ROI on yield data analysis systems through a business process management strategy

Manu Rehani, Nathan Strader, Jeff Hanson
LSI Logic Manufacturing Services, Inc. 23400 NE Glisan St, Gresham, OR USA 97030

ABSTRACT

The overriding motivation for yield engineering is profitability. This is achieved through application of yield management. The first application is to continually reduce waste in the form of yield loss. New products, new technologies and the dynamic state of the process and equipment keep introducing new ways to cause yield loss. In response, the yield management efforts have to continually come up with new solutions to minimize it. The second application of yield engineering is to aid in accurate product pricing. This is achieved through predicting future results of the yield engineering effort. The more accurate the yield prediction, the more accurate the wafer start volume, the more accurate the wafer pricing. Another aspect of yield prediction pertains to gauging the impact of a yield problem and predicting how long that will last. The ability to predict such impacts again feeds into wafer start calculations and wafer pricing. The question then is that if the stakes on yield management are so high why is it that most yield management efforts are run like science and engineering projects and less like manufacturing? In the eighties manufacturing put the theory of constraints[1] into practice and put a premium on stability and predictability in manufacturing activities, why can't the same be done for yield management activities? This line of introspection led us to define and implement a business process to manage the yield engineering activities. We analyzed the best known methods (BKM) and deployed a workflow tool to make them the standard operating procedure (SOP) for yield management. We present a case study in deploying a Business Process Management solution for Semiconductor Yield Engineering in a high-mix ASIC environment. We will present a description of the situation prior to deployment, a window into the development process and a valuation of benefits.

Keywords: Yield Engineering, Yield Prediction, Yield Management, Data Analysis, Business Process Management

1. INTRODUCTION: THE YIELD ENHANCEMENT IMPERATIVE

There are numerous descriptions of the purpose of Yield Engineering[2]. The following figures illustrate the imperative for yield management in an idealized situation. Figure 1(a) shows a theoretical yield curve. Over time the organization "learns" to improve the yield. The effectiveness of the effort is captured in the learning rate which is measured as yield gain per unit time. The objective in this case is a proactive objective. It is to drive for higher learning rates early in the life cycle of a product or technology in order to minimize the yield loss. Figure 1(b) shows the impact of the issues a fab faces. When an issue hits, the objective becomes to react quickly to minimize the impact and duration.

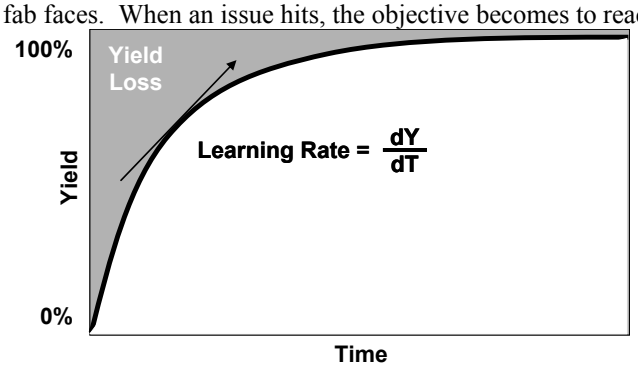

Figure 1(a): Idealized yield learning curve

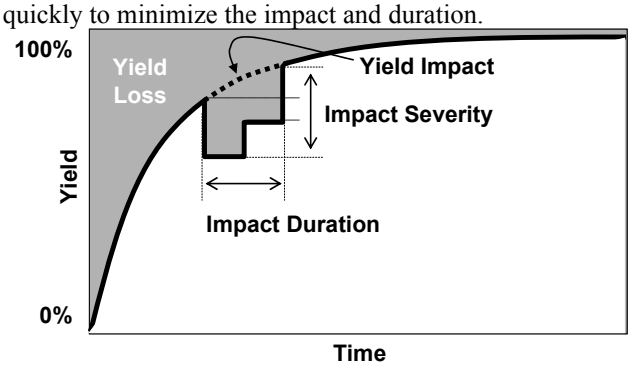

Figure 1(b): Impact of yield issues

Data Analysis and Modeling for Process Control II, edited by Iraj Emami, Christopher P. Ausschnitt, Kenneth W. Tobin, Jr., Proceedings of SPIE Vol. 5755 (SPIE, Bellingham, WA, 2005) · 0277-786X/05/$15 · doi: 10.1117/12.600057

In order to achieve these proactive and reactive targets, most semiconductor fabs have a yield engineering organization that is held accountable for meeting them. While that group is accountable for coordinating the activities they count on many other organizations like Design, Process, Defect, Equipment, Test Engineering and FA to be responsible for many of the tasks.

Achieving early learning rates require that, process-design interactions are studied; defect detection capabilities are calibrated; Equipment & Recipe qualification is completed; Product test requirements are defined and FA capability is lined up. Minimizing yield impact of fab issues demands high velocity action towards understanding the severity, identifying an early warning signal, gauging the risk of re-occurrence and non-detectability, containing the damage, identifying a root cause, quarantining the impacted material and defining a cross functional effort to eliminate the root cause. It is clear that the success of any yield management effort is closely tied to the ability of an organization to effect action across multiple functional groups. What tools methods and processes are available to a yield manager for making this happen?

More often than not, the role of a yield manager morphs from being a "yield cheerleader," cheering all the players along to playing the role of "task master," driving the troops to being the "resident yield apologist," explaining why the targets were missed and why this is the best that can be done. At LSI Logic Manufacturing Services' Gresham facility we took a hard look at this status and reached the conclusion that there has to be a better way. We defined a strategy to follow a manufacturing model and apply it towards "manufacturing" yield information and delivering it to the right people and the right place at the right time for them to act on it[3]. The result has been a software system called "Tool Suite" which integrates and collates data from multiple machine and human sources and provides it to the users as a context-aware "information product". This lends "Tool Suite" to become the information access portal of choice across job functions and changes the role of the yield manager, from those listed above, to that of a "yield execution manager," with the ability to coordinate and facilitate yield management activities across the organization.

2. DEVELOPMENT BACKGROUND & TIMELINE

The effort to find better ways to accelerate the yield learning and reduce the defect densities was started with the introduction of a new technology node. The motivation to deploy the methodology and tools came in the form of stagnating defect density trends about six months into the introduction of a new technology node. Figure 2 shows the flattening of the defect density reduction rate.

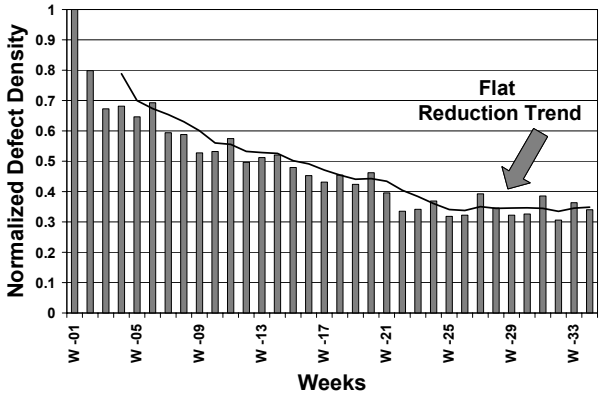

Figure 2: Stagnation in defect density reduction

A steering group was established with the charter to drive improvement in learning rates. Many examples of best practices were already in use, but were not used uniformly in all situations. Engineers were not equally proficient in performing collating the data and performing analysis. Data analysis tools were hard to use and required expert knowledge and consistent use to stay well versed.

An analysis of the way different people, groups and teams went about the business of yield management highlighted some interesting observations:

- Different teams solved similar problems at different rates with different outcomes
- The technical caliber of the teams did not vary much
- Software "savvy" was a key differentiator towards the problem solving rate
- Lack of familiarity with data sources and software tools led to situations where the team either overlooked obvious avenues towards a solution or created extra work in the form of test wafers, DOE, etc.
- Lack of knowledge of work done on similar issues slowed rate of improvement

Based on these observations it was decided that a strategy needs to be put in place to improve data access, software ease-of-use and access to lessons learned.

One of the first questions we tackled was: "If it's the best method, why isn't it used?" As each best practice was examined it seemed that there was some step, either organizationally, procedurally or infrastructure wise that was difficult to do or could not be done in all cases.

It is often said that the cause of most problems is poor communication. This certainly seemed to be true of the process for yield management. The situation is easily represented in terms of the Workflow cube shown in Figure 3.

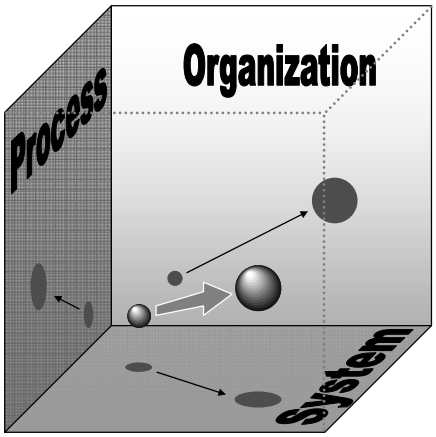

Figure 3: Workflow Cube

The 3-D balls in the cube represent the movement of a yield engineering task in the "Business Process – Organization – System Space." The further and smaller ball represents the initial state and the larger one shows the position where the task needs to be after some action. The workflow cube illustrates that as the job moves in this space it looks very different in each plane, as represented by the shadows, and no movement will happen unless there is a mechanism to facilitate the movement in the individual planes.

For example the initial state of a task may be for a yield technician (organization) to collect test data from the engineering data base (system) to decide to put a lot on hold (business process), but the next step may be for an etch engineer (organization) to review the test and lot history data in the MES (system) and scrap some of the wafers (business process). The actual yield management task must move seamlessly in all three planes. If the realities of any one of the dimensions are not met then the method falls apart creating inefficiency, work-around procedures or lack of action.

The various use cases were mapped out in terms of the context of each plane and were fed into the requirements of the effort. To facilitate proliferation of yield engineering BKM's major organizational changes were beyond the scope of the team and yield engineering, generally, does not have operational process specs. This meant that the system infrastructure had to be designed to provide information to various uses cases in a flexible manner in order to create a robust workflow. That encoded the BKM's into the software tools

3. SOFTWARE DEVELOPMENT APPROACH

The challenge of developing a software system to satisfy cross functional requirements like those required in this case is that it is virtually impossible to gather and define all the requirements needed for the complete solution. At the beginning of the development process, the users and organization will have only a vague idea of what the final solution would look like. If they are pegged to commit to the requirements, those vague ideas will be translated into system development specs. When delivered, the solution will be very close to what was asked for but rather far from what was needed. This would be a common predicament for software designed under a traditional development model. Figure 4 illustrates the timeline and effectiveness scenarios of such an approach.

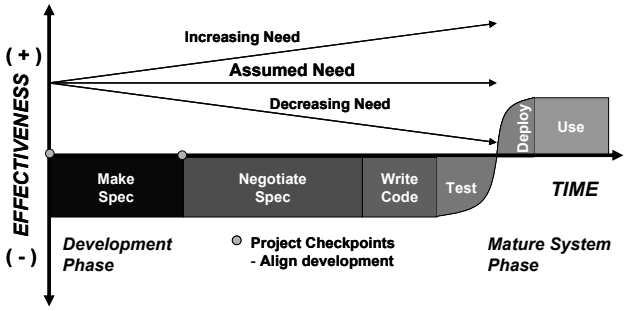

Figure 4: Traditional software development model

There are two main points of interaction with the users. First at the initiation, to understand the need (represented by the Assumed Need line) and the second to get signoff on the spec, which generally drags on for a while. The problem is that from the start the users see very little outcome and only see the effort as a resource sink. After the coding and testing there is a brief deployment phase which may or may not involve the users and then they are handed the keys. At this point the users start seeing some utility of the effort, but more often than not, either the need was miscalculated, or over this long period the need has increased in magnitude or went away. One way around this is to increase user involvement and aid the definition of the final product through, what we call "applied prototypes[5]." These are prototype software releases that attempt to match the required functionality and don't focus on the robustness or quality of code and are actually released for use. This achieves two objectives, one it allows the users to define their needs by actually using a prototype system to do their job and two since the code is not stable the users themselves develop an appreciation for rigorous code and demand it. This gets their support for later stage code improvement efforts. The timeline and utility of the "Applied Prototyping" approach are shown in Figure 5.

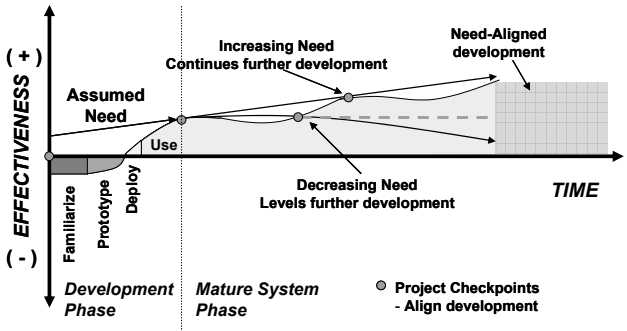

Figure 5: Applied Prototyping based software development

In this approach there are multiple points of interaction between the developers and users. The approach is characterized by a quick phase where, instead of asking the users what they want, the developers familiarize themselves with the job of the users by taking on data system tasks that the users generally do. For example if the users are analyzing the results of a DOE, the developers will run the software under supervision of the users. If the users are

preparing a report then the developers would take on the job of collating the data as instructed by the users and so on. This way the software developers get a keen understanding of the data system related bottlenecks in the business process and can design prototype solutions to solve the problem. The prototype is then released for use and a schedule is set to review the benefit and align future development based on need. A user champion is identified and he or she works closely with the developers to get the functionality to an acceptable level. The application is considered to be in the mature phase at that point and the developers can focus on shoring up the stability. In a traditional development model it is common to get prototypes too, but the main differences are that one the prototypes are not for release and two the requirements define the prototypes which are then used to verify the understanding of the prototypes. In an Applied Prototyping approach instead of "requirement driven prototypes" we have "prototype drive requirements."

4. MANAGING THE YIELD ENGINEERING BUSINESS PROCESS

At a high level the business process of yield engineering can be boiled down to the process of executing faster and better execution of OODA loops[4]. OODA refers to Observe, Orient, Decide and Act as shown in Figure 6.

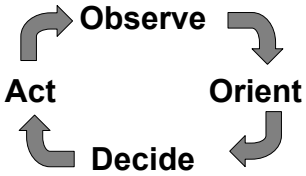

Figure 6: OODA Loop

These are executed through multiple steps across many systems. The steps in the yield engineering flow are shown in Figure 7(a) and the OODA steps are marked in Figure 7(b)

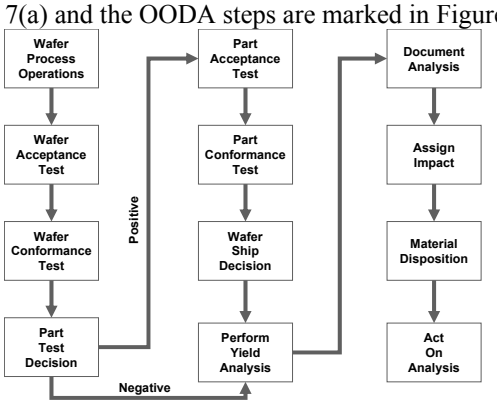

Figure 7(a): Steps in yield engineering flow

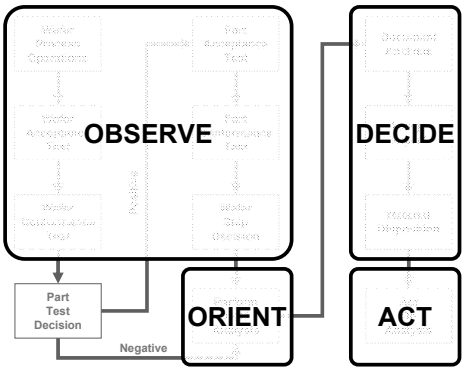

Figure 7(b): OODA phases in the yield engineering flow

Further analysis of the steps and systems involved in these steps highlighted that at our fab, during the "observation" phase, there were eight disparate systems with eleven access points. The time for "orientation" varied by lot and issue, but after that there remained four disparate systems with five access points that had to be managed over the "decision" and "action" phases. This resulted in potentially adding 50-70 minutes to all lots being held before ship.

Figure 8 shows these disparate systems along with the access points.

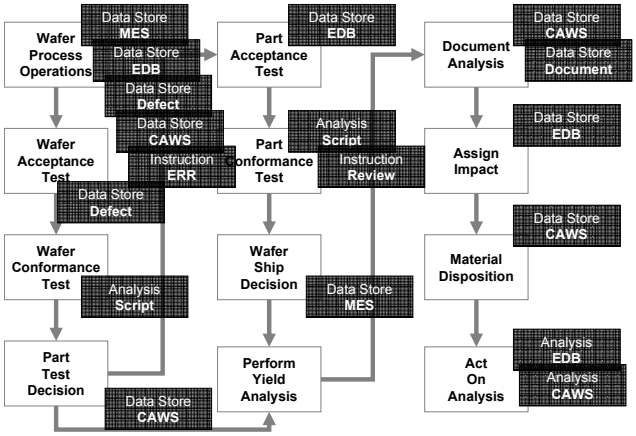

Figure 8: Data access points and systems

Key problems with the status-quo, as described here, were:

- Variation in time-at-yield-step per lot introduced unpredictability in ship dates
- The addition of cycle time was interfering with stable production goals
- Quality of orientation and decision varied from employee to employee
- Follow through to action was delayed and non-uniform across shifts

Figure 9 shows that while two-thirds of the held lots moved on in about 60 minutes or less. The other one-thirds could be held for ten hours or more.

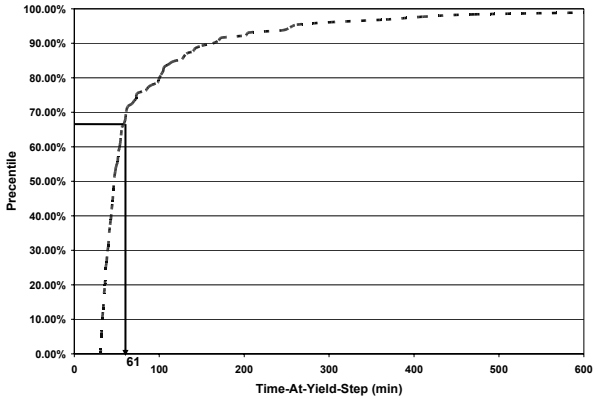

Figure 9: Time-at-yield-step distribution

5. SOFTWARE DESIGN AND ARCHITECTURE

Based on the analysis discussed above a software system named Tool Suite was designed using the "Applied Prototyping" approach described earlier. Each lot and wafer that is processed in the fab has various types of data associated with it, like SPC, Defect, Yield, Wafer Acceptance Test (WAT), Production (MES), Corrective Action Workflow (CAW), Equipment and Test.

The main view of the Tool Suite lists point of entry for various tools organized by functional area as shown in Figure 10.

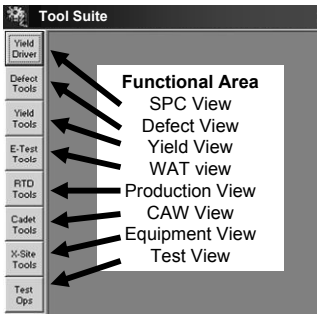

Figure 10: main view of Tool Suite

Each of these data sources has a corresponding Expert User Group (EUG) whose members specialize in the use of the industry standard systems to run the operations in their functional areas. Tool Suite does not replace those systems. Instead Tool Suite was designed to make the job of yield management more accessible for each such EUG. It was built on the assumption that each EUG looks at yield issues in their own way, but should be able to do so in one tool and collaborate with other EUG's in the same tool. For example a defect engineer working on a yield problem will first want to analyze the defect distribution, type of defects etc. and then check the yield, whereas an equipment engineer may first want to look at the tool status logs, then at the effected lots and then at the yields and a yield engineer may start from the test results and work backwards to defects and equipment. The tools shown in Figure 10 open EUG specific portals to allow navigation of the yield engineering workflow cube (Figure 3) from the perspective of the specific EUG. The design intent is illustrated in Figure 11(a) and Figure 11(b). Figure 11(a) shows how industry standard software are designed to cater to EUG's and yield engineers need to develop familiarity with all of them to do their job.

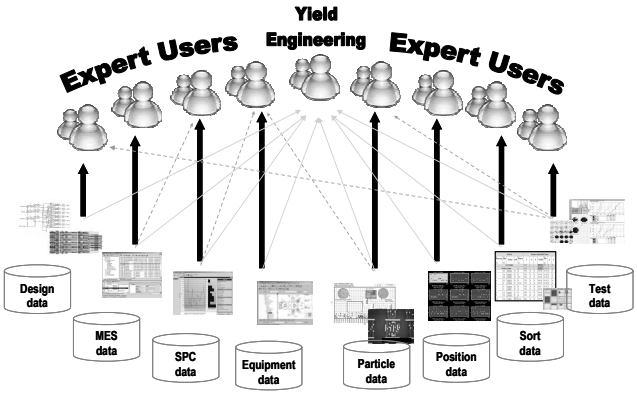

Figure 11(a): Data flow options without Tool Suite

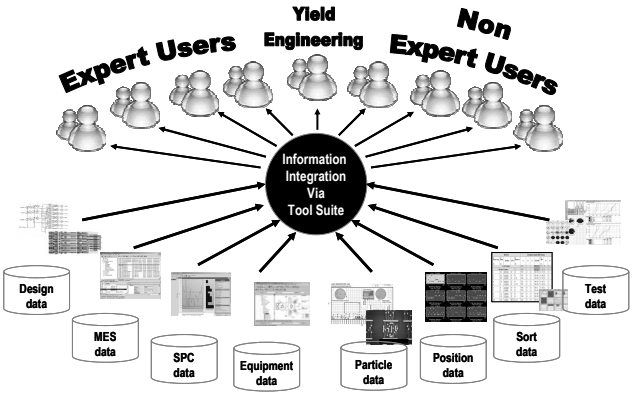

Figure 11(b): Data flow with Tool Suite

This results in keeping EUG's out of the yield engineering business since there is a barrier to learning many different software. Thus there are only a few scant crossover channels among EUG systems and users. Figure 11(b) show the role of Tool Suite as an intermediate software that is designed to server yield-relevant data from each silo system to each EUG and Yield Engineers in a format and flow that they naturally use. It is important to note that Tool Suite does not replace any of the underlying EUG native systems and neither does it re-store all their data. It merely stores some summary data to aid quick drill downs and reporting and beyond that it interfaces with the underlying systems as needed.

The architecture is shown in Figure 12. Report modules in Tool Suite use rely heavily on the summary data and the analysis modules rely on interfaced access to the native systems.

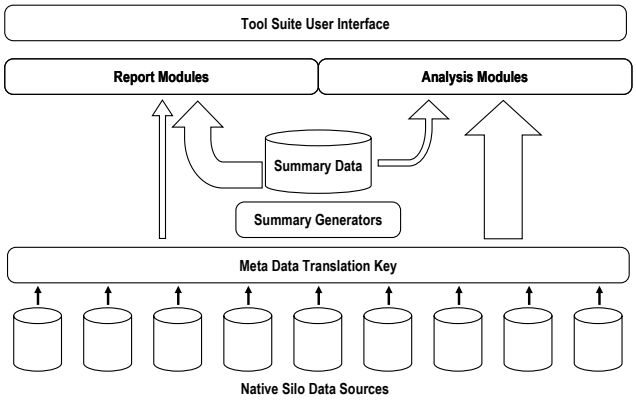

Figure 12: Tool Suite system architecture

The following figures illustrate the advantage of this design and architecture as manifested in the software. Figure 13 shows the access to defect data from two different points-of-entry.

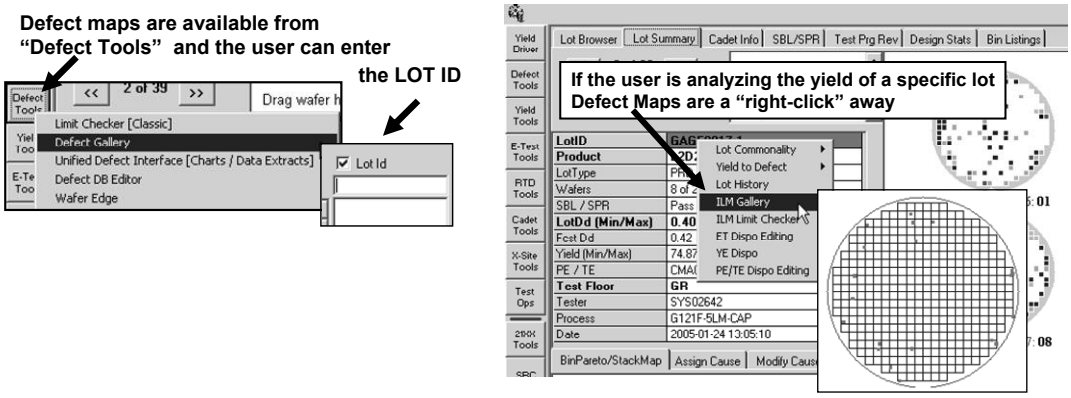

Figure 13: Access to the same defect data, but from different EUG views

The yield engineering view is the most exhaustive. It can be thought of as having done all the collating work across multiple native data sources and prepped it to be ready for review and action. In OODA terminology, it has automated the "observation" phase by gathering all relevant data and made it ready for "orientation". Figure 14 shows the drill-downs available to the yield engineering view.

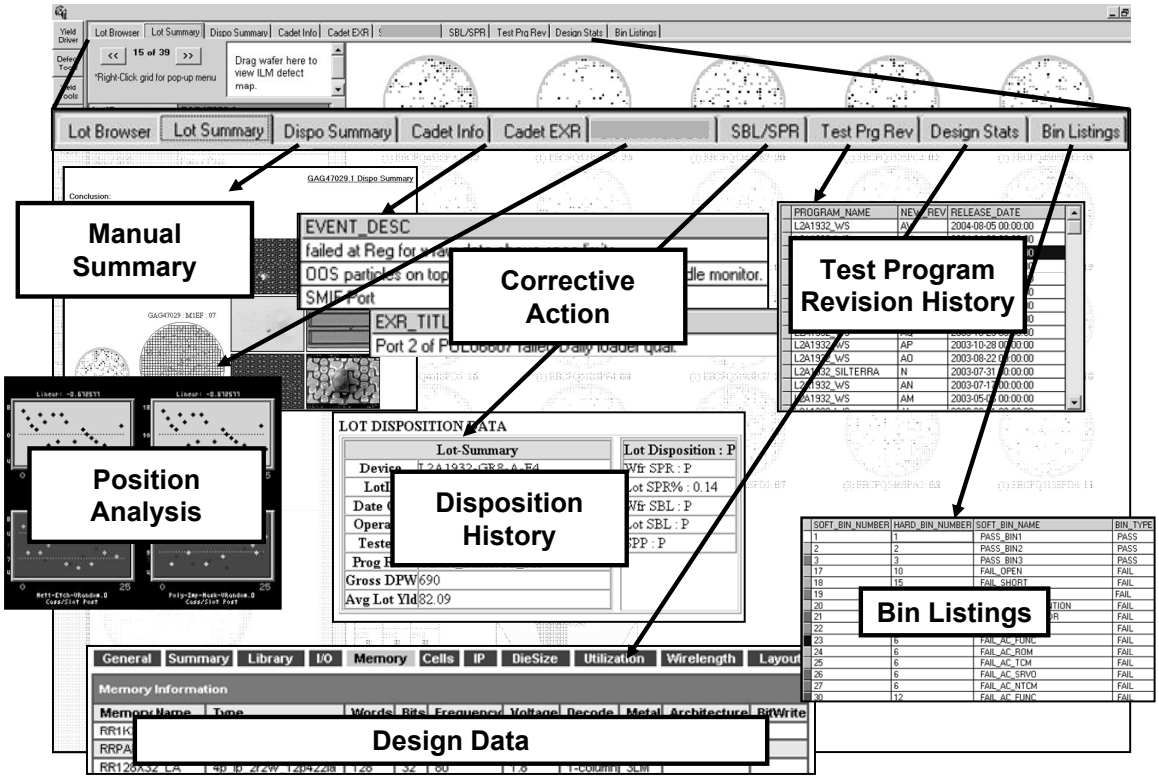

Figure 14: Yield engineering view in Tool Suite.

A list of all lots is made available on the Lot Browser tab. Then for each lot, Tool Suite pre-gathers information like manual summary comments, corrective action records, test program revisions, position analysis, disposition records, bin listings and even design data. It is this specific view that at the crux of transforming the yield engineering efforts at LSI Logic Manufacturing Services. Not only was it successful in facilitating yield learning and recruiting multiple EUG to participate, it also resulted in significant reduction in the time-at-yield-step metric

6. RESULTS

As mentioned earlier, the motivation to deploy Tool Suite came from stagnating yield learning rates. Figure 15(a) shows the overlay of the development cycle with the yield trends.

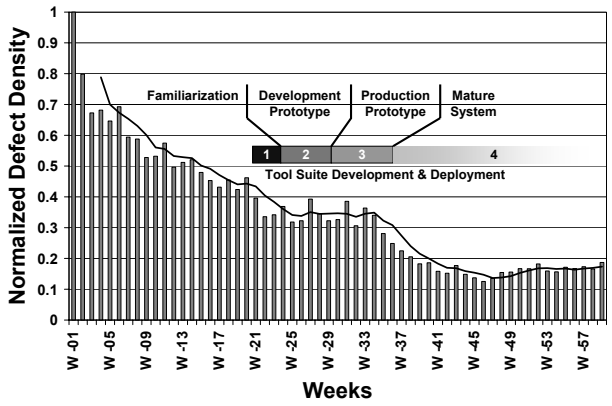

Figure 15(a): Defect density trend along with Tool Suite development & deployment timeline

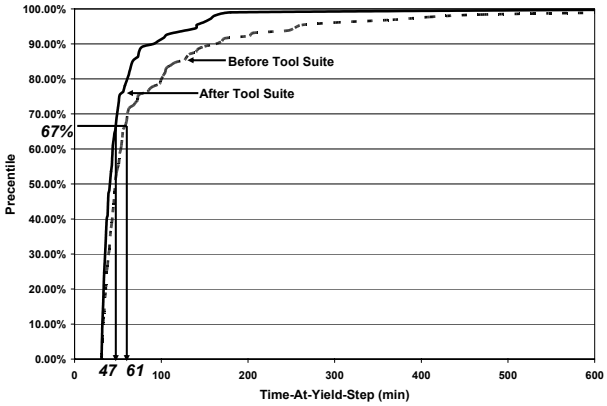

Figure 15(b): Time-at-yield-step; before and after Tool Suite

The development effort for the Tool Suite yield tools comprised of three weeks of familiarization for the developers, five weeks of coding the development prototype and seven weeks of developing and deploying rapid revisions of the Production prototype. At that point it met the functional needs and the user base was well trained, so the development team moved its focus away from functionality and started the job of making the product stable. During the time the prototype versions were released, the defect density also shows another downward trend. The correlation indicates that while yield problems are solved by applying a multitude of talents to solve the problems the right tools can facilitate the job and catalyze the results. Figure 15(b) shows the change in a more direct impact of the Tool Suite yield tools. We observed a general shift in the whole distribution of the time spent by lots for analysis at the yield step. The 67th percentile showed a shift of 23% towards faster processing of lots.

7. SUMMARY

In a semiconductor fabrication environment yield engineering efforts drive cost reduction and product pricing by maintaining high learning rates and delivering on predicted yield targets. Towards this end large amounts of "yield data" must be converted to actionable "yield information" from electrical test, fabrication history, product data and engineering analysis. The task of extracting, sorting, indexing and merging it from multiple silo software systems is laborious, time consuming and inefficient because it takes experts in various functional groups to produce a complete analysis. Industry standard software tools that collect and store fabrication data for use by a single domain or expert user group are necessary for running such operations, but true ROI on these tools comes through integrating their use and access through software systems that remove bottlenecks in the business process.

Figure 16 shows the results of implementing one such system called Tool Suite.

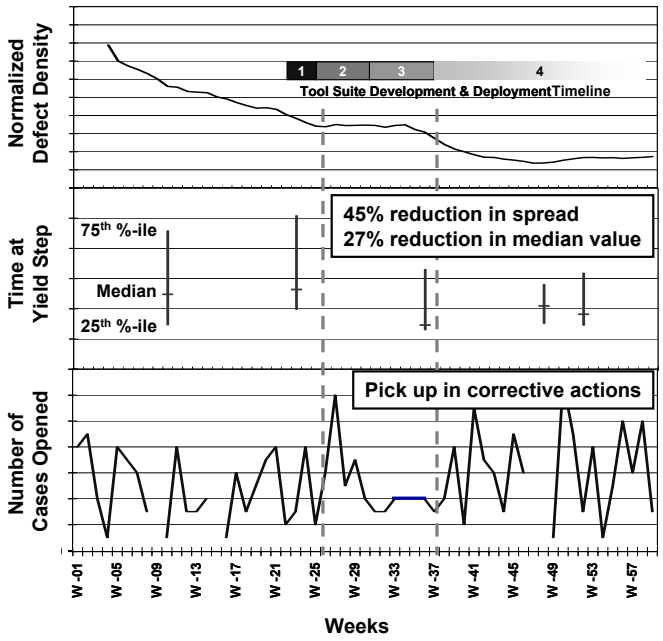

Figure 16: Tool Suite development timeline and corresponding trends of yield engineering activity indicators

The bar in the top plot shows the four stage development and deployment timeline. The top plot shows the corresponding decrease in defect density since Tool Suite went into use, the second plot shows the reduction in median and spread of time spent at yield analysis and the bottom plot shows the trend of another indicator of yield improvement activity, namely the number of corrective action cases. The trend shows an up-tick in the number of new cases opened that corresponds to the time Tool Suite usage gained momentum. This data linked system metrics to business metrics[6] and validated our postulate that tools that integrate the business process across multiple functional areas deliver ROI on the yield data analysis systems that a facility my already have in place.

8. AUTHORS

Manu Rehani is Manager for Strategic Business Development at LSI Logic Manufacturing, Inc. Prior to this role he led the Yield Data Systems group at LSI Logic Corp which was responsible for developing, deploying and sustaining systems that support Yield Data collection and Analysis. His publication and patent portfolio covers the topics of Data Translation to aid decision making and Rapid Software Development Methodologies based on Applied Prototyping. Rehani has been with LSI since 1997 and holds a masters degree in Chemical Engineering from Oregon State University.

Nathan Strader is the System Architect for Tool Suite and the Tool Suite development environment. Nathan practices and leads the implementation of the Applied Prototyping development methodology described in this paper and has worked in the capacity of a System Analyst at LSI Logic since the spring of 2001. During that time he has been the lead software developer on Defect, Yield, and E-Test Business Process Automation solutions for the LSI Logic Manufacturing Services facility. He holds a B.S. in Business and Management Information Systems from Oregon State University.

Jeff Hanson heads the Yield Management and Data Solutions organization at LSI Logic Manufacturing, Inc. Prior to this role he held positions as Sr. Yield Engineer at LSI Logic, Lead Engineer of the Material Analysis Lab at Hyundai Semiconductor, and as a Device Engineer at Motorola. Jeff has been at LSI since 1998 and holds a B.S. in Mechanical Engineering from the University of Texas at Austin.

REFERENCES

1. E. Goldratt, “The Goal,” *North River Press*, 1984
2. C. Weber, V. Sankaran, K.W. Tobin, Jr, G. Scher, "A Yield Management Strategy for Semiconductor Manufacturing Based on Information Theory" *Portland International Conference on Management of Engineering & Technology,* 1999
3. “Putting rules engines to work,” *Info World*, June 28 2004
4. R. Coram, “Boyd: The Fighter Pilot Who Changed the Art of War”, *Little, Brown & Company*, 2002
5. D. Abercrombie, M. Rehani, “"Challenging the Traditional Product Life Cycle Model in Delivering Software Technology Solutions,” *Portland International Conference on Management of Engineering & Technology,* 2003
6. T.S. Brown, “A New Metrics System for IT,” *CIO Magazine*, Dec 2004