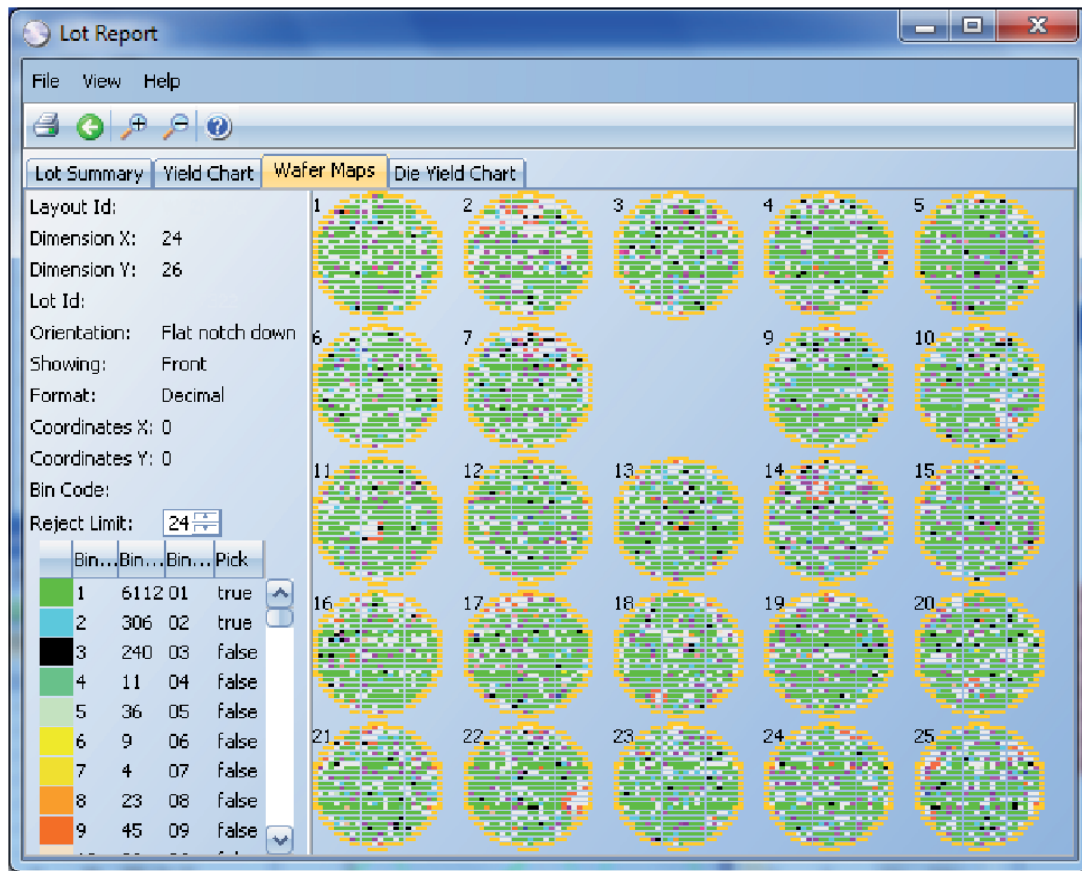


Figure 4.17: Lot report. Wafer maps view

One of these reports is the DPAT report by wafer. This report shows the percentage of dice marked by DPAT by wafer (figure 4.22). Each line represents a wafer and contains the device, lot, wafer ID, bin number for the DPAT defect, number of dice marked as defective by DPAT, the total die count on the wafer, the percentage of dice marked by DPAT over the total die count, and the date when the DPAT algorithm was executed. Another of these reports is the DPAT report by test. This report lists all the tests and their individual yield impact in DPAT (figure 4.23). There is a line for each wafer and test. Each line has the device, lot, wafer ID, test number and name, the sigma multiplier (k) for that test, the number of dice marked by DPAT for that test, the total die count on the wafer, the percentage of dice marked by DPAT over the total die count, and the date when the DPAT algorithm was executed.

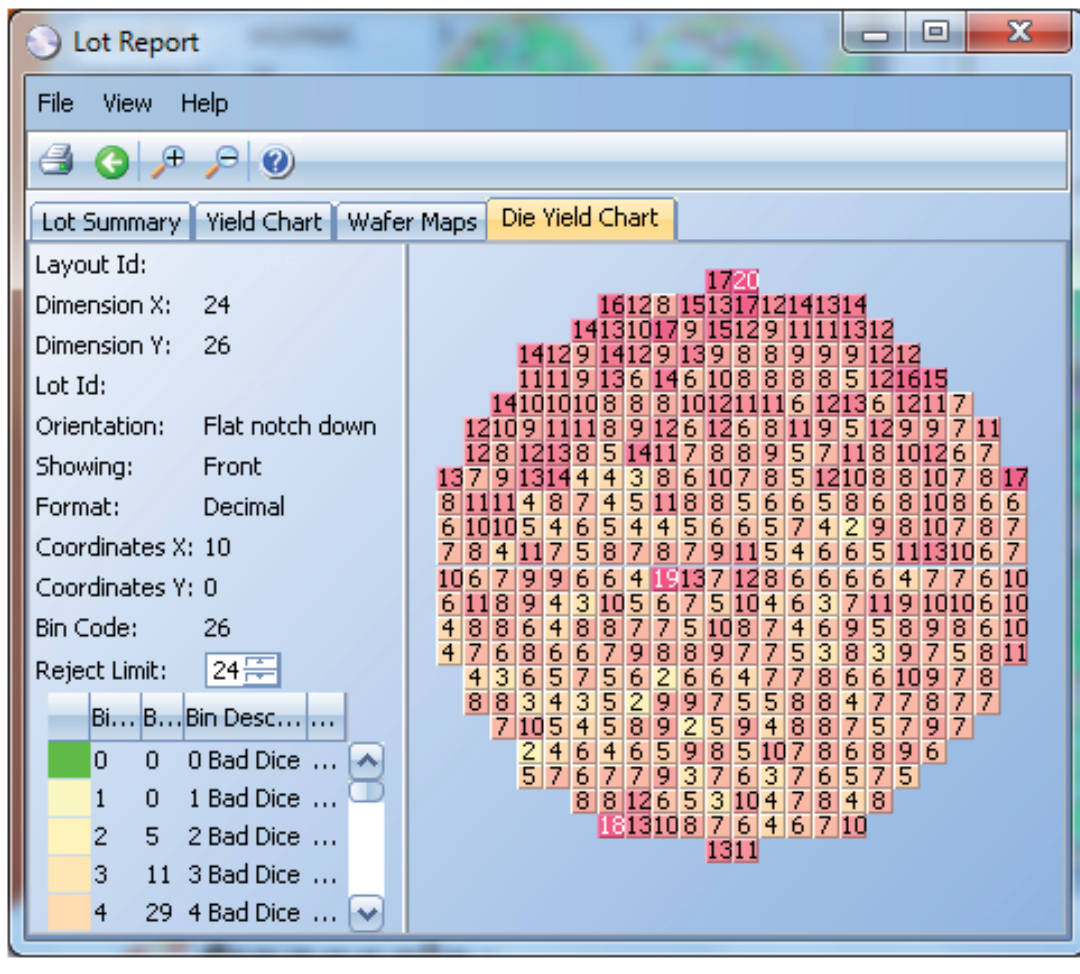


Figure 4.18: Lot report. Yield map

- CQI report. This report helps tuning the DPAT and NNR algorithms by highlighting tests that have a high correlation to a given CQI. Engineers enter the coordinates of the CQI die. The report shows the Z values off the median for that die across all tests (equation 4.12). The robust sigma calculated in this equation is a simple estimation of dispersion, avoiding possible outliers on both ends.

$$Z = \frac{X - p50}{rSigma} \quad (4.12)$$

where $rSigma = \frac{p75 - p25}{1.35}$, X is the test result for the die being analyzed

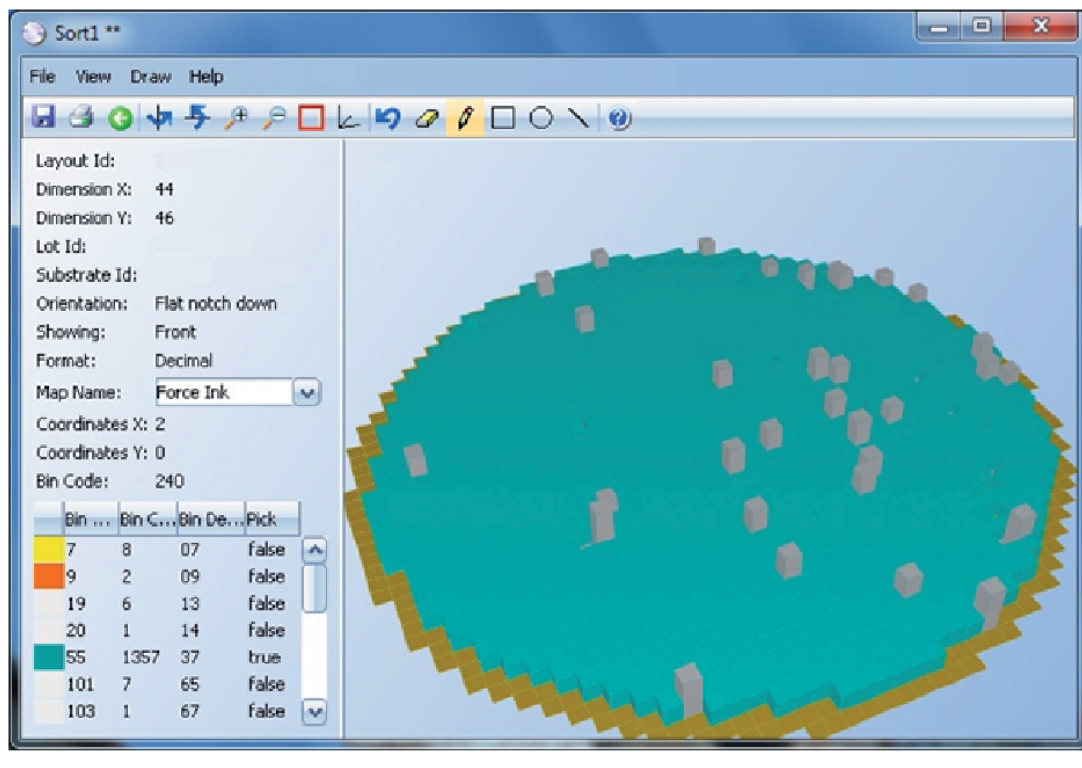


Figure 4.19: *Hard bins wafer view*

- and p25, p50, and p75 are the 25th, 50th, and 75th percentiles for the entire wafer. The Z value is used to identify the parametric tests that have a higher correlation to the CQI. This information is used to set up the DPAT ST algorithm described in section 4.4. Figure 4.24 displays an example of this report. The panel on the left lists all the tests by increasing Z values. The higher the Z value, the stronger the correlation of that test with the CQI. The user can click on any test and the corresponding values will be displayed on the right with a cross marking the location of the CQI. The histogram for the test results is shown at the bottom of the screen.
- **Test Correlation Report.** This report provides a correlation between the hard bin results on a wafer map and all the parametric tests for a given STDF file. The purpose is to provide a wafer map with bin results from final test or with CQIs to identify which tests have a higher correlation with those post probe defects. This report is used for the DPAT ST algo-

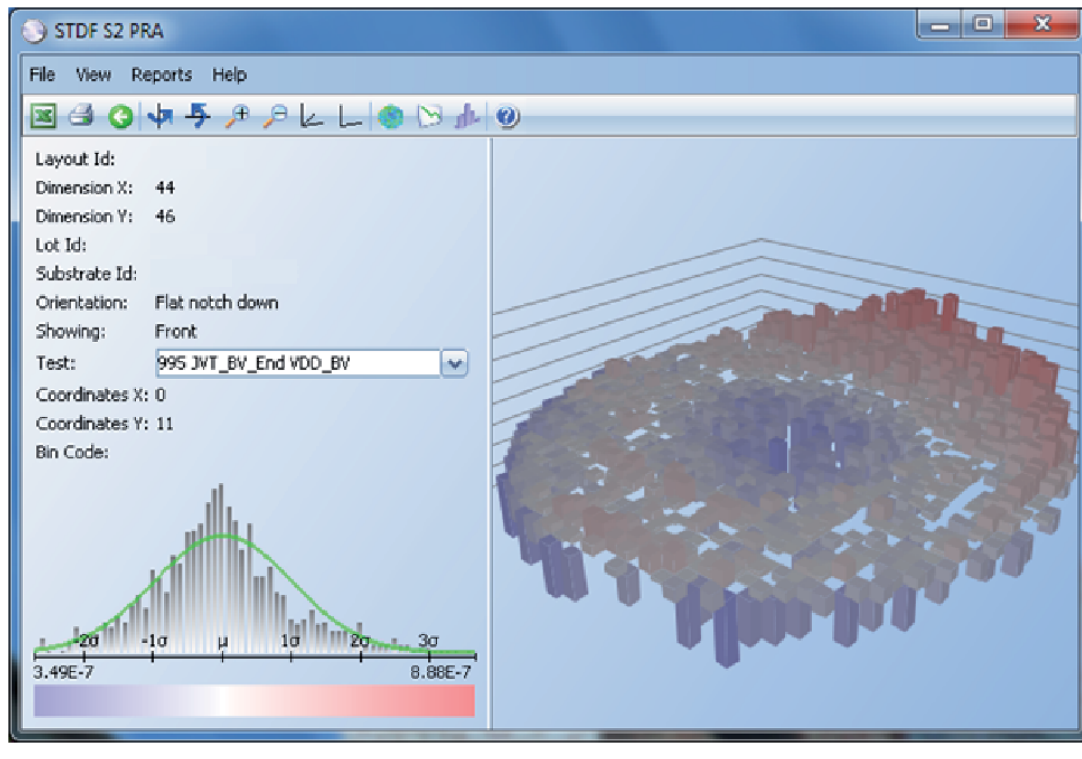


Figure 4.20: Parametric test results wafer view

rithm described in section 4.4. Tests with a strong correlation to past CQIs will be used in a more aggressive way (smaller sigma multiplier k), increasing the chances of filtering future defective material as described in section 4.8.

4.8 Results

This section presents the results obtained with EWM. First, EWM is a holistic, extensive, and automated approach to defect detection. It combines fab visual inspections, probe electrical results and statistical outliers. Before EWM, only probe results were consistently used. Just a few devices included visual inspections and a very small percentage would run some outlier detection algorithms. In addition, it was performed in a heterogeneous way, with small, ad hoc applications and with mostly manual processes.

As mentioned in the previous sections, EWM implements the following algorithms: SBL, BMY, GDBC, SPAT, DPAT, AEC DPAT, NNR, and six new algorithms that can be easily set up and tailored for the needs of each device.

Test Number	Test Name	Duplicate	Categories	Min	Max	Mean	Std Dev	Transformation
995	JVT_BV_End VDD_BV	N	996	3.49E-7	8.88E-7	5.79E-7	8.46E-8	
996	JVT_HV_End VDD_HV_Main	N	993	2.78E-7	6.04E-6	1.04E-6	1.09E-6	Could not convert to normal ...
998	JVT_BV_Start VDD_BV	N	1323	5.21E-7	8.36E-5	1.1E-6	5.45E-6	Could not convert to normal ...
999	JVT_HV_Start VDD_HV_Main	N	1343	3.85E-7	2.93E-5	1.32E-6	1.56E-6	Could not convert to normal ...
1600	Static_Idd VDD_HV_Main	N	1321	0.00104	0.00766	0.00612	2.13E-4	
1601	Static_Idd VDD_BV	N	1331	0.00715	0.0867	0.00795	0.00233	
7000	Vreg_Gross Vreg_Load	N	1244	1.3	1.34	1.32	0.00711	
8300	IDDQ_FSL_Pre_HVST VDD_BV	N	1108	4.31E-5	8.35E-4	6.13E-5	3.34E-5	Could not convert to normal ...
8301	IDDQ_FSL_Pre_HVST VDD_BV	N	1105	4.22E-5	1.1E-4	5.84E-5	4.9E-6	
8302	IDDQ_FSL_Pre_HVST VDD_BV	N	1101	4.18E-5	1.1E-4	5.79E-5	4.7E-6	
8303	IDDQ_FSL_Pre_HVST VDD_BV	N	1108	4.17E-5	1.09E-4	5.79E-5	4.71E-6	
8304	IDDQ_FSL_Pre_HVST VDD_BV	N	1104	4.15E-5	1.08E-4	5.74E-5	4.83E-6	
8305	IDDQ_FSL_Pre_HVST VDD_BV	N	1106	4.14E-5	1.08E-4	5.76E-5	4.72E-6	Could not convert to normal
8306	IDDQ_FSL_Pre_HVST VDD_BV	N	1100	4.12E-5	1.08E-4	5.74E-5	4.72E-6	
8307	IDDQ_FSL_Pre_HVST VDD_BV	N	1103	4.09E-5	1.07E-4	5.69E-5	4.7E-6	
8308	IDDQ_FSL_Pre_HVST VDD_BV	N	1099	4.1E-5	1.08E-4	5.72E-5	4.72E-6	
8309	IDDQ_FSL_Pre_HVST VDD_BV	N	1104	4.08E-5	1.07E-4	5.67E-5	4.83E-6	
8310	IDDQ_FSL_Pre_HVST VDD_BV	N	1103	4.08E-5	1.07E-4	5.7E-5	4.72E-6	
8311	IDDQ_FSL_Pre_HVST VDD_BV	N	1102	4.1E-5	1.08E-4	5.7E-5	4.7E-6	
8312	IDDQ_FSL_Pre_HVST VDD_BV	N	1096	4.04E-5	1.07E-4	5.66E-5	4.91E-6	
8313	IDDQ_FSL_Pre_HVST VDD_BV	N	1106	4.09E-5	1.07E-4	5.69E-5	4.7E-6	
8314	IDDQ_FSL_Pre_HVST VDD_BV	N	1103	4.06E-5	1.07E-4	5.68E-5	4.86E-6	
8315	IDDQ_FSL_Pre_HVST VDD_BV	N	1104	4.05E-5	1.07E-4	5.64E-5	4.74E-6	
8316	IDDQ_FSL_Pre_HVST VDD_BV	N	1108	4.05E-5	1.07E-4	5.67E-5	4.73E-6	
8317	IDDQ_FSL_Pre_HVST VDD_BV	N	1101	4.06E-5	1.07E-4	5.67E-5	4.53E-6	
8318	IDDQ FSL Pre HVST VDD BV	N	1105	4.06E-5	1.07E-4	5.67E-5	4.62E-6	

Figure 4.21: STDF report

Layouts	Lots	Substrate	Bin Code	Count	Total Count	Percentage	Limit	Create Date
<input checked="" type="checkbox"/>	Device	Lot	Wafer1	245	3	402	0.75	2012-11-02 22:49:40
<input checked="" type="checkbox"/>	Device	Lot	Wafer11	245	0	402	0.0	2012-11-02 22:49:42
<input checked="" type="checkbox"/>	Device	Lot	Wafer13	245	1	402	0.25	2012-11-02 22:49:42
<input checked="" type="checkbox"/>	Device	Lot	Wafer14	245	4	402	1.0	2012-11-02 22:49:43
<input checked="" type="checkbox"/>	Device	Lot	Wafer17	245	1	402	0.25	2012-11-02 22:49:43
<input checked="" type="checkbox"/>	Device	Lot	Wafer20	245	1	402	0.25	2012-11-02 22:49:44
<input checked="" type="checkbox"/>	Device	Lot	Wafer21	245	0	402	0.0	2012-11-02 22:49:45
<input checked="" type="checkbox"/>	Device	Lot	Wafer25	245	0	402	0.0	2012-11-02 22:49:46
<input checked="" type="checkbox"/>	Device	Lot	Wafer4	245	2	402	0.5	2012-11-02 22:49:41

Figure 4.22: DPAT yield impact by wafer

Layouts	Lots	Substrate	Test Number	Test Name	Sigma	Count	Total...	Percentage	Create Date
Device	Lot	Wafer14	31820	S3 MAX EXT SHIFT max...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer17	31830	S3 MAX ARRAY INT SHI...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer17	31820	S3 MAX EXT SHIFT max...	6.0	1	402	0.25	2012-11-02...
Device	Lot	Wafer20	31830	S3 MAX ARRAY INT SHI...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer20	31820	S3 MAX EXT SHIFT max...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer21	31830	S3 MAX ARRAY INT SHI...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer21	31820	S3 MAX EXT SHIFT max...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer25	31830	S3 MAX ARRAY INT SHI...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer25	31820	S3 MAX EXT SHIFT max...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer4	31830	S3 MAX ARRAY INT SHI...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer4	31820	S3 MAX EXT SHIFT max...	6.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer1	8112	scan_IDDQ_VDD_20pts...	9.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer1	8104	scan_IDDQ_VDD_20pts...	9.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer1	8118	scan_IDDQ_VDD_20pts...	9.0	0	402	0.0	2012-11-02...
Device	Lot	Wafer1	8200	iddq_range_test vdd 1	9.0	0	402	0.0	2012-11-02...

Figure 4.23: DPAT yield impact by test

Algorithm	Percentage of Wafers
Part Average Testing Algorithms	48% (17% Robust DPAT)
Spatial Algorithms	71%
SBL and BMY	66%
Automated Visual Inspections	17%
Manual Visual Inspections	3%

Table 4.1: Percentage of wafers running each algorithm

These algorithms have been introduced gradually over the last 6 years. During that time period, EWM has processed data for over ten million wafers. At the beginning of 2012, the percentage of wafers that ran each of the algorithms is shown on Table 4.1. NNR was not used in production yet. The percentage of wafers running these algorithms is very high, significantly contributing to device quality.

To determine the effectiveness and efficiency of these algorithms, 289,080

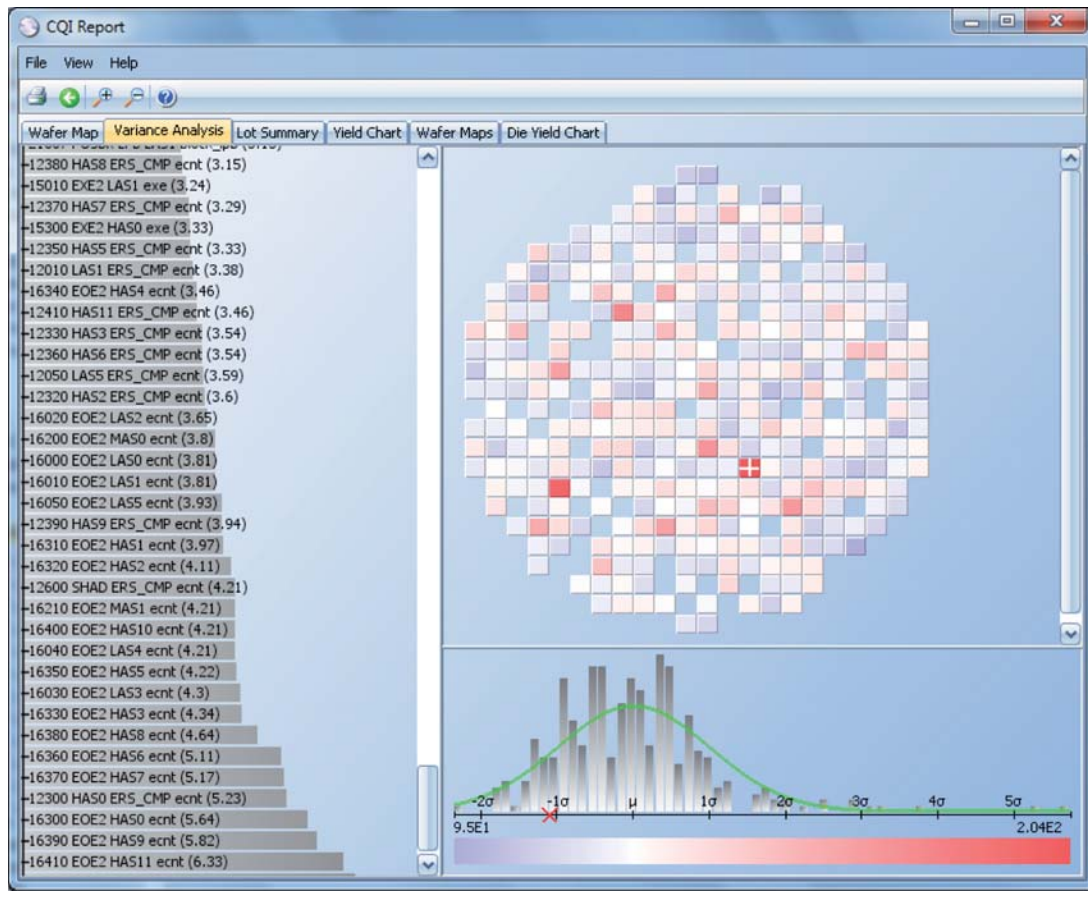


Figure 4.24: CQI report

dice on 495 wafers for the same device have been analyzed (with 205,671 good dice, 83,409 defective dice, and 26 CQIs). Such studies are extremely rare in the field since tracking CQIs is costly.

4.8.1 Spatial algorithms comparison

Figure 4.25 compares the performance of the classic spatial algorithms described in section 4.3. The X axis shows the percentage of CQIs identified and the Y axis the percentage of good dice discarded. The X and Y axes stop at 50% since the allowed yield loss for outlier detection is much less than that.

The orange straight line in figure 4.25 represents the performance of randomly marking dice as a method of detecting outliers. If a percentage of the

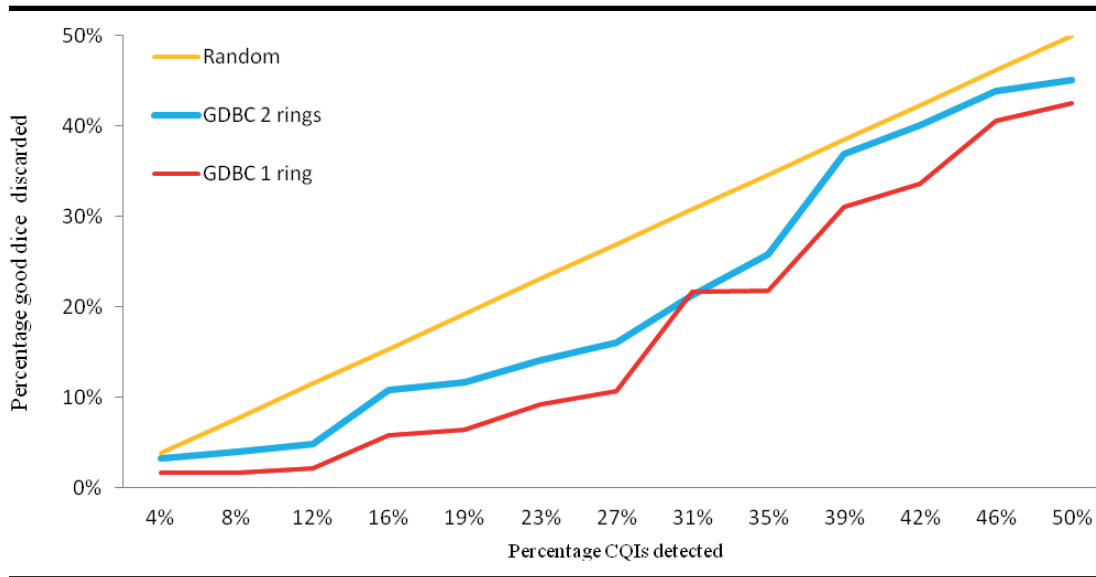


Figure 4.25: Classic spatial algorithms comparison

good dice were marked as outliers and not shipped to customers, the same percentage of CQIs would be detected assuming defects are uniformly distributed on the wafers. For instance, to identify 48% of the CQIs would require discarding 48% of the good dice. Although this is a simplistic assumption, it provides a benchmark for other methods.

The blue line shows the performance of the GDBC algorithm using 2 rings of dice (24 neighbors). Its performance is an improvement compared to the random method. For instance, to be able to identify 34.6% of the CQIs (9 out of the 26 CQIs in this study), 25.8% of good dice would be discarded. The red line is the GDBC algorithm with 1 ring (8 neighbors) which has a better performance than the 2 ring version for this device in particular. It would discard 21.8% of dice to identify 34.6% of the CQIs. The reason why the performance of the 2 ring GDBC is slightly worse than that of the 1 ring is because, with this particular device, defects beyond the immediate ring of neighbors do not contribute to identifying future CQIs and introduce noise in the algorithm.

Lastly, the two new spatial algorithms presented in section 4.4 are compared on figure 4.26. The first algorithm is called GDBC SB (Good Die in a Bad Cluster with Specific Bins) and only considers bad neighbors a specific set of bins. This version performs better than the previous ones as can be seen in figure 4.26 (green line): to detect 34.6% of the CQIs would require 15.9% yield loss which is a significant improvement over the classic GDBC 1

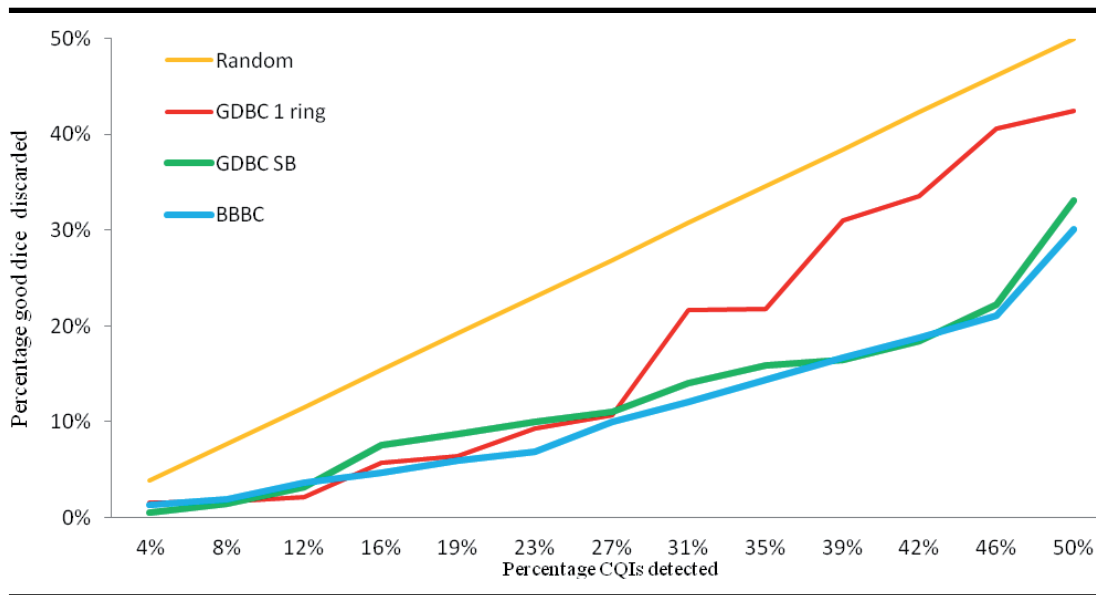


Figure 4.26: *New spatial algorithms comparison*

ring algorithm (red line). The bad bins included in this last GDBC version are determined by analyzing past CQIs and calculating the probability that a given bin number has an adjacent CQI. Figure 4.27 shows this distribution where the X axis represents bin numbers and the Y axis the probability that the given bin number has an adjacent CQI considering all the good neighbors for all the known CQIs. This probability is calculated as the number of CQIs around the given bin number divided by the total number of good dice that surround that bin number. All the known CQIs should be included in this calculation. In the absence of CQIs, all bad bins are included which defaults to the classic GDBC algorithm.

Finally, the other new algorithm created and defined in section 4.4 is called Bad Bin in a Bad Cluster (BBBC). Its performance is displayed on figure 4.26 (blue line). It has a slight performance increase compared to GDBC with specific bins: It would discard 14.4% of dice to identify 34.6% of the CQIs.

4.8.2 Part Average Testing (PAT) algorithms comparison

The classic part averaging testing based methods described in section 4.3 are compared in figure 4.28. The orange straight line represents the performance of randomly filtering dice. The red line is the classic DPAT algorithm,

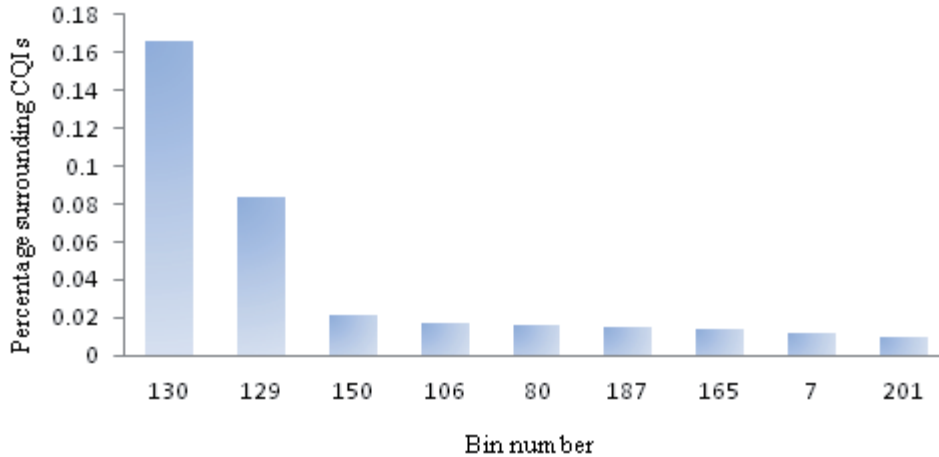


Figure 4.27: *Bin to CQI correlation*

including all the tests applied to this device, approximately 1,500 tests. Using so many tests deteriorates this method's efficiency and effectiveness since many of them have no correlation to CQIs. Identifying 34.6% of the CQIs (9 out of the 26 CQIs) would require discarding 40% of the good dice which is worse than randomly discarding dice.

A new algorithm named DPAT ST (DPAT with Specific Tests) has been developed. It only includes tests that have a correlation with past CQIs. This information is gathered with the CQI and test correlation reports presented in section 4.7.6. All known CQIs should be considered. In the absence of CQIs, a list of tests defined by the division is used. By doing so, the performance improves considerably as shown by the black line on figure 4.28. 34.6% CQIs would be screened out with just a 16.9% yield loss.

The classic algorithms AEC DPAT and NNR are also compared on figure 4.28. But they have been improved by using the specific tests (ST) enhancement as in the DPAT ST algorithm. The green line on figure 4.28 shows the AEC DPAT ST algorithm (AEC DPAT with specific tests). Its performance is slightly worse than DPAT ST: it would take discarding 24% of good dice to identify 34.6% of CQIs. The reason for this degradation is that the limits for each test are calculated with just the 1st, the 50th and the 99th percentiles which do not include much detail about the distribution. Consequently, the performance decreases slightly with respect to the classic DPAT that uses the mean and standard deviation to calculate the limits.

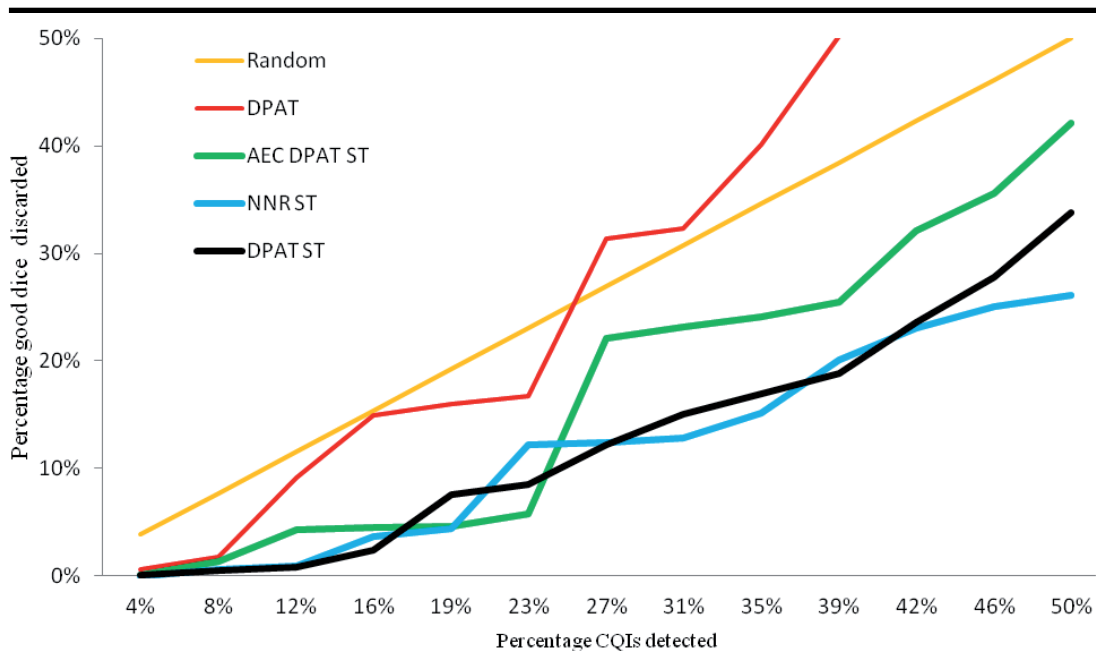


Figure 4.28: Part Average Testing algorithms comparison

The NNR ST algorithm (Nearest Neighbor Residual with Specific Tests) is displayed as a blue line in figure 4.28 and represents a slight improvement over DPAT with specific tests: to screen 34.6% of the CQIs, only 15.1% of the good dice would need to be discarded.

The other algorithms that have been created for this research and presented in section 4.4 are compared on figure 4.29. The DPAT ST algorithm is shown as a red line. This algorithm identifies 34.6% of CQIs with a 16.9% yield loss.

The R DPAT ST version (Robust DPAT with Specific Tests) is displayed as a green line. Performance has been improved: the algorithm only discards 16.5% good dice to screen 34.6% CQIs. By using robust statistics (Grubbs algorithm), possible outliers do not bias the dispersion estimation. And by transforming non-normal data to normal, skewness is factored in (better than the AEC DPAT method).

Finally, the multi-site DPAT ST algorithm that has been created significantly improves the performance (blue line). With just 8.9% yield loss, 34.6% CQIs would be identified. The multi-site option splits results by probe site, eliminating the site to site variance in the outlier detection process thus increasing performance.

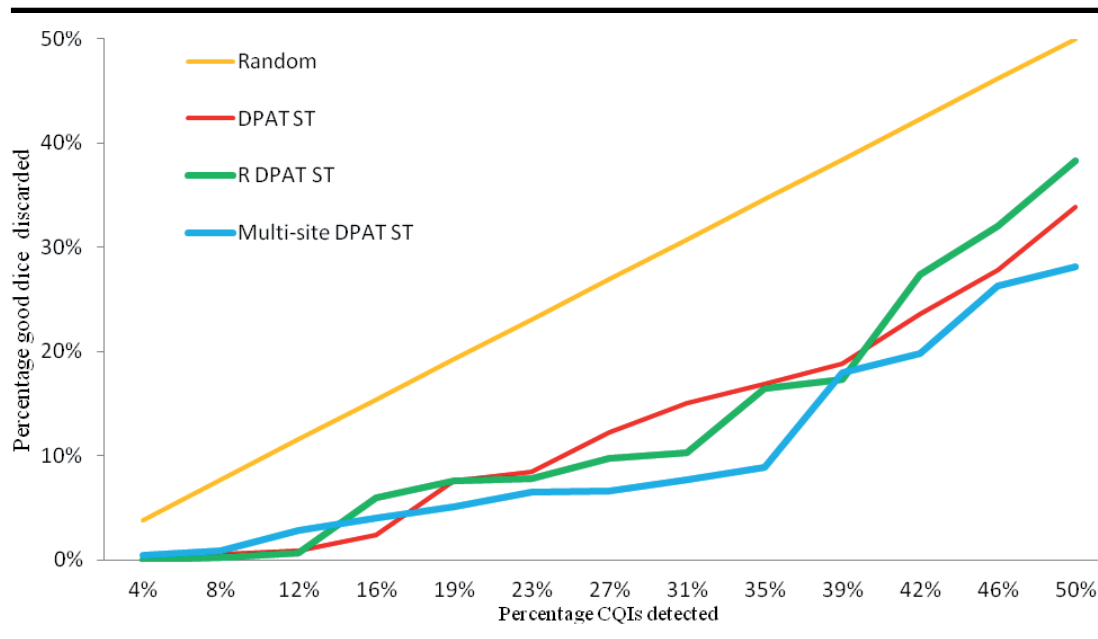


Figure 4.29: *New Part Average Testing algorithms comparison*

Although this exhaustive analysis only includes one device, feedback from other devices suggests that these results can be extrapolated to other devices since qualitative behavior is similar.

4.8.3 Additional benefits

In addition to the quality improvements of outlier detection algorithms, the following benefits have been realized:

- SPAT can be executed in EWM. The SPAT algorithm is traditionally run inside the test program. The performance of this algorithm is poor, but it is normally run to avoid shipping parts outside the specification limits. The limits for SPAT need to be updated every six months to adjust to changing conditions and the test program needs to be re-qualified which is time consuming. By running SPAT in EWM, the test program revalidation step is avoided. In addition, SPAT limits in EWM can be easily relaxed to allow more efficient algorithms (DPAT, NNR) to be the front line of outlier detection.
- Multi-site DPAT and NNR provide a clear advantage when using multi-site probe cards, which is the typical case. If site to site variance is

considerable many dice can be erroneously marked as outliers if the multi-site option is not used.

- Algorithms can be easily set up in EWM. All tests can easily be retrieved from STDF files for the device engineer to choose which ones will be used for DPAT and NNR.
- There is a complete control over algorithms and roadmap. New algorithms can be easily and quickly implemented, developing an advantage over competitors with a similar product. New algorithms can be created and implemented in an area that is strategically important to Freescale.
- EWM supports inkless assembly by implementing the SEMI E142 format [75] for wafer map data transfer. This standard has been elaborated in conjunction with other semiconductor companies. Savings are incurred by not having to ink wafers to mark faulty dice before sending them to assembly.
- The use of this industry standard format for wafer map transfer allows one single data format for external probe and assembly subcontractors presenting a common interface and decreasing development time.
- Deployment of this application to all probe floors in Freescale was carried out in conjunction with process standardization across all probe floors, decreasing variation (and development time) and, consequently, increasing quality. Requirements from all sites were collected throughout several years and prioritized to implement first the ones that had the greatest impact.
- EWM has also increased automation by integrating completely into the unit probe operation, interfacing with the MES for lot split, merge, hold, release and automatic die count updates and with the ERP system for automatic wafer map transfer to assembly. By increasing automation, labor costs have been reduced.
- Finally, hardware resources used by EWM are optimized. Database footprint for wafer maps and recipes stored is small. This is achieved by compressing the wafer map information before storing it in the database. This compression schema is also used when transferring information for display to the user interface, minimizing network traffic.

Chapter 5

Equipment Utilization Tracking and Improvement in Probe and Final Test Areas

Productivity is being able to do things that you were never able to do before.

Franz Kafka

Semiconductor manufacturing follows a cyclical business model in which demand varies considerably. The need to increase equipment utilization is paramount, especially to survive down cycles. A small investment in maintenance can increase utilization considerably but the first step is to measure equipment utilization. Measuring equipment utilization should include enough detail so that utilization losses can be clearly identified, specially the conditions and factors that cause them. These losses can be addressed with proactive and reactive maintenance to drive utilization up. Due to the multitude of factors that contribute to utilization losses, actions implemented to correct them may not have the anticipated effects. Thus, a careful analysis of utilization trend is necessary to quantify the effect of such actions.

Moreover, companies with multiple factories must enforce standards across all the factories to ensure fair comparison. There are normally different equipment utilization tracking applications across factories with overlapping

functionality. Consequently, factory to factory equipment utilization comparisons are not possible. Systems upgrade and maintenance becomes difficult due to multiple software enablers, operating systems, and database versions.

To address all these problems, a global application has been developed during this research to track and increase equipment utilization.

5.1 Introduction

Probe and final test equipment (testers) represent a vast portion of the cost of final manufacturing operations in the semiconductor industry. Maximizing utilization is critical to keep operating costs low.

A key component to increasing equipment utilization is maintenance [19]. Too little maintenance and testers will experience down time more often. More maintenance than necessary will waste valuable engineering and hardware resources. It is critical to find a balance [31].

Along these lines, Cochetoux et al. [12] proposed transforming traditional reactive maintenance practices to condition-based, predictive maintenance.

To address these problems, an application called Tool Time Tracker (TTT) has been entirely developed and deployed to most of the probe and final test floors in Freescale Semiconductor Inc. TTT was developed utilizing the framework, architecture and MDA approach described in chapter 3.

To be able to increase equipment utilization, accurate, real-time data collection is performed to determine equipment utilization and quality metrics. Some of these metrics are defined in the SEMI E79 standard [72] while most of the metrics have been created during this research. Then, an analysis of these metrics (real-time and historical) is conducted to determine corrective actions. In addition, this application monitors these metrics in real-time to warn engineers if they degrade quickly for immediate correction. Moreover, trend analysis is performed off line to identify opportunities to improve equipment utilization and to determine the impact of corrective actions.

This chapter is organized as follows: section 5.2 reviews the related work, section 5.3 introduces the application developed in Freescale Semiconductor Inc. to track equipment utilization in probe and final test. Section 5.4 describes the real-time factory views utilized by engineers to address maintenance issues. Section 5.5 lists all the available reports to help driving equipment utilization improvements and the data retention policy. Section 5.6 addresses report scheduling and finally, section 5.7 presents achieved results.

5.2 Related work

Rasovska et al. [68], Karraya et al. [29] and Matsokis et al. [40] proposed ontology models for semantic maintenance focusing on data collection and analysis to provide advanced maintenance methods.

Batimalay et al. [9] determined that there exists a significant relationship between Preventive Maintenance (PM) and Overall Equipment Effectiveness (OEE).

Following those lines, TTT provides standard equipment performance reporting (Overall Equipment Effectiveness, utilization, detailed equipment losses, etc.) across probe and final test using the SEMI E10 state model [74], the SEMI E58 [76] sub-state model, and the Freescale EPR++ standard for equipment interfaces.

As Kalir et al. [28] stated, using Manufacturing Execution System (MES) information exclusively to determine equipment utilization does not provide accurate information. MES data is not 100% reliable except for Preventive Maintenance (PM) events. The work presented in this chapter combines MES and equipment controller events to draw a precise picture of the equipment status. Data from the MES and equipment controller are sent to TTT as soon as the equipment state changes. These events are merged in real-time providing a precise and unified view of the shop floor.

Another key contribution in this dissertation is the concept of context which is an extension of the state and includes sub-state, device, lot, operator, test program, probe card, loadboard, etc. This allows drill-down analysis which pinpoints factor(s) contributing to losses. Any numerical variable can be graphed versus any context variable and analyzed that way. This functionality sets TTT apart from other commercially available software.

By deploying TTT to all factories, an accurate factory to factory comparison of tester performance and losses is achieved.

In addition, TTT provides a real-time status display of testers and real-time reports, allowing maintenance engineers to react immediately to deteriorating conditions. Historical performance reports are also available to analyze the impact of maintenance measures implemented.

Also, TTT is aware of equipment and resources usage, notifying engineers about preventive maintenance when usage reaches maximum allowed.

Finally, the equipment data collected is used for accurate capacity planning as well.

5.3 Equipment utilization tracking

TTT collects tester data that is used to measure equipment utilization. Detailed losses are also captured by TTT allowing identifying opportunities for

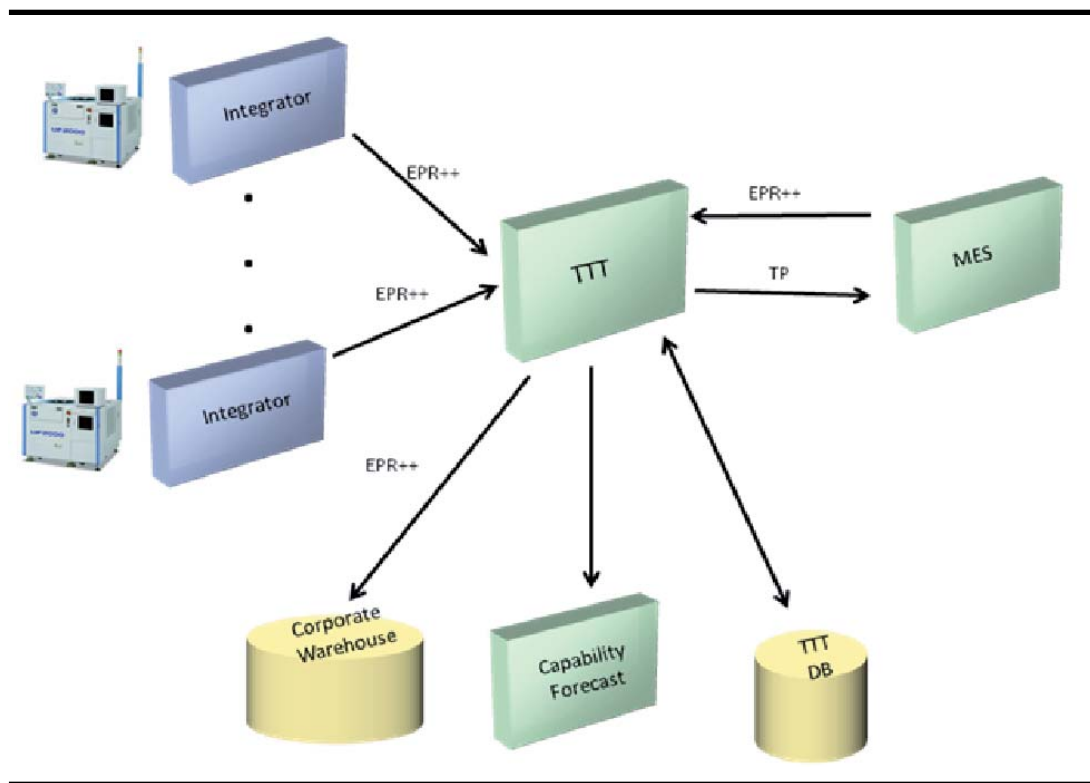


Figure 5.1: *TTT architecture*

improvement, specially maintenance related problems. TTT is written in Java with a client/server architecture, utilizing the framework, architecture and MDA approach described in chapter 3.

5.3.1 Functionality

TTT receives tester utilization data in real time. The Manufacturing Execution System (MES) also sends tester maintenance events to TTT in real time. TTT combines both sets of events to determine the state of each tester [13, 73]. Figure 5.1 shows this architecture.

Events are sent to TTT using the Freescale EPR++ standard for equipment interfaces described in subsection 5.3.4. A similar approach was used by Lee [33] to transfer machine data to higher level systems. Testers also send contextual information about their operation such as sub-state, device, lot and wafer being tested, operator running the tester, load board, probe card in use and numerical data items such as set up time, number of devices tested, yield, test time, etc.

TTT provides reports correlating any pair of contextual and numerical variables such as state to time spent in that state, load board to number of alarms, device to yield, etc. These reports are fundamental to analyze trends and determine problem root causes, identifying maintenance corrective actions (both active and preventive). This is an important contribution in this research.

All these reports are available by shift, day, week, and month. TTT also provides real-time status display of testers in production and real time reports for reactive maintenance.

5.3.2 System context diagram

Each tester in probe is controlled by a prober controller named Integrator (figure 5.1) which governs the operation of the tester. In final test, the tester controller which controls the tester and handler was developed as part of this research and it is described in chapter 3 (SC2). Both, prober and tester controllers generate EPR++ messages for TTT informing about the detailed tester activity. This happens automatically without the operator intervention.

TTT combines all the information received by the equipment controller and the MES and decides the SEMI E10 [74] state of the equipment and the sub-state according to the SEMI E58 standard [76]. Maintenance events received by the MES overwrite events generated by prober and tester controllers. TTT also sends the new state to the MES so that both, TTT and the MES, are in synchrony.

Mathew et al. [39] used a database based on the MIMOSA OSA-EAI specification to support their data driven application. In a similar fashion, TTT stores all equipment data in an Oracle database (figure 5.1).

Additionally, TTT generates periodic reports with data needed by the capacity forecasting system and it also sends equipment data to the corporate data warehousing application where engineers can perform custom queries.

5.3.3 SEMI E10 standard

The SEMI E10 specification [74] establishes six basic equipment states (productive, standby, engineering, scheduled downtime, unscheduled downtime, and noon-scheduled) as shown on figure 5.2. Additional visibility and greater resolution of equipment operation are achieved by creating sub-states according to the SEMI E58 specification [76] that map to the six basic

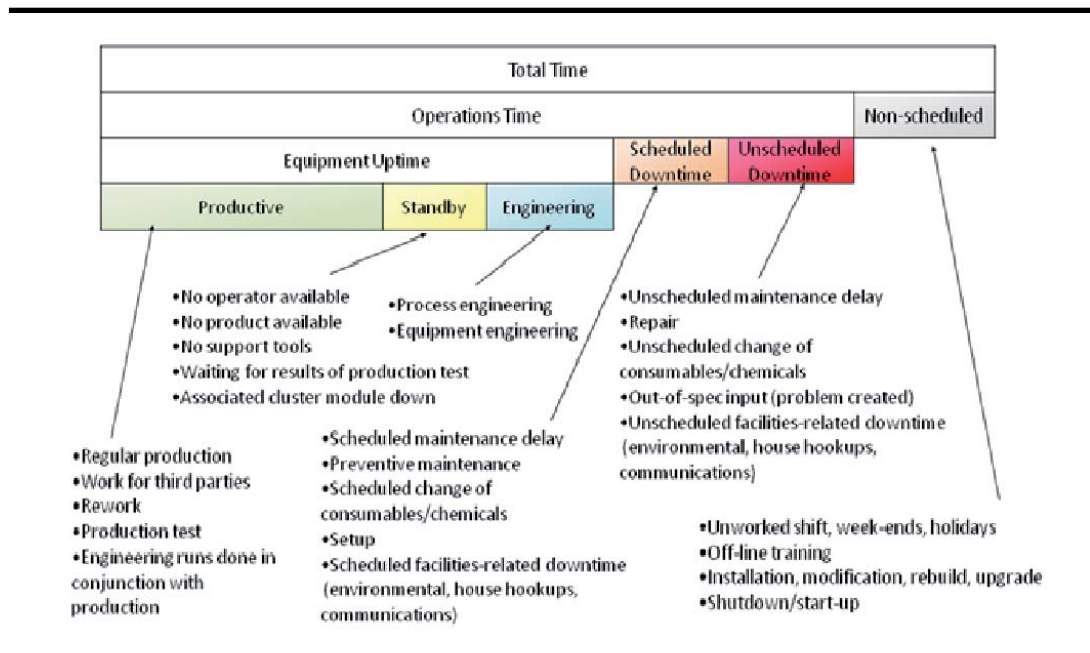


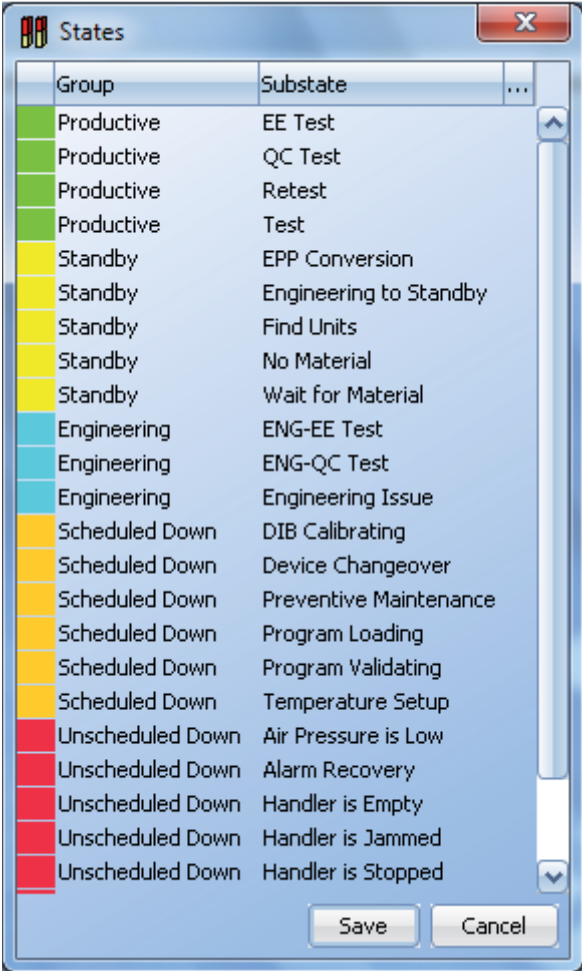
Figure 5.2: SEMI E10 standard

SEMI E10 states as displayed on figure 5.2. For instance, Process engineering and equipment engineering sub-states are mapped to the E10 engineering state. This allows customizing the SEMI E10 standard [74] to specific operating scenarios and equipment performance objectives while maintaining compliance with the SEMI E10 major state definitions and utilization metrics. Figure 5.3 shows a list with some of the sub-states defined in TTT and their mapping to one of six SEMI E10 states. The SEMI E58 standard defines 35 sub-states which have been extended to 89 to include the specific conditions for probe and final test.

By implementing the SEMI E10 state model [74], TTT provides standard equipment performance reporting across all probe and final test floors in Freescale.

5.3.4 Freescale EPR++ standard for equipment utilization

This section describes a Freescale standard set of semiconductor equipment event messages between equipment in the probe and final test areas and the targeted data collection host. EPR++ stands for Equipment Performance Reporting. It was developed in Freescale and provides a common



The screenshot shows a window titled "States" with a close button (X) in the top right corner. The window contains a table with two columns: "Group" and "Substate". The table lists various equipment states and their corresponding sub-states, each with a colored background. At the bottom of the window, there are "Save" and "Cancel" buttons.

Group	Substate
Productive	EE Test
Productive	QC Test
Productive	Retest
Productive	Test
Standby	EPP Conversion
Standby	Engineering to Standby
Standby	Find Units
Standby	No Material
Standby	Wait for Material
Engineering	ENG-EE Test
Engineering	ENG-QC Test
Engineering	Engineering Issue
Scheduled Down	DIB Calibrating
Scheduled Down	Device Changeover
Scheduled Down	Preventive Maintenance
Scheduled Down	Program Loading
Scheduled Down	Program Validating
Scheduled Down	Temperature Setup
Unscheduled Down	Air Pressure is Low
Unscheduled Down	Alarm Recovery
Unscheduled Down	Handler is Empty
Unscheduled Down	Handler is Jammed
Unscheduled Down	Handler is Stopped

Figure 5.3: States and sub-states

framework for equipment event message format allowing a single application to be used for performance reporting and a source of capacity planning and analysis. The contribution in this research is the addition of the concepts of context and numerical variables to the EPR++ standard which greatly enhance data mining.

EPR++ messages contain states based on the SEMI E10 specification [74], limited predefined equipment events (see table 5.1), context and numerical variables. As mentioned, context and numerical variables allow reporting on equipment specific losses. Figure 5.4 shows the six states defined in the SEMI E10 standard and the possible transitions triggered by the

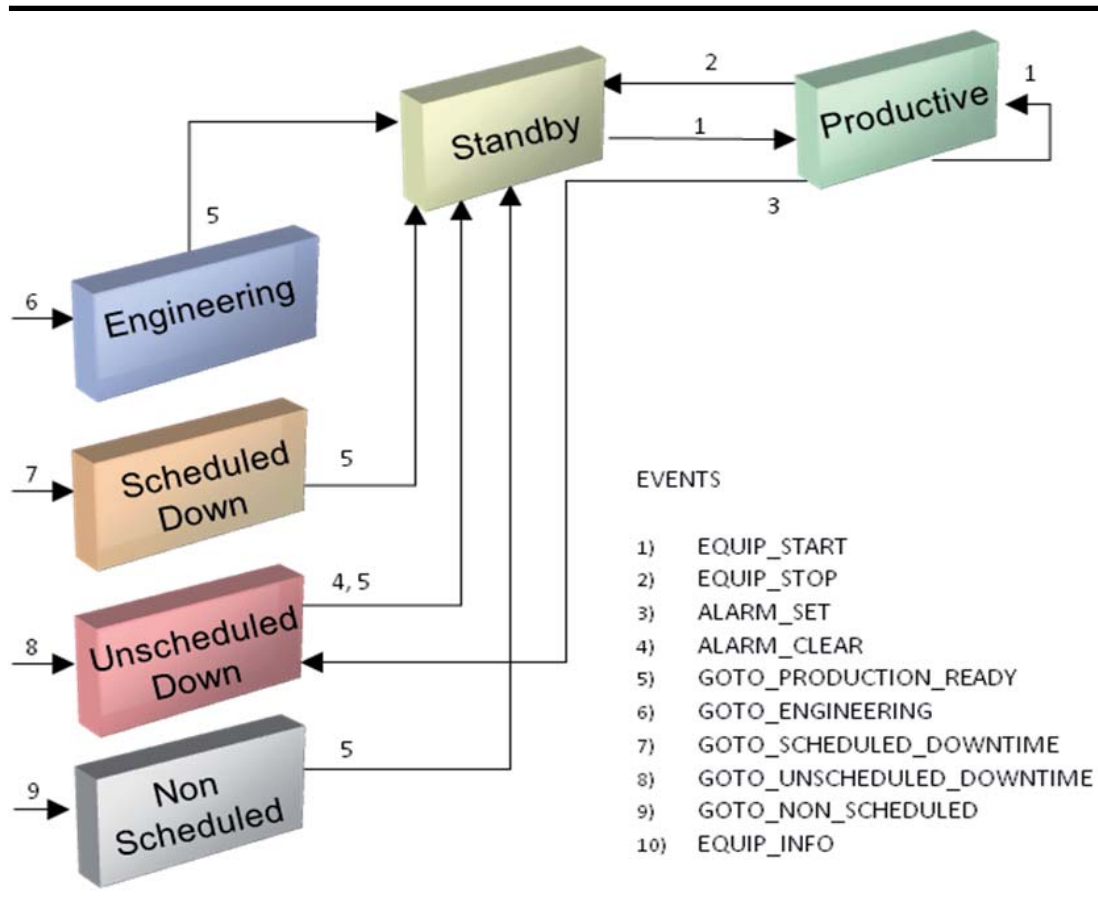


Figure 5.4: EPR++ state transitions

events listed in table 5.1.

An example of EPR++ message is listed in figure 5.5. In that example, equipment J750-5 generates a GOTO_PRODUCTION_READY event signalling that it has made a transition to the Standby state at 14:41:27 on August 11th 2010 (Event element). The previous state was Productive which had started at 14:30:55 on August 11th 2010 (State element). The context and numerical variables recorded during that productive state are listed in the message (Context and Numerical elements). Providing context and numerical variable allows collecting precise information about the equipment operation which is used for identifying factors for utilization losses.

Context variables provide information about the condition of the equipment, like lot being processed, device, operator running the equipment, etc. For probe and final test, the list of context variables is:

Event	Action
1 EQUIP_START	Operator enters lot ID, loads test program and starts testing.
2 EQUIP_STOP	Equipment is finished processing.
3 ALARM_SET	An alarm occurs which requires equipment to stop or damage to equipment, product or people will occur.
4 ALARM_CLEAR	Alarm clears and the equipment can begin working again.
5 GOTO_PRODUCTION_READY	User transitions equipment to standby.
6 GOTO_ENGINEERING	User transitions equipment to engineering mode.
7 GOTO_SCHEDULED_DOWN	Equipment goes to scheduled down state.
8 GOTO_UNCHEDULED_DOWN	User transitions equipment to unscheduled down, or an alarm occurs and the equipment needs attention.
9 GOTO_NON_SCHEDULED	Equipment is not scheduled to work.
10 EQUIP_INFO	This event is used to send updated context and numerical data items and it will not initiate a state change.

Table 5.1: *EPR++ events*

- Probe and final test: sub-state, lot, device, operator and crew.
- Probe: re-probe, probe card, probe card type, wafer, and probe pass.
- Final test: res-test, handler, load board, sites available, flow, jam alarm, test program and temperature.

Numerical variables provide quantifiable data items about the equipment operation. For probe and final test, the numerical variables are:

- Probe and final test: devices processed, good devices count and test time.
- Probe: touch increment, number polishes and wafers processed.

```

<EquipmentEvent>
  <Equipment Type="" Location="" ID="J750-5" IsPrimary="true" />
  <Event ID="GOTO_PRODUCTION_READY" Time="2010-08-11T14:41:27Z" />
  <Context Name="SUBSTATE" Value="Running" Units="" />
  <Context Name="LOT_ID" Value="Lot1" Units="" />
  <Context Name="DEVICE_ID" Value="Device1" Units="" />
  <Context Name="TEST_PROGRAM" Value="Program1" Units="" />
  <Context Name="TEMPERATURE" Value="50" Units="" />
  <Context Name="RETEST" Value="N" Units="" />
  <Context Name="SITES_AVAILABLE" Value="8" Units="" />
  <Context Name="HANDLER_ID" Value="Handler1" Units="" />
  <Context Name="LOADBOARD_ID" Value="Loadboard1" Units="" />
  <Context Name="OPERATOR_ID" Value="Operator1" Units="" />
  <Context Name="CREW" Value="C" Units="" />
  <Numerical Name="INSERTIONS" Value="4" Units="" Count="1" />
  <Numerical Name="DEVICES_PROCESSED" Value="32" Units="" Count="1" />
  <Numerical Name="CNT_GOOD" Value="29" Units="" Count="1" />
  <Numerical Name="TEST_TIME" Value="429" Units="" Count="1" />
  <Numerical Name="HND_INDEX_TIME" Value="230" Units="" Count="1" />
  <State StartTime="2010-08-11T14:30:55Z" ID="Productive" />
</EquipmentEvent>

```

Figure 5.5: EPR++ example

- Final test: index time, insertions and hard bin results by site.

New context and numerical variables can be easily added to TTT as it has been done through several releases.

5.4 Real time factory views

TTT includes a dashboard with four different views of the shop floor in real time: utilization, production, maintenance, and engineering views. These real-time views allow engineers to quickly identify the equipment that cause most of the problems. Typically 80% of the problems come from just 20% of the resources [53]

5.4.1 Utilization view

In the utilization view each icon displays the current SEMI E10 state [74] for the corresponding tester. This provides a high level view of equipment losses. Figure 5.6 shows this view. Each color represents a different state

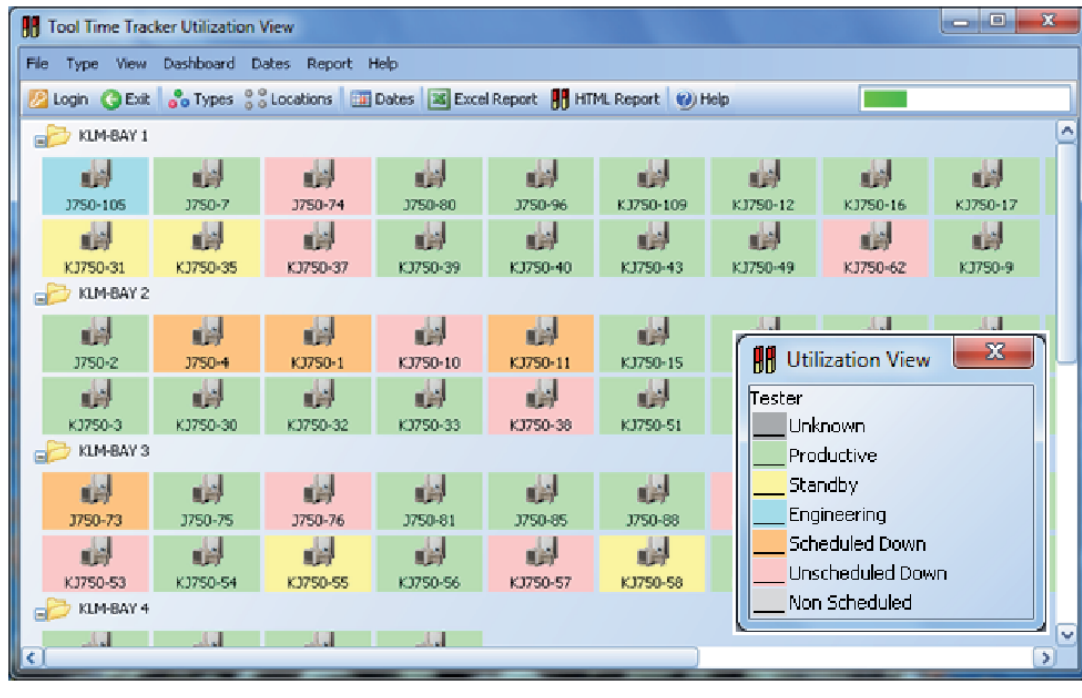


Figure 5.6: Utilization view

for quick state identification. Green is an intuitive color to indicate normal operation (productive), yellow draws attention for standby equipment and red highlights equipment that need immediate attention (unscheduled down). Finally, blue is used to denote equipment in engineering mode, orange for preventive maintenance and setup (scheduled down), and gray for non-scheduled equipment.

5.4.2 Production view

In the production view testers turn orange when they have completed testing 80% of the lot and red after 95% to flag that a new lot needs to be started. This prompts operators to bring a new lot to the tester. Otherwise the tester icon remains green to indicate that no action is required. Figure 5.7 shows an example of this view. The sub-state, device being tested and the temperature are also displayed.

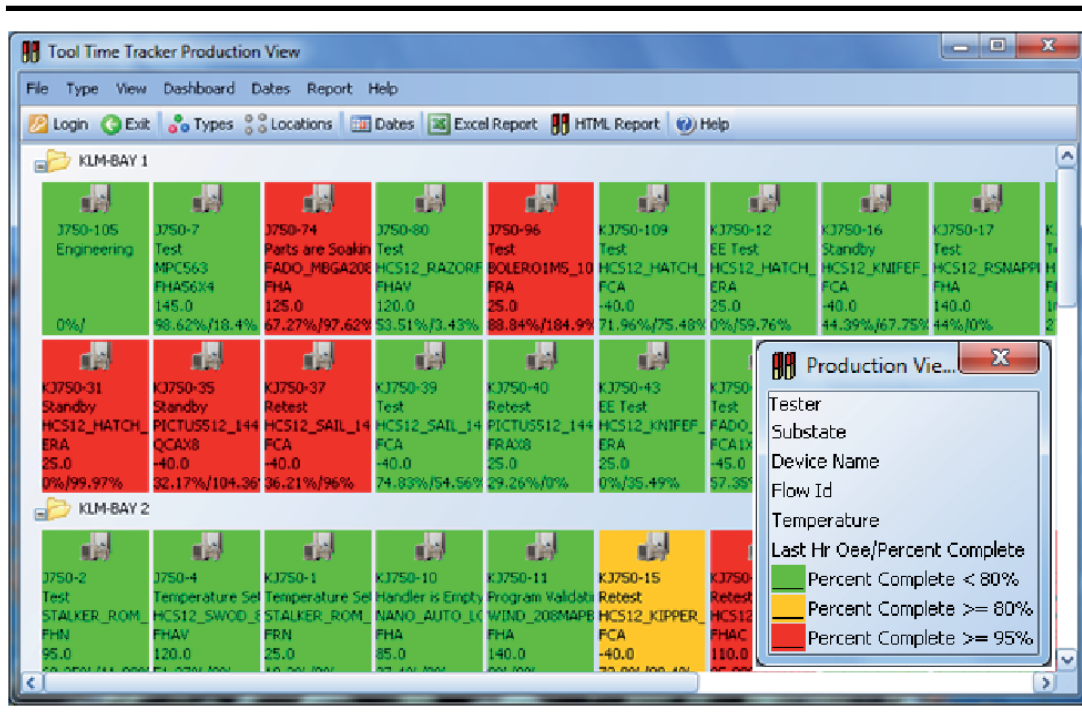


Figure 5.7: Production view

5.4.3 Maintenance view

In the maintenance view testers turn orange if current performance is under 90% and red if below 50% or the tester is in unscheduled down state (caused by the tester issuing an alarm). Performance is defined as the number of devices tested divided by the maximum number of devices that can be tested. This prompts the maintenance crew to stop the tester and repair the problem. Figure 5.8 shows an example of this view. The sub-state, device being tested and load board ID are also displayed.

5.4.4 Engineering view

In the engineering view (figure 5.9) testers turn orange if last hour yield falls under 80%, red if below 50%. Otherwise they remain green. The last hour yield is defined as the number of good devices tested in the last hour divided by the total number of devices tested in the same period. Low yield

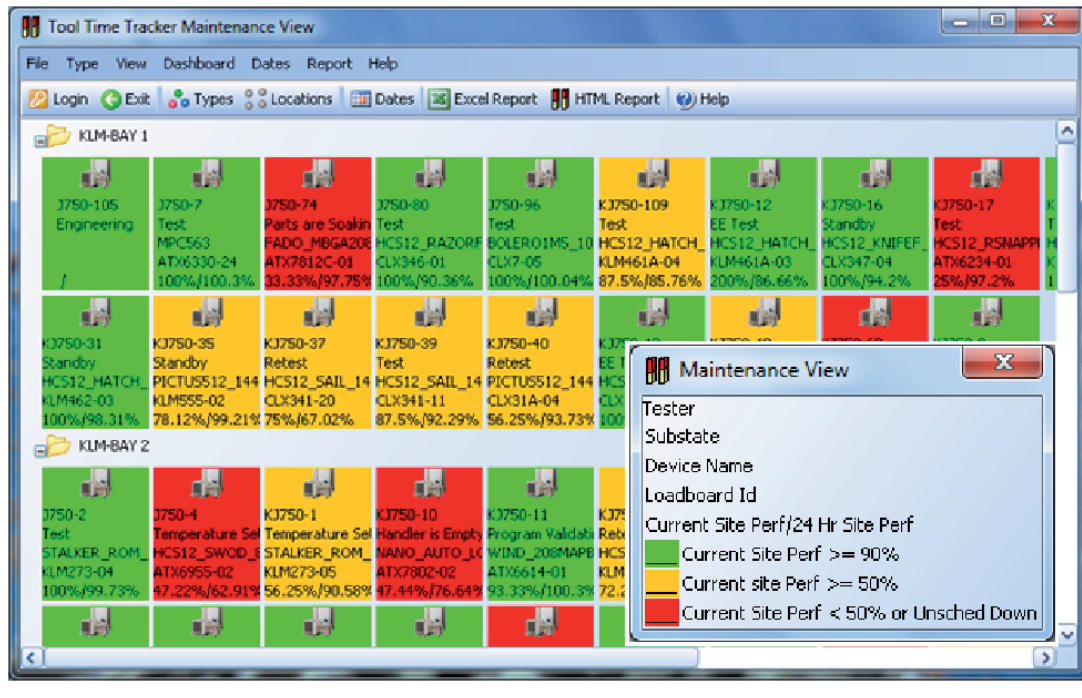


Figure 5.8: Maintenance view

may indicate a problem with the load board or probe card. Engineers will examine them to determine if they need to be repaired or if the low yield is attributed to the devices being tested.

5.4.5 Tester transition chart

The information for every tester can be expanded to show all the EPR++ events for the last 24 hours. The engineer selects a tester and a new screen pops up showing the changes in the SEMI E10 state and the main context and numerical variables for the last 24 hours. Figure 5.10 shows an example in which the changes in tester state, lot being tested, device, temperature, handler, load board and other context variables are clearly seen. The test yield produced by each test site is also displayed as a solid graph. Maintenance engineers can easily see if the yield for a given site is much lower than that of the rest of the sites and identify the load board associated to the tester at that time. If the differential is too high, the load board is serviced. This chart also allows determining other conditions that could be affecting the load board performance such as temperature and handler.

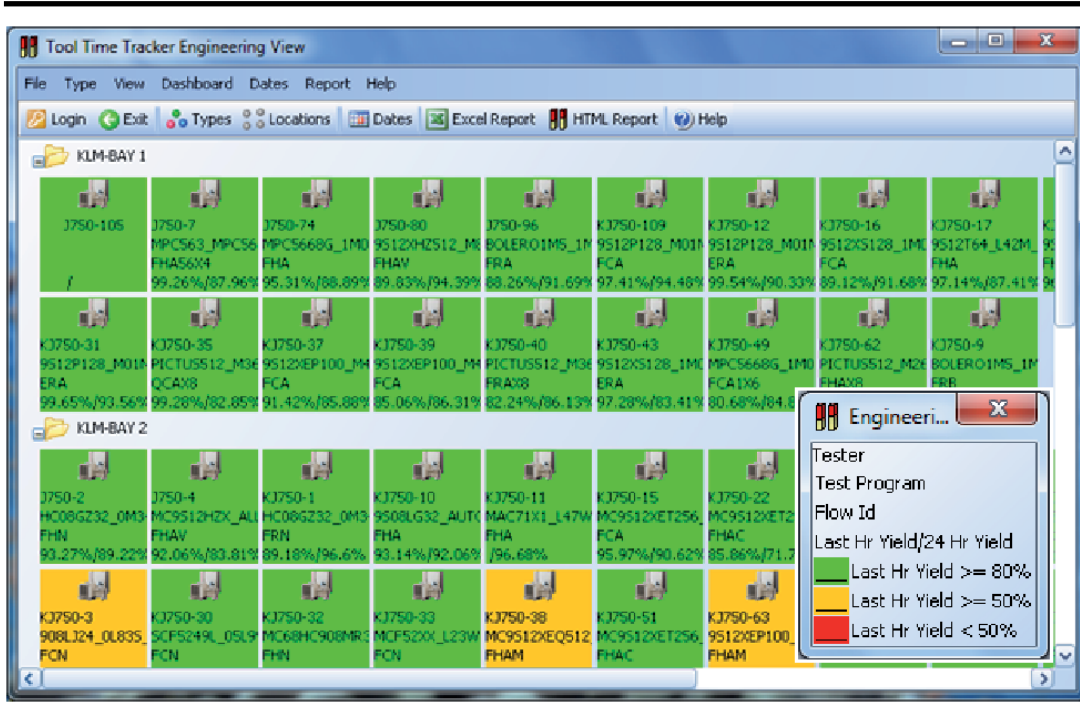


Figure 5.9: Engineering view

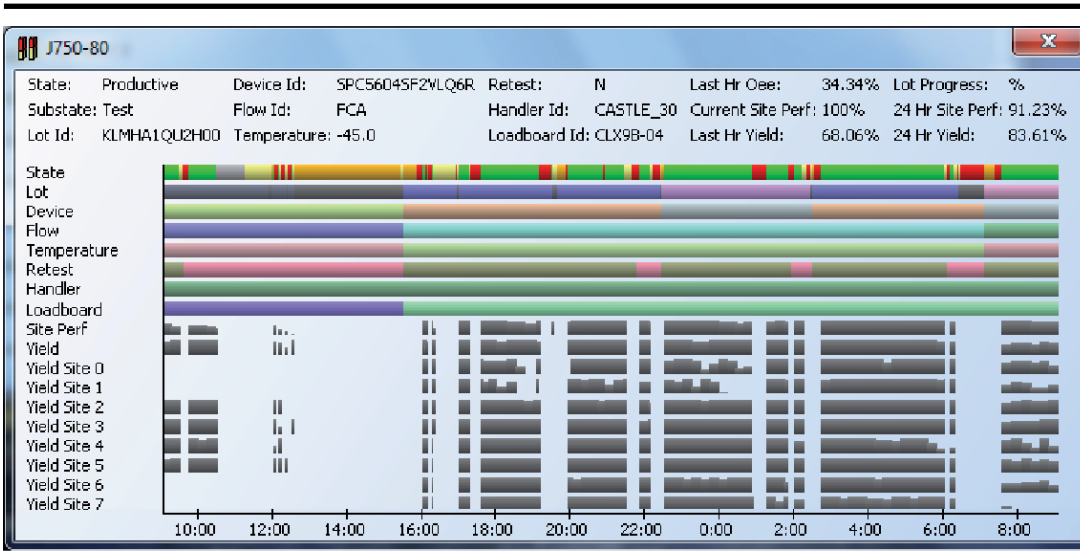


Figure 5.10: Tester transition chart

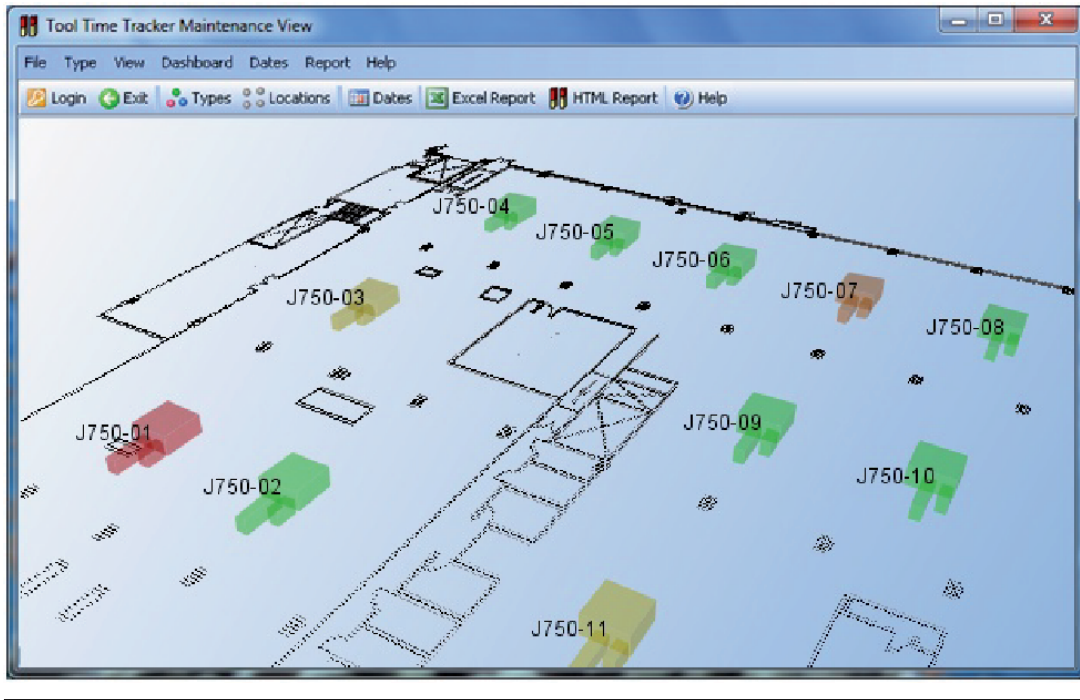


Figure 5.11: 3-D shop floor view

5.4.6 Shop floor 3-dimensional view

A 3-dimensional virtual factory view of the shop floor is also available. Testers are displayed at their physical position in the factory with the icon color indicating their state (figure 5.11).

5.5 Reports and data retention

TTT provides standard equipment performance reporting across probe and final test. A couple of these metrics are defined in the SEMI E79 standard [72] and are: Overall Equipment Effectiveness (OEE) and utilization E10. All the other metrics presented in this section have been created during this research.

These reports are fundamental to analyze trends and determine problem root causes, Engineers study historical information and perform data mining

to identify the factors responsible for utilization losses. Once that the ultimate causes have been identified, corrective actions (active and preventive) will be put in place to increase utilization.

All these reports are available by shift, day, week, and month. Users can also specify which testers or groups of testers will be included in the reports.

Additionally, an important contribution is data summarization which is performed to retain information for long periods of time while minimizing the amount of used database space.

5.5.1 Maintenance report

TTT keeps track of load board and probe card usage. Engineers specify the maximum allowed number of uses for any load board and probe card type. When this maximum usage has been reached, TTT automatically notifies the corresponding engineers to signal due preventive maintenance.

5.5.2 Utilization trend

This report shows the utilization evolution for a group of testers over a period of time. The user can select a tester type or one or more individual testers. They also select a period of time that can be given by two time stamps, or multiple shifts, days, weeks, or months. Figure 5.12 shows this report. In that example the date is 4 weeks from work week 52 in 2010 to work week 3 in 2011. The utilization evolution for the group of equipment selected can be easily observed. The overall effect of implemented measures to improve utilization can be analyzed with this report.

5.5.3 Utilization by tester

This report displays utilization for each tester for a group of testers and a time period. The utilization of each tester in the selected group can be easily compared. This report highlights testers that may have a problem. The user can select a tester type or one or more individual testers. They also select a period of time as specified in the previous section. Figure 5.13 shows an example of this report. In that report, the selected date is 1 week which is work week 1 in 2011. The utilization of each tester in the selected group can be clearly compared to determine testers that are underperforming.

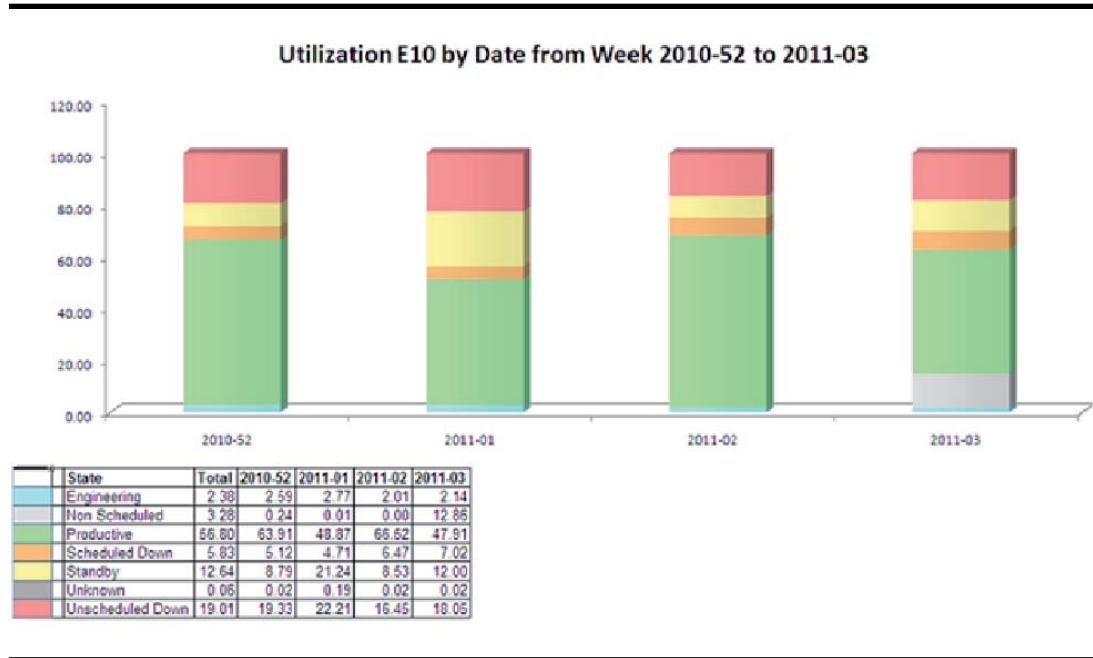


Figure 5.12: Utilization trend report

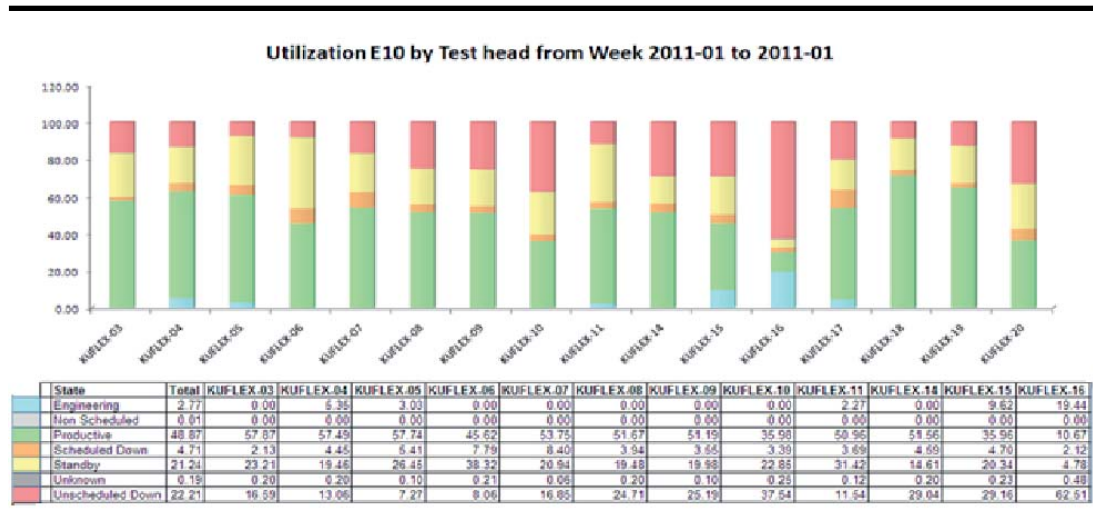


Figure 5.13: Utilization by tester report

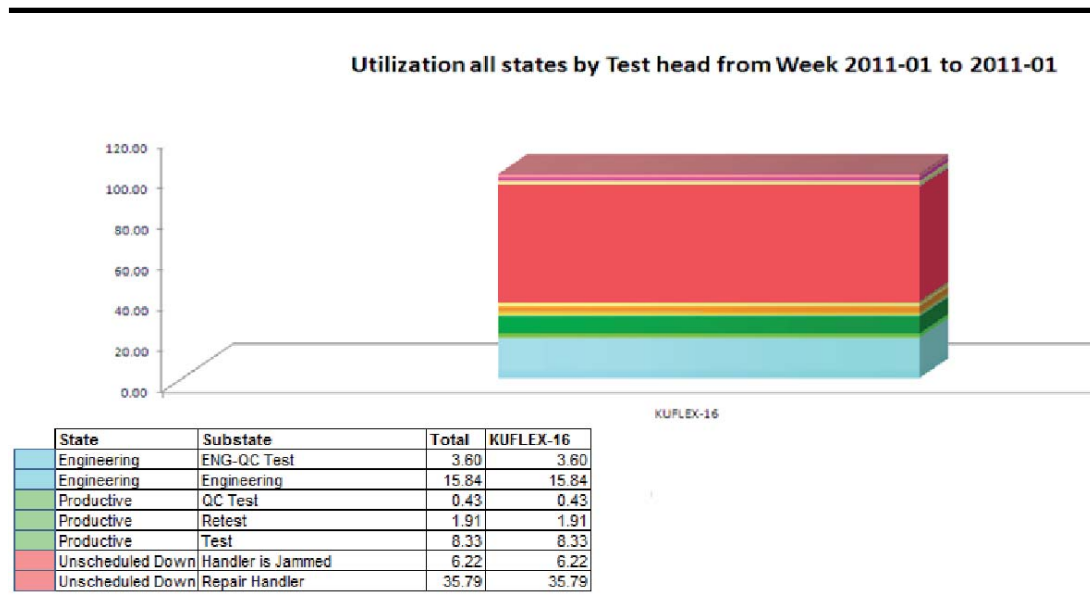


Figure 5.14: Detailed losses report

5.5.4 Detailed losses

This report shows detailed sub-states according to the SEMI E58 specification [76] for each tester. Engineers can select a tester type or one or more individual testers and a time period as specified in the previous sections. This allows identifying where losses occur. SEMI E10 states like unscheduled down are broken down into more specific sub-states to identify the exact reason why the tester was down and the time it spent on that sub-state. An example of this report can be seen on figure 5.14.

Typically, engineers will run a utilization trend report first (figure 5.12) to identify time period(s) where utilization deteriorated. On figure 5.12, it is noticeable that productive time (green color) is lower for work week 1 in 2011. Then, they would drill down by generating a utilization by tester report for that time period (figure 5.13). This report highlights which equipment has lower utilization. In this example, there is a tester with about 50% of unscheduled downtime (red color). Next, they would run a detailed losses report for that time period and equipment to uncover the specific causes as shown on figure 5.14. In this case, the handler was jamming very often (6.22% of the time spent in sub-state Handler is Jammed) and the maintenance crew

spent considerable time fixing it (35.79% in sub-state Handler Repair). A thorough repair or a new handler would have saved considerable time.

In addition to these reports, engineers can query the TTT database to perform ad hoc data mining.

5.5.5 Average test time and yield by product line

This report is used for accurate capacity planning, it provides a list of product lines (group of devices) and the associated test time and yield for the equipment and time period selected.

5.5.6 Other report types

There are many predefined reports that engineers can generate on the fly.

Each report offers several ways to group the information: by all testers, by tester equipment type, by tester, by date, by crew, by crew and tester, by device, and by product line.

Below is the list of predefined reports:

- OEE (Overall Equipment Effectiveness). This metric quantifies how well a manufacturing unit performs relative to its designed capacity, during the periods when it is scheduled to run. The formula to calculate OEE is:

$$\begin{aligned} \text{OEE} = & \frac{\text{Good devices}}{\text{Sites available}} \times \frac{\text{Test time} + \text{Index time (Test)}}{\text{Total time}} \\ & + \frac{\text{Good devices}}{\text{Devices tested}} \times \frac{\text{Test time} + \text{Index time (Retest)}}{\text{Total time}} \end{aligned} \quad (5.1)$$

- UOEE (Uncompromising Overall Equipment Effectiveness). This metric is also a simplified version of OEE and comprehends tool utilization for all products (good units, bad units, and retests). UOEE focuses on capital utilization.

$$\text{UOEE} = \frac{\text{Test time (State=Productive)}}{\text{Total time}} \quad (5.2)$$

- Utilization E10. This report shows equipment utilization using the states defined in the SEMI E10 standard [74] for definition and measurement of equipment reliability, availability and maintainability.

- Utilization all states. This report provides equipment utilization according to the SEMI E58 standard [76] which includes more details than the E10 report. Additionally, three metrics are included:

$$\begin{aligned} \text{First Pass Site} &= \frac{\text{Sites available}-\text{Devices tested}}{\text{Sites available}} \\ \text{Performance Loss} &\times \frac{\text{Good devices}}{\text{Devices tested}} \\ &\times \frac{\text{Test time} + \text{Index time (Substate = Test)}}{\text{Total time}} \end{aligned} \quad (5.3)$$

$$\begin{aligned} \text{First Pass} &= \frac{\text{Devices tested}-\text{Good devices}}{\text{Devices tested}} \\ \text{Yield Loss} &\times \frac{\text{Test time} + \text{Index time (Substate = Test)}}{\text{Total time}} \end{aligned} \quad (5.4)$$

$$\begin{aligned} \text{Retest} &= \frac{\text{Good devices}}{\text{Devices tested}} \\ \text{Gain} &\times \frac{\text{Test time} + \text{Index time (Substate = Retest)}}{\text{Total time}} \end{aligned} \quad (5.5)$$

- Devices processed. It shows the total number of devices processed by the equipments selected in the period specified.
- Yield. This report displays the yield for the equipment and time period selected. Yield is defined as the number of good devices divided by the total number of devices tested.
- Site Yield. It is defined as the good number of devices tested divided by the total number of devices tested but broken down by each test site.
- Total Test Time. This report provides the total test time for the equipment and period specified.
- Average Test Time. It is defined as the test time divided by the number of devices tested.
- Yield Loss. It is defined by the following formula:

$$\text{Yield Loss} = \frac{\text{Devices tested}-\text{Good devices}}{\text{Devices tested}} \times \frac{\text{Test time} + \text{Index time}}{\text{Total time}} \quad (5.6)$$

EQUIPMENT	STATE	SUBSTATE	LOT	DEVICE	HANDLER	TIME	DURATION	DEVICES	GOOD	TEST_TIME
Machine1	Productive	Test	Lot1	Device1	Handler1	9/26/2011 15:04:00	32000	64	62	22500
Machine1	Productive	Test	Lot1	Device1	Handler1	9/26/2011 15:04:32	40000	80	70	27000
Machine1	Standby	Idle				9/26/2011 15:05:12	60000	0	0	0
Machine1	Standby	Idle				9/26/2011 15:06:12	32000	0	0	0
Machine1	Productive	Test	Lot1	Device1	Handler1	9/26/2011	72000	144	132	49500
Machine1	Standby	Idle				9/26/2011	92000	0	0	0

Figure 5.15: Data summary

5.5.7 Data summarization and retention

Data records are summarized at the end of each shift, day, week, and month. Numerical variables are added for records whose context variables have the same values. Figure 5.15 shows how this summarization schema works.

Detailed records with information about the moment they happened are kept for 2 weeks. At the end of each shift, detailed data records are summarized for that shift. These records are kept for the last month only. Similarly, at the end of the day, week, and month, the records for the period of time that they expand are summarized. Data summarized by day, week, and month are kept for 2 months, 1 quarter, and 1 year respectively, as shown on figure 5.16. Older records are automatically deleted by TTT. These retention periods are based on the experience gathered in the factories. It is a compromise between detail needed for analysis and database space.

Using this approach allows to keep detailed records for the immediate past and less detailed data for longer periods of time. This schema retains information for long periods of time minimizing the amount of used database space.

5.6 Report scheduling

Any report can also be scheduled to be generated automatically (every hour, daily, weekly, monthly) and published on the application web site

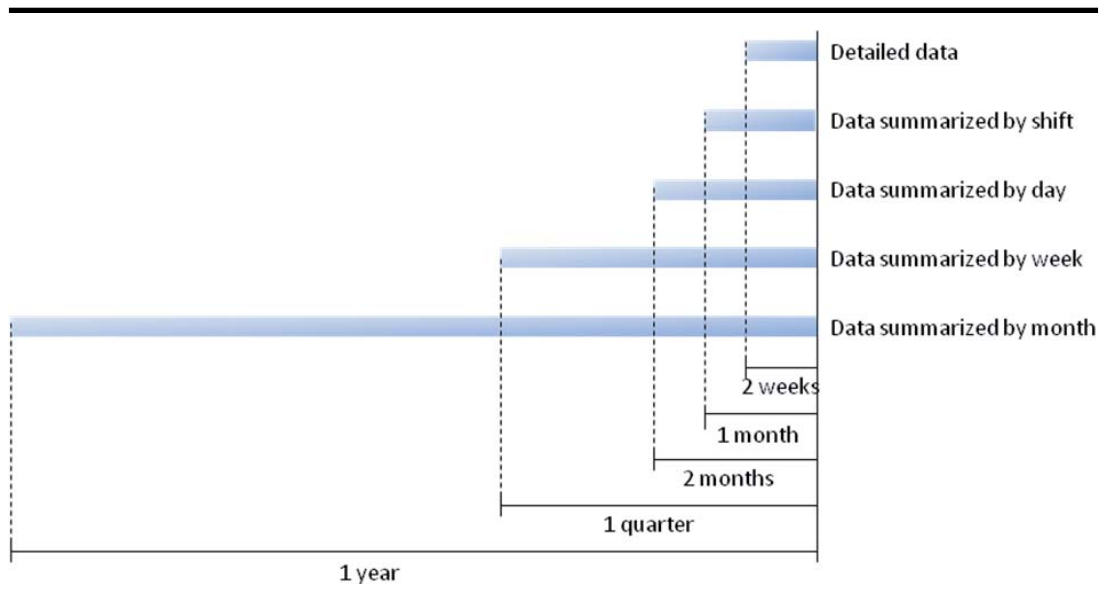


Figure 5.16: *Data retention*

and/or emailed to a list of engineers. Figure 5.17 shows the list of reports scheduled to run at different times. Administrators can add, delete or update any existing report schedule. When editing or creating a new report schedule, the screen on figure 5.18 appears.

Very little code had to be written to create these windows. The framework presented in chapter 3 automatically generates these screens from XML screens definitions. The MDA described in chapter 3 as well further reduced the amount of code that had to be written, incurring substantial savings in development time and resources.

Finally, figure 5.19 shows the application web page with the list of reports published.

5.7 Results

An application named Tool Time Tracker (TTT) has been developed using the framework, architecture and MDA approach presented in chapter 3. TTT correlates all data pertaining to state change serving the function of event router from equipment controllers and the MES. It combines equipment data and MES maintenance events in real time to reflect the state of each

Period	Number Periods	Report Type	Grouping	Types	Locations	Is HTML
Day	10	OEE	All	J750-1EX,J750-1K,J750-1KA,J750-512,...		false
Day	1	OEE	Crew	UF_1KHP,UJFX-A,UJFX-D		true
Day	1	UOEE	Crew	UF_1KHP,UJFX-A,UJFX-D		true
Day	1	Utilization all states	Crew	UF_1KHP,UJFX-A,UJFX-D		true
Day	1	Utilization all states	Test head		KLM-J750-TPM	false
Day	7	OEE	Date	J750-STX,J750-XZ		false
Day	7	OEE	Date	J750TPIP		false
Month	10	Tester Utilization	Test head	J750-1EX,J750-1K,J750-1KA,J750-512,...		false
Raw	1	Utilization all states	Crew	UF_1KHP,UJFX-A,UJFX-D		true
Raw	168	OEE	Crew and Test Head		KLM-UFLEX-BAY 1,KLM-...	true
Raw	168	UOEE	Crew and Test Head		KLM-UFLEX-BAY 1,KLM-...	true
Raw	12	Handler Jam Alarms	Test head	J750-1EX,J750-1K,J750-1KA,J750-512		true
Raw	1	Utilization all states	Test head	Ultraflex-KTM		false
Raw	12	Handler Jam Alarms	Test head		KLM-BAY 1,KLM-BAY 2,K...	false
Raw	12	Handler Jam Alarms	Test head		KLM-BAY 1,KLM-BAY 2,K...	true
Raw	12	Handler Jam Alarms	Test head		KLM-J750-TPM	false

Figure 5.17: List of scheduled reports

tester at any moment since relying on MES events only does not provide accurate information [28]. Thus, TTT is a single, accurate and unified source for equipment performance data.

It implements a simple yet powerful equipment state model based on the SEMI E10 standard [74] while recording detailed information about losses according to the SEMI E58 standard [76].

Hundreds of probers and testers send equipment events in real-time to TTT in three probe floors and three final test floors in Freescale Semiconductor Inc. TTT receives and combines MES and equipment controller events in less than a second after the state or any context variable has changed for any equipment. On average, the number of events received daily by the six TTT instances is 250K.

Equipment performance data is also fed to corporate data warehousing for analysis and to the capacity planning system.

TTT real-time reports allow engineers to quickly spot equipment whose performance is degrading. It also notifies engineers when preventive maintenance is needed by keeping track of resource usage. In addition, the real-time and historical reports allow performing data mining to identify causes for utilization losses and analyzing utilization trends to assess the effect of preventive maintenance measures implemented.

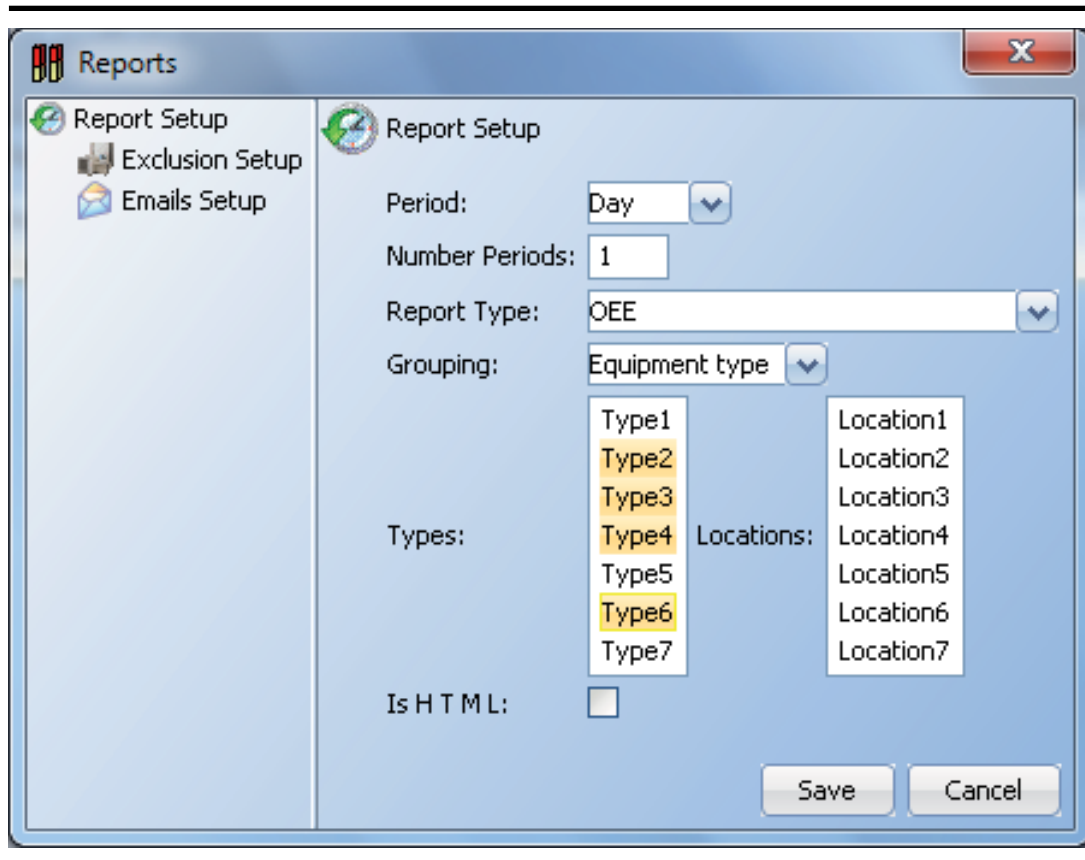


Figure 5.18: Report scheduling screen

Data records are summarized by merging the ones with the same context. By using this schema, information can be kept for months while minimizing the amount of database space needed and keeping the necessary level of detail. Table 5.2 shows the amount of database space used for one area in a factory for each summary period. The total size is 16.6 Gigabytes. If the summary process was not in place and detailed data was kept for 1 year, the amount of data needed would be 37.3 Gigabytes. Consequently, this summary schema decreases the database space needed (and the queries response times) by 55.5% without losing analysis capability.

Productive time has increased due to the leverage of TTT. Figures 5.20 and 5.21 show the increase in productive time in two final test factories in Freescale Semiconductor Inc. during the ten months following TTT deployment.

As figure 5.20 shows, factory 1 increased productive time by 7.5% in the

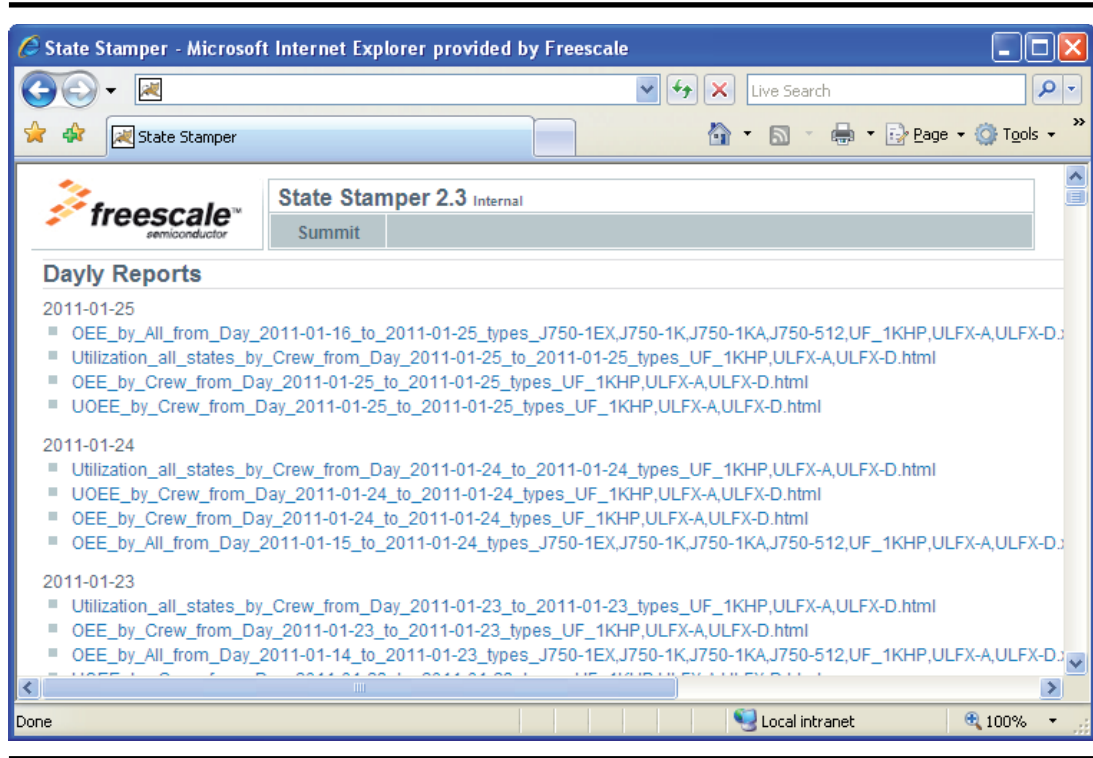


Figure 5.19: Reports published on application web page

Period	Number records	DB space (Gb)
Detailed	1,750,957	1.5
Shift	1,592,625	1.4
Day	3,194,325	2.8
Week	5,018,730	4.4
Month	7,458,976	6.5
Total	19,015,613	16.6

Table 5.2: Database size

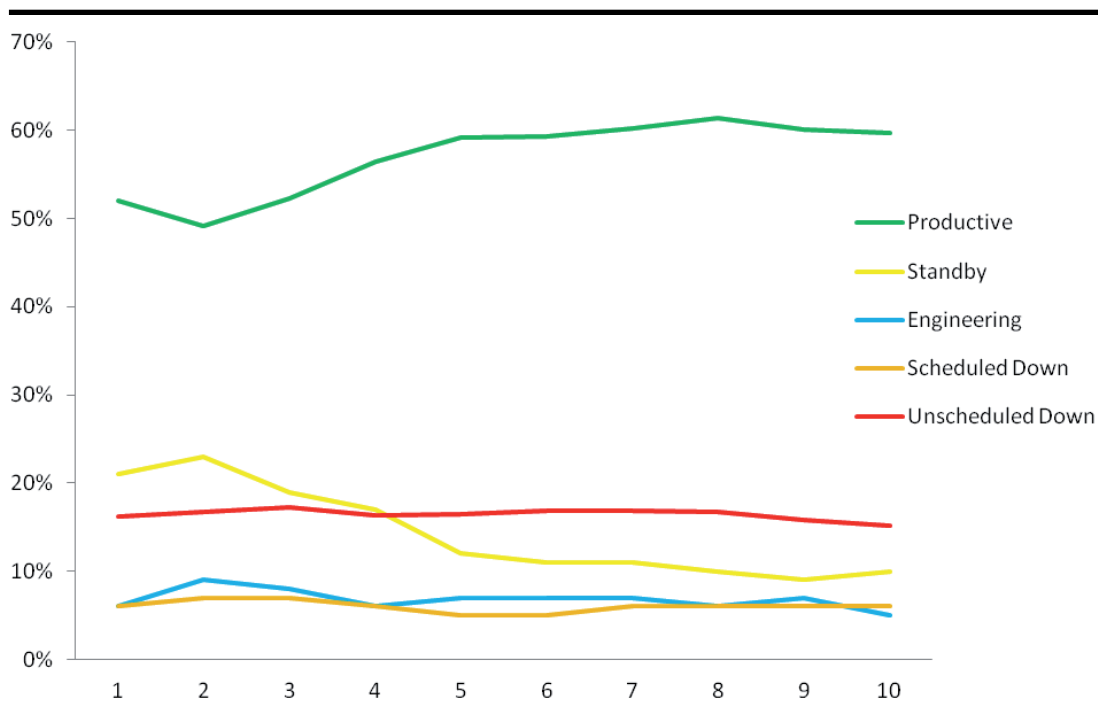


Figure 5.20: Utilization improvement. Factory 1

ten month period. Unscheduled downtime decreased slightly (from 16% to 15%). The detailed and accurate data collected by TTT allowed identifying downtime causes more precisely, allocating preventive maintenance to the items that needed it most. During these ten months, scheduled downtime and engineering time remained almost constant (around 6% each). Although, the biggest reduction was in standby time (from 21% to 10%). This was achieved by the real time displays that highlighted testers that were idle to prompt operators and managers to attend them quickly, reducing idle time.

Figure 5.21 shows that factory 2 increased productive time by 6.2% in the ten month period after TTT was released to this factory. Unscheduled downtime decreased from 13.6% to 11.3%. Again, the rich data collection approach of TTT made possible to better highlight utilization losses and address them with smarter preventive maintenance. During these ten months, scheduled downtime and engineering time remained almost constant (about 6% and 7% respectively). Similar to factory 1, there was a considerable reduction in standby time from 15% to 8%. As in factory 1, the real time dashboard provided by TTT identified testers that needed immediate assistance, reducing idle time.

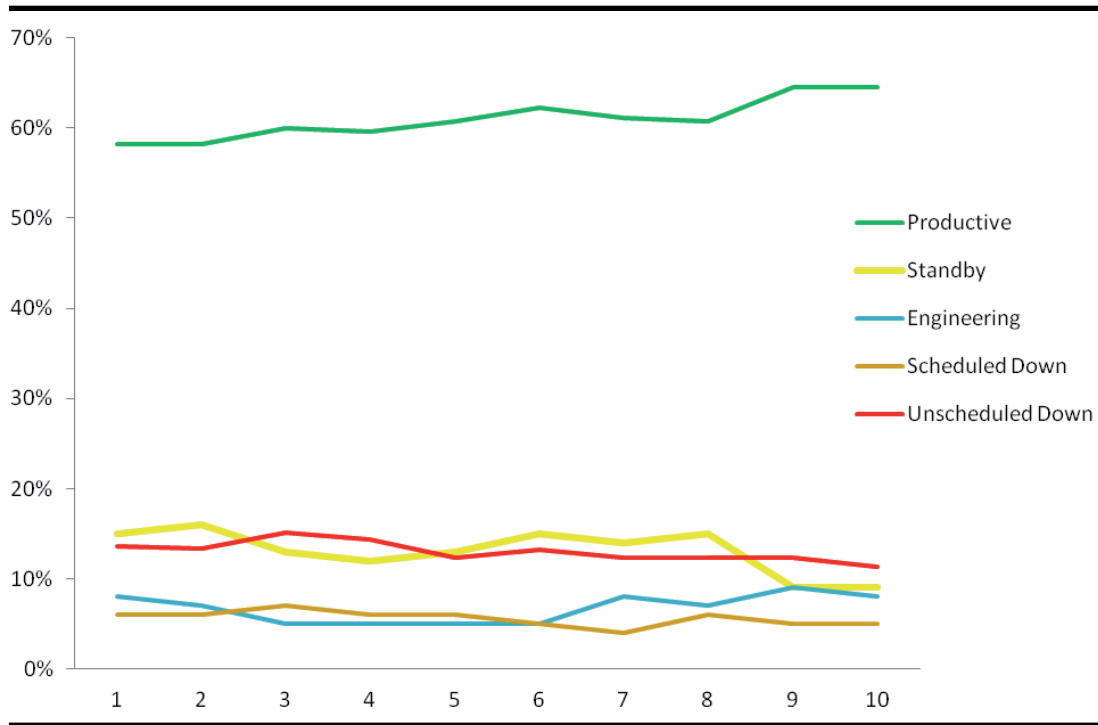


Figure 5.21: Utilization improvement. Factory 2

These results emphasize the significance of collecting detailed and accurate performance information and the importance of recording context and numerical variables to enable data mining to uncover the true causes of utilization losses.

Additionally, the detailed and accurate information collected by TTT will be the data source for a new application in development that will schedule the probe and final test areas.

Part III

Final Remarks

Chapter 6

Conclusions and Future Work

Quality is not an act, it is a habit.

Aristotle

In this dissertation, a new architecture and framework that could face the specific demands of the semiconductor industry have been introduced. Moreover, they have been applied to develop three applications used in production in the probe and final test areas in Freescale Semiconductor Inc. in Asia, Europe and North America with significant results.

Freescale Semiconductor Inc. is a global leader in embedded processing solutions for the automotive, consumer, industrial and networking markets. Freescale manufactures microprocessors, microcontrollers, sensors, and analog integrated circuits.

This framework is very conducive for quick prototyping and development. With the Model Driven Architecture (MDA) approach utilized, a fully functional application is automatically generated. Logic can easily be added to the code generated without having to implement data display and persistence functions. Thus, development time has been reduced considerably: a 69% and 40% reduction in lines of code written for two of these applications. The code generated is Java which runs on Unix, Windows and Macintosh platforms, allowing to deploy the applications to virtually any computer in the company.

The three applications are highly maintainable and upgradable: developer efficiency has improved about 25%. Junior developers are able to leverage this framework in less than 40 hours of training. Additionally, by using

class diagrams and XML screens to describe the model, documentation has improved. Moreover the use of open source tools decreased costs.

The resources used by the framework (CPU, memory, disk space, database, and network traffic) are minimized. Traffic and transmission times are reduced 93% for medium size data items and 48% for extra large ones. Average data transferred between server and clients for one of these applications alone is 125 Mbytes each day (with the framework compression approach). The average number of requests is 30,000 per day.

The first application developed with this architecture and framework is known as Station Controller 2 (SC2). SC2 is an advanced operator interface that controls both, the tester and handler hardware in final test. Hundreds of testers in three final test floors in Asia and North America are running SC2. It has increased quality and reduced operator training by providing a unified user interface as tester controller and incurred considerable savings by avoiding buying very expensive software while allowing complete customization for manufacturing requirements.

SC2 setups the tester by loading the test program and the handler by loading a handler recipe. It also monitors lot yield, site delta yield, and consecutive failure limits and it automatically disables problematic sites.

In addition, SC2 monitors handler recipe. And it will set the parameters again if they have been changed in the middle of a lot. Moreover, SC2 provides centralized configuration for tester and handler settings which are stored in a database and can be viewed or edited from any computer within Freescale while implementing a production control feature that prevents pending setups from being used. Additionally, a history of tester and handler setups is kept, including what was changed and by whom. This allows traceability and rolling back to a previous release if needed. These settings can also be easily exported to other factories.

Also, SC2 sends EPR++ events to TTT with detailed utilization data. And it integrates with the Manufacturing Execution System (MES) to download lot attributes and processing parameters. Additionally, at the end of lot, a record of the setups that were used is uploaded to the MES system for traceability.

Finally, SC2 is extremely modular. The core functionality is decoupled from the tester and handler specific drivers. New tester and handler types are easily supported by adding new drivers without having to modify the core logic. This provides a considerable reduction in development time and testing.

The second application implemented is named Electronic Wafer Mapping (EWM). It combines wafer visual and electric defects and applies outlier detection algorithms to decrease the number of Customer Quality Incidents (CQIs). EWM runs in five probe floors in Asia, Europe and North America and has processed data for over 10 million wafers to date. This contributes significantly to the company's quality initiatives by decreasing the number of CQIs.

This application implements several known outlier detection algorithms and novel variations of those as well as new algorithms developed in this thesis. These algorithms can be easily set up in EWM and tailored for the needs of each device.

An analysis of 289,080 dice on 495 wafers with 205,671 good dice and 26 CQIs has been performed to determine the efficiency and effectiveness of these methods. The novel enhancements for these algorithms and the new ones have significantly increased performance. Although this exhaustive analysis only includes one device, feedback from other devices suggests that these results can be extrapolated to other devices.

Additionally, EWM supports inkless assembly by implementing the SEMI E142 format for wafer map data transfer. This standard has been elaborated in conjunction with other semiconductor companies as part of this research. The use of this standard has decreased development time by providing a common interface with probe and assembly floors (both, internal and external).

Moreover, EWM is perfectly integrated in the probe floor and has automated multiple processes: lot splitting and merging, lot holding and releasing, automatic die count updates and automatic wafer map transfer to assembly. Consequently, labor costs have been reduced. In addition, the framework used to implement EWM minimizes hardware resources usage: the compression schema provided by the framework makes database footprint for wafer maps and recipes minimal, as well as network traffic between clients and servers.

Another application implemented with this framework is Tool Time Tracker (TTT). It runs in three probe floors and three final test floors in Freescale Semiconductor Inc. in Asia and North America. Hundreds of probers and testers send equipment events to TTT. On average, 250K events are sent daily to TTT throughout all the factories.

TTT combines Manufacturing Execution System (MES) maintenance events and equipment data in real time to capture the state and parameters of

each tester at any moment. It is a single source of standard utilization and detailed losses information across all probe and final test factories. This allows a fair productivity comparison across all factories.

It also implements a powerful equipment state model based on standards SEMI E10 [74] for high level states and SEMI E58 standard [76] for detailed sub-states to capture specific losses.

TTT real time dashboard allows engineers to quickly identify degrading conditions and react to them swiftly. Additionally, it keeps track of resource usage and notifies engineers when preventive maintenance is due. Also, historical reports allow performing data mining to identify causes for utilization losses, driving productivity increases. Utilization trend reports are also analyzed to determine the effect of the measures implemented.

Moreover, the summarization schema implemented in TTT allows keeping data for months while minimizing the amount of database space needed.

Productive time has increased 7.5% and 6.2% respectively in two final test factories in Freescale Semiconductor Inc. during the ten months following deployment.

Future improvements for the framework and architecture include using UML [81] activity diagrams (in addition to class diagrams) to describe to a high degree object oriented applications [27, 52]. The behavioral nature of activity diagrams makes them appropriate for designing the high level logic, (although it is not practical to generate all the code from activity diagrams). Combination of class and activity diagrams for code generation could produce a higher percentage of the final source code [27, 30]. In addition, complete design documents would be created in the first place.

For EWM, future plans include uploading class probe results automatically. In class probe, test programs check an entire reticle instead of individual dice. Currently, the entire wafer is discarded if a reticle is faulty or a probe engineer has to manually mark all the dice in the reticle as failed.

New reporting capabilities will also be added to display effect of algorithms on individual tests and to provide additional features for robust DPAT and other algorithms reports.

Additionally, new algorithms will be implemented, including regression as proposed by Roehr et al. [70], and Schuermyer, et al. [71]. And existing algorithms will be enhanced, for example adding the ability to choose DPAT or

NNR at the test level, and extending the GDBC algorithm to include a second ring of dice which may outperform GDBC with 1 ring when die size is small.

For TTT future enhancements, maintenance optimization decision techniques based on data collected by TTT will be implemented. A neural network [24] could be used to determine optimum preventive maintenance of load boards and probe cards based on usage and yield parameters.

Finally, the framework and architecture presented in this dissertation are being used in a fourth application currently in development. This fourth application is a scheduler for the wafer fabrication area [32, 42] and it is expected to be deployed to three factories in North America by the end of 2013.

Bibliography

- [1] K. Anderson, "Innovative Yield Modeling using Statistics" in the 17th Annual SEMI/IEEE Advanced Semiconductor Manufacturing Conference, 2006.
- [2] T. W. Anderson, and D. A. Darling, "Asymptotic Theory of Certain Goodness of Fit Criteria" in *Annals of Mathematical Statistics*, 1952.
- [3] Apache OJB website (<http://db.apache.org/ojb>).
- [4] Apache Struts website (<http://struts.apache.org>).
- [5] Apache Tomcat website (<http://tomcat.apache.org>).
- [6] T. S. Barnett, A. D. Singh, M. Grady, and K. Purdy, "Yield-Reliability Modeling: Experimental Verification and Application to Burn-In Reduction" in *Proceedings of the 20th IEEE VLSI Test Symposium*, 2002.
- [7] T. S. Barnett, M. Grady, K. Purdy and A. D. Singh, "Exploiting defect clustering for yield and reliability prediction" in *Proceedings IEEE Computers and Digital Techniques*, 2005.
- [8] S. P. Bates. "Silicon Wafer Processing"
(<http://www-old.me.gatech.edu/jonathan.colton/me4210/waferproc.pdf>).
- [9] K. Batumalay, and A. S. Santhapparaj, "Overall Equipment Effectiveness (OEE) through Total Productive Maintenance (TPM) Practices – A Study across the Malaysian Industries" in *International Conference for Technical Postgraduates (TECHPOS)*, 2009.
- [10] C. Y. Chang, J. W. Chang, and M. D. Jeng, "An Unsupervised Self-Organizing Neural Network for Automatic Semiconductor Wafer Defect Inspection" in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.

- [11] Y. Chou, A. M. Polansky, and R. L. Mason, "Transforming Non-Normal Data to Normality in Statistical Process Control" in *Journal of Quality Technology*, 1998.
- [12] P. Cochetoux, A. Voisin, E. Levrat, and B. Iung, "System performance prognostic: context, issues and requirements" in *Proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems*, vol. 10, 2010.
- [13] S. Dilorio, "SEMI Equipment Performance Standards Integration" in *SEMICON West 2006 STEP. Methods to Measure/Improve Equipment Productivity*, San Francisco CA, Semiconductor Equipment and Materials International, 2006.
- [14] Eclipse website (<http://www.eclipse.org>).
- [15] E. Estevez, I. Sarachaga, F. Perez, D. Orive, M. Marcos, "Model Driven Design in Industrial Automation" in *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.
- [16] ExoLab Castor website (<http://www.castor.org>).
- [17] R. Z. Frantz, R. Corchuelo, "A Software Development Kit to Implement Integration Solutions" in *Proceedings of 27th ACM Symposium on Applied Computing*, 2012.
- [18] Freescale website (<http://www.freescale.com>).
- [19] L. Fumagalli, F. Di Leone, F. Jantunen, and M. Macchi "Economic value of technologies in an e-Maintenance platform" in *Proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems*, vol. 10, 2010.
- [20] A. Ganguly, R. Nilchiani, J. V.Farr, "Evaluating agility in corporate enterprises" in *International Journal of Production Economics*, 2009.
- [21] F. Grubbs, "Procedures for Detecting Outlying Observations in Samples" in *Technometrics*, 1969.
- [22] W. Hargassner, B. Dorninger, "Leveraging OSGi and Eclipse Technologies for a Service Oriented Middleware in Industrial Machines" in *Proceedings of the 16th IEEE International Conference on Emerging Technologies and Factory Automation- ETFA*, 2011.
- [23] HTML website (<http://www.w3.org/TR/html>).

- [24] S. Ierace, R. Pinto, L. Troiano and S. Cavalieri, "Neural Network as an Efficient Diagnostic Tool: a Case Study in a Textile Company" in *Advanced Maintenance Engineering*, vol. 1, part 1, 2011 .
- [25] S. Illyes. "Statistical bin limits: an approach to wafer dispositioning in IC fabrication" in *Proceedings of the IEEE/SEMI Advanced Semiconductor Manufacturing Conference and Workshop (ASMC)*, 1990.
- [26] Java Excel API website (<http://jexcelapi.sourceforge.net>).
- [27] S.L. Jim, "From UML diagrams to behavioral source code", PhD. thesis, Universiteit van Amsterdam, 2006.
- [28] A. Kalir, Y. Nahum and A. Sharon, "The Hidden Productivity. How to Get More Out of Your Equipment" in *Advanced Semiconductor Manufacturing Conference, ASMC*, 2012.
- [29] M. H. Karraya, B. Chebel-Morello and N. Zerhounia, "Towards a Maintenance Semantic Architecture" in *8th IEEE International Conference on Industrial Informatics*, 2010.
- [30] H. J. Kohler, U. Nickel, J. Niere and A. Zundorf, "Integrating UML diagrams for production control systems" in *Proceedings of the 22nd international conference on Software engineering*, pp. 241–251, 2000.
- [31] K. Komonen, "A Strategic Asset Management Model: A framework of a plant level model for strategic choices and actions" in *Euromaintenance Conference Proceedings Brussels*. Belgium, 2008.
- [32] Y. F. Lee, Z. B. Jiang, and H. R. Liu, "Multiple-objective scheduling and real-time dispatching for the semiconductor manufacturing system" in *Computers & Operations Research*, 2009.
- [33] J. Lee, "Design of Self-Maintenance and Engineering Immune Systems for Smarter Machines and Manufacturing Systems" in *Proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems*, vol. 10, 2010.
- [34] A. Lineham, E. O'Toole and S Clarke, "Model-Driven Automation for Simulation-Based Functional Verification" in *ACM Transactions on Design Automation of Electronic Systems*, 2012.
- [35] W. R. Mann, "Improve Semiconductor Die Reliability" in *IEEE Design & Test of Computers*, 2008.

- [36] M. Marcos, E. Estevez, "Model-Driven design of Industrial Control Systems" in IEEE Intl. Symposium on Computer-Aided Control System Design, 2008.
- [37] E. J. Marinissen, A. Singh, D. Glotter, M. Esposito, J. M. Carulli, A. Nahar, K. M. Butler, D. Appello and C. Portelli, "Adapting to Adaptive Testing" in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010.
- [38] G. Martin, L. Lavagno and J. Louis-Guerin, "Embedded UML: A merger of real-time UML and co-design" in the 9th International Symposium on Hardware/Software Co-Design, pp. 23–28, 2001.
- [39] A. Mathew, L. Zhang, S. Zhang, and L. Ma, "A Review of the MIMOSA OSA-EAI Database for Condition Monitoring Systems" in Proceedings of the 1st World Congress on Engineering Asset Management (WCEAM), pp. 837-846, 2006.
- [40] A. Matsokis, H. M. Karray, B. Chebel-Morello and D. Kiritsis, "An Ontology-based Model for providing Semantic Maintenance" in Proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems, vol. 10, 2010.
- [41] D. Migl, "Zero Defect Mission Requires an Arsenal" in International Test Conference, 2006, VOL 2, pages 1072-1073.
- [42] L. Mönch, J. W. Fowler, S. Dauzère-Pérès, S. J. Mason and O. Rose, "A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations" in Journal of Scheduling, 2011.
- [43] M. J. Moreno-Lizaranzu, R. A. Wysk, J. Hong and V. Prabhu, "A hybrid shop-floor control system for food manufacturing" in IIE Transactions 2001 Volume 33, Number 3, 193-202.
- [44] M. J. Moreno-Lizaranzu, F. Cuesta, "Aplicaciones Avanzadas para la Automatización en la Industria de Semiconductores" in XXXIII Jornadas de Automática. Vigo, 2012.
- [45] M. J. Moreno-Lizaranzu, F. Cuesta, "Equipment Utilization Tracking and Improvement in Semiconductor Industry in Probe and Final Test Areas" in Proceedings of the 2nd IFAC-IMS Workshop on Advanced Maintenance Engineering Services & Technology (A-MEST), 2012.

- [46] M. J. Moreno-Lizaranzu, F. Cuesta, "Model Driven Architecture Approach in the Semiconductor Industry: A Practical Implementation" in Proceedings of the 17th IEEE International Conference on Emerging Technologies & Factory Automation (ETFAs). 2012.
- [47] M. J. Moreno-Lizaranzu, F. Cuesta, "Monitorización Automática en Tiempo Real de Equipos de Pruebas de Semiconductores" in XXXIII Jornadas de Automática. Vigo, 2012.
- [48] M. J. Moreno-Lizaranzu, F. Cuesta, "Electronic Wafer Mapping and Zero Defect Algorithms in Unit Probe" in IEEE Transactions on Industrial Electronics (submitted for publication).
- [49] M. J. Moreno-Lizaranzu, F. Cuesta, "Automatización, Reducción de Defectos e Incremento de la Utilización de Maquinaria en la Industria de Semiconductores" in Revista Iberoamericana de Automática e Informática Industrial (submitted for publication) .
- [50] Motorola Museum of Electronics Staff, "Motorola - A Journey Through Time & Technology" in Motorola Univ. Press, 1999.
- [51] P. Mullenix, J. Zaloski and A. J. Kasten, "Limited Yield Estimation for Visual Defect Sources" in IEEE Transactions on Semiconductor Manufacturing, 1997.
- [52] K. D. Nguyen, Z. Sun, P. S. Thiagarajan and W.F. Wong, "Model-driven SoC design via executable UML to SystemC" in 25th IEEE International Real-Time Systems Symposium, pp. 459–468, 2004.
- [53] T. O'Hanlon, "Computerized Maintenance Management and Enterprise Asset Management Best Practices", 2005 (<http://www.cmmcity.com>).
- [54] Obeo Acceleo website (<http://www.acceleo.org>).
- [55] M. J. Ohletz, and F. Schulze, "Requirements for Design, Qualification and Production of Integrated Sensor Interface Circuits for High Quality Automotive Applications" in Microelectronics Journal archive. Volume 40 Issue 9, September, 2009.
- [56] M. J. Ohletz, and F. Schulze, "Design, Qualification and Production of Integrated Sensor Interface Circuits for High-Quality Automotive Applications" in Microelectronics Journal, 2009.
- [57] OMG Model-to-text implementation website. <http://www.omg.org/spec/MOFM2T/1.0>.

- [58] M. P. Ooi, C. Chan, S. Lee, W. L. Chin, L. Y. Goh, Y. C. Kuang and S. Demidenko, "Critical Assessment of Die Level Predictor Models" in 33rd IEEE/CPMT International Electronic Manufacturing Technology Symposium (IEMT), 2008.
- [59] M. P. Ooi, C. Chan, S-L. Lee, A. A. Mohanan, L. Y. Goh, and Y. C. Kuang, "Towards Identification of Latent Defects: Yield Mining using Defect Characteristic Model and Clustering" in IEEE/SEMI Advanced Semiconductor Manufacturing Conference, 2009.
- [60] M.P. Ooi, E.K.J. Sim, Y. C. Kuang, S. Demidenko, L. Kleeman, and C.W.K. Chan, "Getting More From the Semiconductor Test: Data Mining With Defect-Cluster Extraction" in IEEE Transactions on Instrumentation and Measurement, 2011.
- [61] M. P. Ooi, Y. C. Kuang, C. Chan and S. Demidenko, "Predictive Die-Level Reliability-Yield Modeling for Deep Sub-Micron Devices" in IEEE International Symposium on Electronic Design, Test & Applications, 2008.
- [62] Oracle Java Database Connectivity (JDBC) website (<http://www.oracle.com/technetwork/database/features/jdbc/index.html>).
- [63] Oracle Java Naming and Directory Interface (JNDI) website (<http://www.oracle.com/technetwork/java/jndi>).
- [64] Oracle Java Web Start (JWS) website (<http://www.oracle.com/technetwork/java/javase/javawebstart>).
- [65] Oracle Java (1995-2012) website (<http://www.java.com>).
- [66] Oracle JavaMail (<http://www.oracle.com/technetwork/java/javamail>).
- [67] Oracle JSP website (<http://www.oracle.com/technetwork/java/javaee/jsp>).
- [68] I. Rasovska, B. Chebel-Morello, and N. Zerhouni. "Process of s-maintenance: decision support system for maintenance intervention" in IEEE Conference on Emerging Technologies and Factory Automation, 2005.
- [69] W. C. Riordan, R. Miller, and E. R. St.Pierre, "Reliability Improvement and Burn-in Optimization through the Use of Die Level Predictive Modeling" in Proceedings 43rd Annual IEEE International Reliability Physics Symposium, 2005.

- [70] J. L. Roehr, "Measurement Ratio Testing for Improved Quality and Outlier Detection" in IEEE International Test Conference, 2007.
- [71] C. Schuermyer, B. Benware, K. Cota, R. Madge, R. Daasch, L. Ning, "Screening VDSM Outliers using Nominal and Subthreshold Supply Voltage IDDQ" in Proceedings International Test Conference, 2003.
- [72] SEMI E79 Specification for Definition and Measurement of Equipment Productivity, 1999.
(<http://ams.semi.org/ebusiness/standards/semistandard.aspx>).
- [73] SEMI 116 Specification for Equipment Performance Tracking, 2007
(<http://ams.semi.org/ebusiness/standards/semistandard.aspx>).
- [74] SEMI E10 Specification for Equipment Reliability, Availability and Maintainability, 2009 (http://www.semi.org/en/standards/ctr_031244).
- [75] SEMI E142 Specification for Substrate Mapping, 2006
(<http://ams.semi.org/ebusiness/standards/semistandard.aspx>).
- [76] SEMI E58 Specification for Automated Reliability, Availability, and Maintainability, 2001.
(<http://ams.semi.org/ebusiness/standards/semistandard.aspx>).
- [77] N. G. Shankar, and Z. W. Zhong, "Rule-Based Inspection of Wafer Surface" in Proceedings. 4th International Conference on Control and Automation, 2003.
- [78] A. D. Singh, P. Nigh, and C. M. Krishna, "Screening for Known, Good Die (KGD) Based on Defect Clustering: An Experimental Study" in Proceedings, International Test Conference, 1997.
- [79] A. Solanki, K. Prasad, R. O'Reilly, and Y. Singhal, "Inertial MEMS Test Challenges" in IEEE 17th International Mixed-Signals, Sensors and Systems Test Workshop, 2011.
- [80] J. Tolvanen, "Making model-based code generation work" in Embedded Systems Europe, 2004.
- [81] UML website (<http://www.uml.org>).
- [82] R. Wysk, S. Joshi and M. J. Moreno-Lizaranzu, "Data Driven Control for Shop Floor Operations" in Intl. IEEE Robotics and Automation Conference, 1998.

- [83] Z. Yong, W. Bing, Z. Liang and Y. Yupu, "The Extended Nearest Neighbor Classification" in Proceedings of the 27th Chinese Control Conference, 2008.

This document was typeset on January 10, 2013 at 13:50 using class RGBOOK α2.14 for L^AT_EX₂_ε. As of the time of writing this document, this class is not publicly available since it is in alpha version. Only members of The Distributed Group are using it to typeset their documents. Should you be interested in giving forthcoming public versions a try, please, do contact us at contact@tdg-seville.info. Thanks!