Data Mining for Improving a Cleaning Process in the Semiconductor Industry

Dan Braha and Armin Shmilovici

***Abstract*—As device geometry continues to shrink, micro-contaminants have an increasingly negative impact on yield. By diminishing the contamination problem, semiconductor manufacturers will significantly improve the wafer yield. This paper presents a comprehensive and successful application of data mining methodologies to the refinement of a new dry cleaning technology that utilizes a laser beam for the removal of micro-contaminants.**

Experiments with three classification-based data mining methods (decision tree induction, neural networks, and composite classifiers) have been conducted. The composite classifier architecture has been shown to yield higher accuracy than the accuracy of each individual classifier on its own. The paper suggests that data mining methodologies may be particularly useful when data is scarce, and the various physical and chemical parameters that affect the process exhibit highly complex interactions. Another implication is that on-line monitoring of the cleaning process using data mining may be highly effective.

***Index Terms*—Composite classifiers, data mining, laser cleaning, machine learning.**

I. Introduction

DIE YIELD is defined as the ratio between the number of good chips and the total number of chips on a silicon wafer. It has been recognized that maintaining high yield levels is an important element to the competitiveness of semiconductor manufacturing companies [1].

The yield of a silicon wafer is affected by many factors. Some of these factors are microscopic particles, metallic contaminants, photoresist, and other organic residues which are located on the wafer. These micro-contaminants can damage some of the chips on the wafer during the fabrication process, thus reducing yield. As device geometry continues to shrink, these micro-contaminants have an increasingly negative impact on yield. By diminishing the contamination problem, semiconductor manufacturers will significantly improve the wafer yield.

Traditionally, yield improvement have been achieved through the use of statistical and experimental design techniques [1]. Other methods, such as yield modeling and simulation techniques [1], [24] also depend on elaborate statistical techniques. Unfortunately, there are practical limitation in automating the statistical techniques when complex interactions and nonlinearities are involved in the underlying models. Moreover, the large amount of data in current semiconductor databases make it almost impractical to manually analyze them for valuable decision-making information. This difficulty is mainly due to the large amount of records, which contain hundreds of attributes, that need to be simultaneously considered in order to accurately model the system's behavior. The need for understanding complex interaction effects, and automated analysis and discovery tools for extracting useful knowledge from huge amounts of raw data has led to the development of knowledge discovery in databases (KDD) and data mining methodologies [2]. Some researchers use the term KDD to denote the entire process of turning low-level data into high-level knowledge. The term data mining is considered a single step in the process that involves finding patterns in the data. Other researchers define data mining as the entire process of knowledge discovery [3]. In this paper, we adopt the second definition.

In this paper, we present a comprehensive and successful application of data mining methodologies to an emerging dry cleaning technology called advanced wafer cleaning. Advanced wafer cleaning is an innovative laser-based technology that utilizes the interaction between a laser beam and a micro-contaminant in a gaseous atmosphere for the removal of micro-contaminants [18]–[20].

While the advanced laser technology is innovative and successful in most aspects of wafer cleaning, the wafer cleaning process still needs some improving and refining. To this end, a series of experiments have been conducted in order to understand the laser cleaning mechanisms and identify the attributes that are significant in the cleaning process. Based on the data collected, three classification-based data mining methods (including decision tree induction, neural networks, and composite classifiers) have been employed, to enhance the understanding of the cleaning process. The purpose of data mining systems based on classification methods is to categorize the given data into a given predefined number of categorical classes and determine to which group a new data item belongs.

Several outcomes have been obtained: 1) building a series of *models* to represent the data set; 2) deducing the set of parameter values that will obtain the highest success rate of the various target functions (e.g., success in removing micro-contaminants); 3) identifying conditions under which additional particles are being formed as well as conditions under which wafers are being damaged during the cleaning process; and 4) demonstrating the usability of different data mining techniques, composite classifiers in particular, for handling realistic industrial

Manuscript received March 27, 2001. This work was supported by the consortium for control and management of manufacturing clusters in the semiconductor industry, as part of the MAGNET program of the Israeli Ministry of Trade and Industry.

D. Braha is with the Center for Innovation in Product Development, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: Braha@MIT.EDU).

A. Shmilovici is with the Department of Industrial Engineering and Management, Ben-Gurion University, Beer-Sheva 84105, Israel (e-mail: armin@bgumail.bgu.ac.il).

Publisher Item Identifier S 0894-6507(02)01027-8.

0894–6507/02$17.00 © 2002 IEEE

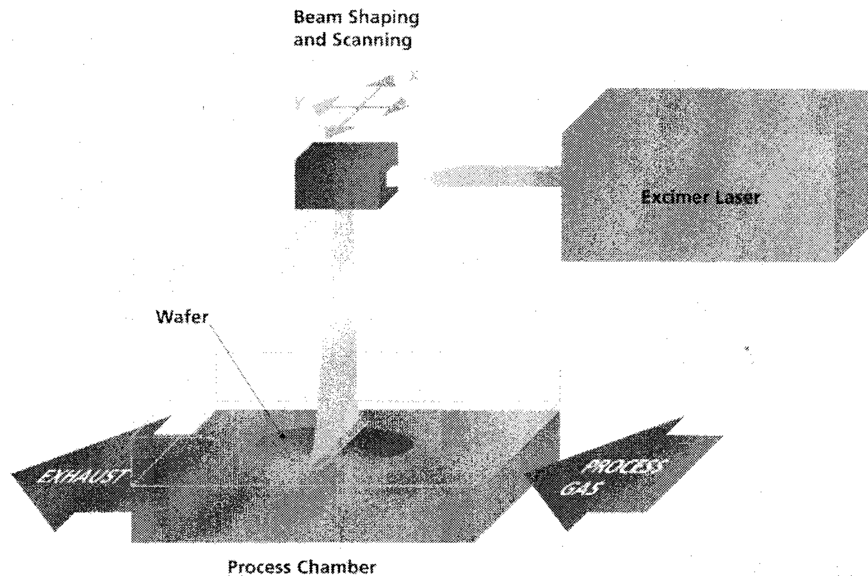

Fig. 1. The cleaning process.

setting where data is scarce. The paper suggests that data mining methodologies may be particularly useful when data is scarce and the various physical and chemical parameters that affect the process exhibit highly complex interactions. Another implication is that on-line monitoring of the cleaning process using data mining may be highly effective.

The paper is organized as follows. Section II explains the dry-cleaning technology for the contamination problem and the process that needs to be optimized. Section III describes the feature selection and experimental setting. Section IV provides a brief overview of knowledge discovery in databases and data mining. In Sections V–VII, the applicability of decision tree induction, neural networks, and composite classifiers to the cleaning problem is presented. Section VIII concludes the paper.

II. A Dry Cleaning Process for Micro-Contamination Control

Yield is one of the most important indices determining the success in the semiconductor manufacturing business. Micro-contaminants are small particles made of metals, photoresist, or other organic and nonorganic residues. Micro-contaminants can originate either from the atmosphere in the fab (hence the ultra-clean room requirement), the materials, or the tools that are used. As device geometries continue to shrink, contaminants have an increasingly negative impact on yield. Micro-contaminants on the masks or wafers can cause electrical short circuits between aluminum lines. They can also cause open circuits or breaks in aluminum traces. Either of these is fatal to the functionality of the chip. A "killer defect" is less than half the size of the device critical dimension, which means that for 0.18-μm technology, the cleaning process has to be efficient for particles less than 0.1 μm.

Submicrometer particle removal becomes harder because of the nature of adhesion forces as compared to removal forces. The main physical adhesion forces that operate between the submicrometer particles and the wafer surface are Van-der-Waals, capillary, and electrostatic forces. Although the relative amount of these forces depends on the nature and chemical composition of both the particle and the wafer surface, they are proportional to the particle diameter. The removal force, on the other hand, is proportional to the square or the cube of the diameter [18], [19]. Therefore, it is harder to overcome adhesion forces for smaller particles.

Contamination control is a major issue in VLSI fabrication. With each new device generation, the cleaning requirements become more stringent, and the number of cleaning steps increases. Today, up to 30% of the manufacturing time may be spent in cleaning processes. Presently, wet-chemistry cleaning using chemicals (such as ammonium hydroxide and hydrochloric acid) is used in conjunction with ultrasonic or megasonic in most wafer cleaning stages. There are a number of fundamental problems associated with wet-chemistry cleaning. 1) Nonuniform chemical reaction can cause the gate oxide to be quickly etched and damaged. 2) The cleaning liquid can become a contaminating source, since nondissolved foreign material remains in the liquid and can be re-deposited. 3) At deep submicrometer levels, the layer of the liquid chemical in contact with the wafer surface remains stationary and consequently is unreplenished. Megasonics is not always effective and could become destructive. 4) Most cleaning agents are highly toxic to humans and to the environment, so expensive facilities are needed for their handling and disposal. Thus, as wafer processing becomes more aggressive and the tolerance for contaminants reduced, traditional cleaning techniques based on wet-chemistry cleaning become less adequate. The trend toward all-dry processing can overcome potential drawbacks associated with wet chemistries.

A new laser-based wafer cleaning technology, called *Advanced Wafer Cleaning*[1] , has recently been developed [18]–[20]. Fig. 1 demonstrates the *Advanced Wafer Cleaning* process.

[1]It has been commercialized for single-step dry photoresist stripping, as well as for single-step full wafer cleaning [20].

A wafer is uniformly heated in a vacuum chamber, with a controlled mixture of O_2, O_3 (ozone) and NF_3 (nitrogen tri-fluoride). An excimer laser system together with a scanning optical system can generate very short and well-shaped light pulses that can accurately scan an area of 200×200 mm. The gases become highly reactive during the short laser pulse (about 30 nsec) and photoresist polymer bonds are broken under the laser spot. Once the particles are lifted off the surface, they are carried away by a flow of the process gas. As the laser beam scans the entire wafer, photoresist and embedded contaminants are removed. The reactive products are continuously pumped out of the process chamber through a catalytic converter making the process safe and environmentally friendly.

The physical mechanisms involved into the removal of micro-contaminants from the wafer surface can be roughly partitioned [18], [21] into three types:

1) *Shock mechanisms*—due to sudden heating and expansion of the substrate and the particle, explosive evaporation of the thin liquid film around the particle, and expansion of ionized process gases.
2) *Oxidation and laser induced chemical etching*—interaction of the particle with the reactive gas mixture that can cut chemical bonds and oxidize organic particle.
3) *Laser Etching*—the laser's photons can cut chemical bonding and reduce adhesion forces to the substrate.

The efficiency of the cleaning process is directly related to the speed of the photo-thermal process (i.e., the interaction between the laser beam and the reactive gases) which depends on the optical flow of the laser. The rapid expansion of the gases due to their heat absorption from the laser pulse introduces acceleration forces that act on the particle. Yet, excessive optical power may damage the wafer surface. An attempt was made to develop an analytical (first principles) model, which involved the solution of coupled nonhomogenous partial differential equations. Unfortunately, experimentation failed to confirm the predictions of the complex model.

Fig. 2 presents a micro-contaminants experimental map, before and after the cleaning process. More than 98% of the particles larger than 0.24 μm in diameter were removed. One of the challenges of the *Advanced Wafer Cleaning* technology is to optimize the laser cleaning process further, to particles with 0.1 μm diameter. One of the goals of the classification-based methods presented in Sections V–VII is to identify conditions under which a minimal 85%–90% cleaning rate may be realized for particles with 0.1 μm diameter.

III. Feature Selection and the Experimental Setting

The dry cleaning process, which is controlled by several attributes, has not yet reached the point where the exact values of each attribute needed to optimize the process are known. Moreover, mixing the wrong attribute values together can lead to added particles and even damage the wafer.

A series of experiments were executed in order to understand the laser cleaning mechanisms and identify the attributes that are significant in the cleaning process. Two types of wafer substrates were used in the experimental setting—bare silicon wafers and silicon-dioxide coated wafers—which represent the two mostly used substrates for cleaning. Each wafer was contaminated with about 1500 particles with a diameter of 0.1 μm. Before and after each firing/cleaning experiment, each contaminated wafer was imaged with a very accurate imaging system that counted the particles and targeted their location on the wafer for the laser firing. A microscope was used to validate the imaging results and to look for any possible damage to the wafer substrate in the targeted areas.

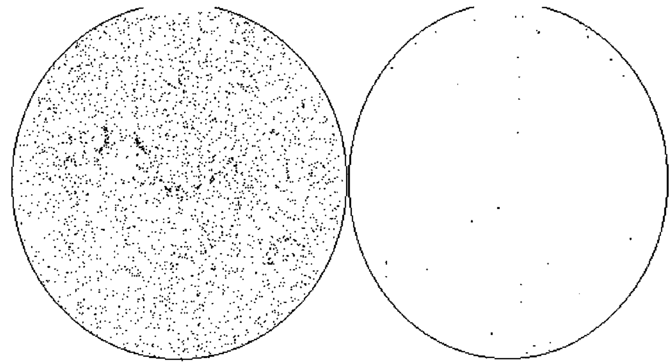

Fig. 2. Contamination map. Left: Before cleaning. Right: After cleaning.

There are many *input variables* (attributes or features), which are involved in the operation of the process. These input variables can be partitioned into two groups. The *energy factors* influence the energy of the laser pulse, and include: (1) the intensity of the laser, called Fluence (F); (2) the number of pulses in one cleaning session (N); (3) the number of cleaning passes over the wafer (n); (4) the angle between laser beam and the substrate surface (α); (5) the temperature of the preheated substrate (T); (6) the frequency of the pulse-train (f); and (7) the delay between subsequent pulse trains to the same spot (Delay). The *gaseous flow factors* include: (1) the pressure of combined gaseous atmosphere (P); (2) ozone flow (O_3); (3) oxygen flow (O_2); and (4) nitrogen tri-fluoride flow (NF_3).

There were two *output* (or *dependent*) *variables*: (1) Percentage of particles moved from their original location (%Moval), and (2) Percentage of particles removed from the target wafer (%Removal). Table I presents typical records generated by the experiments.

IV. The Data Mining Process

Data Mining is often defined as the process of extracting valid, previously unknown, comprehensible information from large databases in order to improve and optimize business decisions [2].

Data Mining techniques are at the core of the data mining process, and can have different goals depending on the intended outcome of the overall data mining process. Most data mining goals fall under the following main categories [5], [26]: data processing, verification, regression, classification, clustering, association, sequential pattern analysis, model visualization, and deviation analysis. A variety of techniques are available to enable the above goals. The most commonly used techniques can be categorized in the following groups [2], [5], [6], [26]: statistical methods, artificial neural networks, decision trees,

TABLE I
TYPICAL RECORDS * FROM THE CLEANING EXPERIMENTS

Removal (%)	**Moval (%)**	**Delay**	**f**	**T**	**(α)**	**n**	**N**	**F**	**NF_3**	**O_3**	**O_2**	**p**
60	67	0.15	15	5	9.0	4	3.0	5	7.0	16	5	30
85	89	0.15	15	5	9.0	4	1.0	8	2.0	24	5	30
46	59	0.15	15	5	9.0	4	5.0	2	2.0	24	5	30

Due to the commercial confidentiality of the process some of the data have been uniformly altered.

rule induction, case-based reasoning, Bayesian belief networks, and genetic algorithms and evolutionary programming. Several techniques with different goals can be applied successively to achieve a desired result. For example, in order to identify the attributes that are significant in a photolithography process, clustering can be used first to segment the wafer-test database into a given predefined number of categorical classes, then classification can be used to determine to which group a new data item belongs.

In Sections V–VII, we employ data mining methods which fall under the category of *classification* (e.g., neural networks, Bayesian networks, decision trees, and example-based learning [6]). We provide a brief description of the classification methods used and their utilization to the *Advanced Wafer Cleaning* process. For example, consider that each data item corresponding to a list of cleaning input variables (a parameter setting of the cleaning process) is classified to several categories according to the output variable of percentage of particles removed from the target wafer (%Removal). Given a historical data set of cleaning input variables and their %Removal classes, the classification method can identify the class of %Removal to which a *new* set of cleaning input variables is most likely to fit.

In practice, one class is more desirable than the others (e.g., the class with the highest %Removal). We would like to identify the operating conditions of the cleaning process that consistently derive that class. Since the classification is performed based on experimental data, it means that the model for that class (e.g., a decision tree) will take into considerations any physical constraints on the parameters (e.g., maximum pressure). In a practical factory situation, an on-line or off-line control procedure will use the classification model to determine the parameters of the cleaning process that generates the highest %Removal, and will tune the cleaning process accordingly.

V. Applying Decision Tree Induction to the Cleaning Process

A. Decision Trees

A decision tree is a tree-shaped structure that represents sets of decisions. Each nonterminal node represents a test or decision to be carried out on a single attribute value (i.e., input variable value) of the considered data item, with one branch and sub-tree for each possible outcome of the test. A decision tree can be used to classify a particular data item by starting at the root of the tree and moving through it until a terminal node (leaf) indicating a class is encountered. When a terminal node is reached, a decision is made. At each nonleaf decision node, the attribute (i.e., input variable) specified by the node is tested, which leads to the root of the sub-tree corresponding to the test's outcome. A decision tree can also be represented as a set of if-then rules[2] by creating one rule for each path from the root node to a leaf node.

ID3 and its successor C4.5 are two of the most widely used and practical methods for inductive inference, and have been successfully applied to a wide range of learning tasks [6], [7]. The C4.5 is a decision tree algorithm that uses training examples to construct a classification model, which describes the connection between classes and attributes. Once it has learned, the classification model can classify new, unknown instances. The C4.5 algorithm is best suited for problems with the following characteristics: 1) Instances are described by a fixed set of attributes and their values, where each attribute may have discrete values (for example *Hot, Mild, Cold*) as well as continuous values; 2) the target function has discrete output values; 3) the training data may contain errors, both in the classification of the training examples as well as errors in the attribute values that describe these examples; and 4) some training data may contain missing or unknown attribute values.

C4.5 employs a top-down, greedy construction of a decision tree. It begins with checking which attribute (input variable) should be tested at the root of the tree. Each instance attribute is evaluated using a statistical test (called *information gain*) to determine how well it classifies the training examples. The best attribute is selected and used as the test at the root node of the tree. A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant nodes. The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree. By placing the attributes with higher information gain closest to the root, the algorithm favors selecting shorter trees over longer ones. The algorithm can also backtrack to reconsider earlier choices by using a pruning method called *rule post-pruning* [6], [7]. The rule post-pruning method is utilized in order to overcome the overfitting problem, which often arises in learning tasks[3]. Overfitting occurs when the decision tree constructed classifies the training examples perfectly, but fails to accurately classify new unseen instances. It may occur when there is no noise in the data or when the number of training examples is too small to produce a representative sample of the true target function.

B. Implementation

While C4.5 can deal with continuous values for the different input variables (attributes), the class target function should

[2]This form is generally easier to understand than other representations (e.g., neural networks).

[3]Experiments have shown that overfitting decreases the accuracy of learned decision trees by 10%–25% on most problems.

have discrete values. To that end, the data corresponding to the two continuous output variables (i.e., %Moval, %Removal) have been discretized prior to applying the algorithm. Two methods have been employed to transform a continuous dependent variable into discrete classes. In the first method, human experts define discrete classes (detailed in Table II) based on their experience and intuition. In the second method, we have applied Kohonen's self-organizing maps (SOM) [8] to search for hidden clusters (i.e., "natural classes", detailed in Table V) in the input vectors. The SOM model is a two-layered neural network, where the output units are fully connected via weights to the inputs nodes. The input patterns are presented sequentially to the input layer[4], and a competitive learning rule is used to choose the "winner" (class) as the output unit with weight vector closets to the current input pattern. Subsequently, groups of output nodes are formed, which correspond to input patterns with similar characteristics.

Several experiments have been conducted. Each experiment consisted of 3 separate runs, where each run uses a different target function: %Moval, %Removal, or %Moval and %Removal together as a pair. Whenever the target function is defined by the pair $\langle$ %Moval, %Removal $\rangle$, the predefined categorical classes have been defined as follows: an instance is classified to a high-ranking class only if *both* %Removal and %Moval comply with the conditions of belonging to that class; otherwise, the instance is associated with a class according to the lowest rank of the elements in the pair.

Knowledge Representation: Decision tree induction is able to turn low-level data into high-level knowledge, which is represented in the form of a decision tree. The extracted knowledge can also be represented as a set of if-then rules by creating one rule for each path from the root node to a leaf node. Although both representations are equivalent, for illustrative purposes, we demonstrate both of them.

The decision trees are read from top-down. When there is a decision to be made, an indentation in the structure will be noticed. The tree employs a case's attribute values to map it to a leaf designating one of the classes. Every leaf of the tree is followed by "(n)" or "(n/m)" For instance, in the first tree displayed in Fig. 3, the last leaf of the decision tree is classified as class 4 (7/1), for which n is 7 and m is 1. The value of n is the number of cases in the data file that are mapped to this leaf, and m (if it appears) is the number out of them that are classified incorrectly by the leaf.

The equivalent representation as a set of if-then rules is described as follows. Consider the following rule in the rule set:

$$\text{Rule 1:} (13,\ \text{lift } 2.5) F <= 5 \text{ and } N <= 1.2 \text{ -> class } 1[0.933].$$

In the above representation several ingredients appear. 1) A rule number, which is arbitrary and serves only to identify the rule. 2) Statistics (n, lift x) or (n/m, lift x), which summarize the performance of the rule. More specifically, n is the number of training cases covered by the rule, and m (if it appears) shows how many out of them do not belong to the class predicted by the rule. The lift x is the estimated accuracy of the rule divided by the prior probability of the predicted class. 3) One or more conditions that must all be satisfied if the rule is to be applicable. 4) A class predicted by the rule. 5) A value between 0 and 1, which indicates the confidence with which this prediction is made.

[4]That is, a sequence of input node values forms an input pattern.

```
Decision tree:
:
      Alpha = 4.6: 1 (21/6)
      Alpha = 9.0:
      :...F <= 5.0:
          :...N <= 1.2: 1 (8)
          :   N > 1.2:
          :   :...O3 <= 8: 1 (6/2)
          :       O3  > 8: 2 (16/6)
          F  > 5.0:
          :...NF3  > 13: 1 (6/1)
              NF3 <= 13:
              :...Delay = 0.00: 3 (16/6)
                  Delay = 0.14: 4 (2)
                  Delay = 1.05: 2 (1)
                  Delay = 1.17: 2 (8/4)
                  Delay = 0.15:
                  :...N <= 1.2: 3 (4)
                      N  > 1.2: 4 (7/1)
:

Error: 29.6%
```

Fig. 3. Simplified decision tree for experiment 1 with target function %Moval.

When a rule set like the one described above is used to classify a case, it may happen that several of the rules are applicable (that is, all their conditions are satisfied). If the applicable rules predict different classes, there is an implicit conflict that could be resolved in two ways: 1) we can adopt the rule with the highest confidence or 2) we can attempt to aggregate the rules predictions to reach a verdict. The latter strategy is used in this paper; that is, each applicable rule votes for its predicted class with a voting weight equal to its confidence value, the votes are aggregated, and the class with the highest total vote is chosen as the final prediction. There is also a default class, here negative, that is used when none of the rules apply.

The Experimental Results: Four different experiments have been conducted (each consisted of 3 separate runs). In the first three experiments, the effect of the number of output classes on the accuracy has been investigated. Table II summarizes the results. The experiments show that as the number of classes of the underlying target function decreases, the error decreases and so does the dimension of the decision tree. Due to the complex interactions between classes in the parameter space, this result could not have been predicted. The best results have been achieved in Experiment 3 that employed only two classes. Naturally, we are mostly interested in the highest class, since it represents desirable operating conditions of the cleaning process.

A small portion of the decision tree generated for Experiment 1 (with target function %Moval) is given as follows.

As another example, Table III presents a portion of the decision trees generated for the third experiment. Every decision tree can be represented by an equivalent rule-base—which might be better suited for some applications. For example, Table IV presents the equivalent rule-base representation for the target function %Moval & %Removal (right column in Table III).

In Experiment 4, the Kohonen's Self Organizing Map (SOM) has been applied in order to find if there exists an "intrinsic"

TABLE II
SUMMARY OF RESULTS

	Experiment 1	Experiment 2	Experiment 3
	Class 1: 0% - 49% Class 2: 50% - 74% Class 3: 75% - 89% Class 4: 90% -100%	Class 1: 0% - 69% Class 2: 70% - 84% Class 3: 85% - 100%	Class 1: 0% - 84% Class 3: 85% - 100%
%Moval	Error: 29.6%	Error: 15.4%	Error: 7.7%
%Removal	Error: 17.3%	Error: 12.5%	Error: 6.6%
%Moval & %Removal	Error: 13.5%	Error: 12.5%	Error: 13.0%

TABLE III
A PORTION OF THE DECISION TREES GENERATED FOR EXPERIMENT 3

%Moval	%Removal	%Moval & %Removal
F <= 7.0: 1 (48/2) F > 7.0: :...Delay = 0.14: 3 (2) Delay = 0.15: 3 (10) Delay = 1.05: 1 (1) Delay = 1.17: 1 (17/1) Delay = 0.00: :...N <= 1.8: 1 (7/1) N > 1.8: 3 (19/4)	F <= 7.0: 1 (48) F > 7.0: :...Delay = 0.14: 1 (2/1) Delay = 0.15: 3 (10) Delay = 1.05: 1 (1) Delay = 1.17: 1 (17/1) Delay = 0.00: :...O3 > 23: 1 (7) O3 <= 23: :...F > 8.0: 1 (8/2) F <= 8.0: :...N <= 1.8: 1 (4/1) N > 1.8: 3 (9/2)	F <= 7.0: 1 (48) F > 7.0: :...Delay = 0.00: 1 (26/8) Delay = 0.14: 1 (2/1) Delay = 0.15: 3 (10) Delay = 1.05: 1 (1) Delay = 1.17: 1 (17)
Error: 7.7%	Error: 6.6%	Error: 8.7%

TABLE IV
RULES GENERATED FOR EXPERIMENT 3 (TARGET FUNCTION %MOVAL & %REMOVAL)

Rule 1: (48, lift 1.2)	F <= 7.0	-> class 1	[0.980]
Rule 2: (37, lift 1.2)	F>7.0 and Delay = 1.17	-> class 1	[0.974]
Rule 3: (35/8, lift 0.9)	F>7.0 and Delay = 0.00	-> class 1	[0.757]
Rule 4: (2, lift 0.9)	F>7.0 and Delay = 1.05	-> class 1	[0.750]
Rule 5: (4/1, lift 0.8)	F>7.0 and Delay = 0.14	-> class 1	[0.667]
Rule 6: (10, lift 5.0)	F>7.0 and Delay = 0.15	-> class 3	[0.917]
	Error: 8.7%		

clustering of the instances in the feature space. As neural networks can deal with continuous class values, the instances have been entered with their original success rates of %Moval and %Removal. The SOM network resulted in 3 categories. The highest-ranking class was class c and the lowest was class a. The instances have been classified according to those categories and run through C4.5 for classification. This produced an equivalent classification accuracy to that of experiment 3 (which also has three classes). This experiment validates the efficiency of using SOM for determining a "natural" number of classes when a human expert is not available. A portion of the generated decision tree is illustrated in Table V.

TABLE V
A SIMPLIFIED DECISION TREE GENERATED WHEN USING SOM

Decision Tree for 3 class clustering by Kohonen's Neural Network
F <= 7.0: a (47/4) F > 7.0: :...Delay = 0.14: c (2) Delay = 0.15: c (9) Delay = 1.05: a (1) Delay = 1.17: a (16/3) Delay = 0.00: :...NF3 <= 7.0: c (21/6) NF3 > 7.0: a (4)
Error: 13.0%

Conclusions: It should be noted that due to various extraneous factors, the underlying data set is of a relatively small volume. It is recognized that, in high dimensional spaces (as in the cleaning process), it may be difficult to obtain high classification rates with a small amount of data [2], [5], [13]. Despite this fact, the accuracy of the generated decision trees (see Table II) has been relatively good. The usefulness of the decision tree induction is further demonstrated by the fact that all the trees are centered around the same parameters and the same values, with somewhat difference in their combination. The re-

TABLE VI
2 CLASSES CLASSIFICATION

Network d	Network c	Network b	Network a	
150	150	60	60	No. of Neurons in hidden layer
150	50	150	50	No. of back-propagation iterations
91%/95%	91%/91%	93%/90%	90%/90%	Classification success %Removal/%Moval
78%/86%	82%/86%	78%/82%	78%/84%	Prediction success %Removal/%Moval (using cross validation testing)

sults indicate that full scale experiments or on-line collection of data (e.g., by sensory technology) will yield models that have high prediction accuracy.

Besides building a *model* to represent a data set, the generated decision trees have enabled: 1) deduction of the set of parameter values that will obtain the highest success rate of the various target functions and 2) identification of conditions under which additional particles are being formed as well as conditions under which wafers are being damaged during the cleaning process. In a practical factory situation, for example, we will use a feedback control mechanism to adjust the parameters of the cleaning process to the region indicated by rule 6 in Table IV.

VI. Applying Neural Networks to the Cleaning Process

A. Neural Networks

Neural networks are a class of systems modeled loosely after the human brain [5], [6], [25]. Biological learning systems are built of very complex webs of interconnected neurons. Artificial neural networks (ANNs) are built out of a densely interconnected set of simple processing units, simulating neurons. Each neuron is linked to certain of its neighbors with varying coefficients of connectivity that represent the strengths of these connections. Learning is accomplished by adjusting these strengths to cause the overall network to output appropriate results. ANN have been successfully used to solve complex problems such has voice recognition and face identification [25].

Designing a neural network consists of: 1) arranging neurons in various layers; 2) deciding the type of connections among neurons for different layers as well as among the neurons within a layer; 3) deciding the way a neuron receives input and produces output[5]; and 4) determining the strength of connections within the network by using a training algorithm to "learn" the appropriate values of connection weights.

A highly successful method for training multi-layer neural networks is called the backpropagation algorithm (a form of supervised learning, see [25]). The algorithm begins by constructing a network with the desired number of hidden and output units and initializing all network weights to small random values. Given this fixed network structure, the main loop of the algorithm then repeatedly iterates over the training examples. For each training example, it applies the network to the example, calculates the error[6] of the network outputs for this example, computes the gradient with respect to each component of the weight vector, and then updates all weights in the network. The gradient step is iterated (often thousands of times, using the same training examples multiple times) until the network performs acceptably well.

A variety of termination conditions is used to halt the procedure. One may choose to halt after a fixed number of iterations through the loop, or once the error on the training examples falls below some threshold, or once the error on a separate validation set of examples meets some criterion. The choice of termination criterion is important because too few iterations can fail to reduce error sufficiently, and too many can lead to overfitting the training data. The backpropagation algorithm is suited to the cleaning problem since it can handle discrete or continuous output vector, as well as handle training examples that contain noise or error.

B. Implementation

Neural networks can handle discrete or continuous output variables. However, since our main goal is to construct a classification model rather than exact prediction of the %Moval and %Removal values, the continuous dependent variables have been transformed into discrete classes.

In the first experiment, the output variables have been classified into two categories: 0%–74% (Class 0), 75%–100% (Class 1). The best performance has been obtained when using hyperbolic tangent functions for the hidden layer neurons, and logarithmic sigmoid functions for the output layer neurons. Several runs[7] have been performed with several network architectures (i.e., different numbers of hidden layer neurons) as summarized in Table VI.

In the second experiment, the output variables have been classified into four categories: 0%–49% (Class 1), 50%–74% (Class 2), 75%–89% (Class 3), and 90%–100% (Class 4). The best performance (73% accuracy) has been obtained when using hyperbolic tangent functions for the hidden layer neurons, and linear functions for the output layer neurons. As for decision tree induction, better prediction accuracy has been obtained whenever the number of classes of the underlying target function is small. The relatively small volume of the underlying data set may explain the sharp decline in performance whenever the number of classes increases. However, this result could not have been predicted in advance due to the complex parameter space and complex neural net architecture.

Observing that different neural network architectures perform better on some instances and worse on others suggests that an

[5]Several types of nonlinear transformations are used, in each neuron, as the basis for constructing multilayer networks.

[6]That is, the squared deviation between the network output values and the target values for these outputs.

[7]Each run has been repeated 3–5 times to check for consistency of the results.

increase in the classification accuracy may be obtained by combining several neural network classifiers together. This motivates us to investigate the composite classifier approach in Section VII, where a verification of the above intuition is provided.

VII. Applying Composite Classifiers to the Cleaning Process

A. Composite Classifiers

Combining the predictions of a set of classifiers, even simple ones, has been shown to be an effective way to create composite classifiers that are more accurate than any of the component classifiers on disparate domains as identification of satellite data [9], handwritten character recognition [10], economic and weather predictions [11], and protein sequence identification [12]. The past several years has witnessed a resurgence of research effort by the machine learning and pattern recognition communities to learn how to create and combine an ensemble of classifiers. The interest in classifier combination is heightened by the possibility that by intelligently combining a set of "simple" classifiers we may be able to perform classification better than with the sophisticated algorithms currently available [13]. Moreover, the accuracy of a sophisticated learning algorithm may be enhanced by combining it with 'simple' classifiers. For an excellent review of various composite classifiers design criteria and architectures, see [5], [13].

There are many architectures for combining classification algorithms of which the primary ones are [13]: stacked generalization [14], boosting [16], bagging [15], and recursive partitioning [17]. Composite training begins with training each component classifier (learning algorithm) separately (e.g., see Fig. 4), Then a combination classifier learns to output the correct prediction based on the predictions of the components. The combining algorithm can be a simple voting (the most common prediction is selected) or it can be a more sophisticated learning algorithms.

The fundamental question about classifier combination is whether classifiers from any given model class can be combined to create a composite classifier with *higher accuracy* [5]. Several strategies have been observed to achieve a higher accuracy of the composite classifier. The first strategy is to construct component classifiers that are highly accurate as independent classifiers [10]. The second strategy is to have composite classifiers that are diverse, not highly correlated, and behave very differently from one another [5], [6], [11]. Diversification strategies include: training the component classifiers on different samples of the training set; using classifiers from different model classes, (e.g., decision trees and neural networks, or neural networks with different architectures); making the classifiers applicable to separate regions of the space of instances; selecting component classifiers that apply different features; and varying the dependent parameters of a classifier. The third strategy is to avoid prohibitively expensive classifiers both in terms of number of component classifiers and the computational resources assigned for their training.

In the following, we briefly describe the two composite classifier architectures employed in this paper.

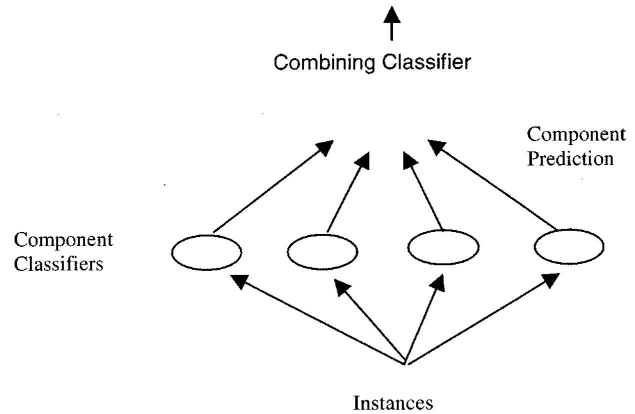

Fig. 4. Composite classifier architecture.

Stacked Generalization: Stacked generalization [14] assumes that we have a two-layer architecture in which the classifiers to be combined are called *level-0* classifiers, and the combining classifier is the level-1 classifier (the layering can be enlarged to more than two layers). The basic idea is that we have been given a set of n level-0 learning algorithms (components), a level-1 learning (combining) algorithm, and a training set of T_0 classified instances. The n level-0 learning algorithms should be distinct, so that diverse level-0 classifiers are obtained, for the reasons specified earlier. The algorithm has usually two phases [14], [22]:

Training Phase

1) The component classifiers are trained using leave-one-out cross validation as follows. For each instance in the data set, each of the n level-0 classifiers is trained using the remaining instances. Subsequently, the held-out instance is classified using each of the trained level-0 classifiers.
2) The level-1 classifier is trained with a set of T_0 vectors created as follows. Each level-1 training vector has $n+1$ elements corresponding to the concatenation of the predictions of each of the n level-0 component classifiers together with the actual class of that instance.
3) The level-0 classifiers are re-trained on the entire training set, since they have not been trained on the entire training set.

Application Phase: When presented with a new instance (whose class is unknown), the instance is classified using each of the level-0 classifiers. The predictions of the level-0 classifiers generate an input vector for the level-1 classifier. The generated vector is then classified by the level-1 classifier, which outputs the prediction for the new instance.

Boosting: Boosting [16] starts with a given component classifier, called the *base classifier*, C_1. Boosting is a randomized algorithm that filters the training set to train two additional component classifiers, C_2 and C_3. The first classifier C_1 is trained on a set of training examples. A fair coin is then tossed (randomly choose 0 or 1). If the coin comes up heads, then apply C_1 to new instances until C_1 makes a mistake. If the coin comes

TABLE VII
COMPOSITE CLASSIFIER OF C4.5 DECISION TREES FOR PREDICTING %REMOVAL, %MOVAL

Boosting Algorithm	**Component Classifiers**							
Majority Vote	*DT 7*	*DT 6*	*DT 5*	*DT 4*	*DT 3*	*DT 2*	*DT 1*	
85.6%	--	--	--	--	76.0%	67.3%	82.7%	**% Success %Removal**
79.8%	--	--	--	--	80.8%	72.1%	73.1%	**% Success %Moval**
83.7%	70.2%	77.9%	78.8%	73.1%	76.0%	67.3%	82.7%	**% Success %Removal**
80.8%	-	71.2%	75.0%	68.3%	80.8%	72.1%	73.1%	**% Success %Moval**

TABLE VIII
CHARACTERISTICS OF THE NEURAL NETWORKS USED IN STACKED GENERALIZATION-1

Network name	**N2**	**N5**	**Comb_1**	**Comb_2**	**Comb_3**	**Comb_3 After init**
# units in hidden layer	60	10	100	60	60	100
Number of iterations	50	50	50	50	50	50
Hidden layer function	Tansig	Tansig	Tansig	Tansig	Tansig	Tansig
Output layer function	Purelin	Logsig	Purelin	Purelin	Logsig	purelin

up tails, then apply C_1 to new instances until C_1 makes a correct classification. In either case, add the triggering (mistaken or correct) instance to a training set for C_2. This process of flipping a coin and adding a training instance is repeated until a sufficient number of examples have been collected to train C_2. Finally, new instances are drawn from a distribution of training instances, and are classified by C_1 and C_2. If they disagree the instance is added to a training set for C_3. When C_3 has been trained of a sufficient number of examples, the training stops. The composite classifier is applied by taking a majority vote of C_1, C_2 and C_3. An extension of this method (called Adaptive Boosting or *Adaboost* [16]), which can combine a large number of weak component classifiers, has been applied in this paper.

In Section VII-B, we apply the above techniques to the wafer cleaning process, and demonstrate their ability to enhance the prediction accuracy of weak component classifiers.

B. Implementation

First, the continuous dependent variables (i.e., %Removal, %Moval) have been transformed into four discrete classes as previously shown in Table II. Four experiments have been conducted. The first experiment runs a composite classifier, which is constructed by utilizing the boosting architecture. The other three experiments run composite classifiers, which are constructed by utilizing the stacked generalization architecture.

Adaptive Boosting: In this experiment, different decision trees have been constructed and combined by utilizing the adaptive boosting architecture. Two modes of adaptive boosting have been tested. In the first mode, adaptive boosting architectures that combine three decision trees (for %Removal) and three decision trees (for %Moval) have been generated. In the second mode, adaptive boosting architectures that combine seven decision trees (for %Removal) and six decision trees (for %Moval) have been generated.

The results of the composite classification are presented in Table VII. We observe that better results are obtained for the %Removal data than for the %Moval data. Adding decision trees 4–7 results in a reduction in accuracy for the %Removal data. In addition, as shown for %Moval in Table VII, combining the predictions of a set of classifiers *does not necessarily* create a composite classifier that is more accurate than any of the component classifiers. These rather surprising results may be attributed to the relatively small volume of data, the inferiority of the additional decision trees (e.g., DT4 is inferior to DT1) and the similarity (lack of diversity) between them.

Stacked Generalization-1: In this experiment, different neural networks (see Table VIII) have been combined by utilizing the stacked generalization architecture. The component classifiers have been trained by applying the backpropagation algorithm using the complete data set. The combining algorithm is also a neural network (see Table VIII). The results of the composite classifiers for %Removal and %Moval are summarized in the top section of Table IX.

It is observed that combining the predictions of a set of neural network classifiers creates composite classifiers that are more accurate than any of the component classifiers. In this case, each component classifier has "expertise" in some sub-region of the instance space; and thus combining their predictions performs better than individually.

Stacked Generalization-2: In this experiment, two different neural networks and a decision tree classifier have been combined by utilizing the stacked generalization architecture. The combining algorithm is a neural network. The results of the composite classifiers for %Removal and %Moval are summarized in the middle part of Table IX.

TABLE IX
DIFFERENT COMPOSITE CLASSIFIER EXPERIMENTS FOR PREDICTING %REMOVAL AND %MOVAL

Combining Algorithms						Component Classifiers								
Com_1+init	Com_1	Com_2+ init	Com_2	Com_3+ init	Com_3	N1	N2*	N3	N4	N5*	N6	C4.5 tree	Rec. An.	
--	--	--	76	78	79	--	72	61	70	73	63	--	--	**%Remov.**
--	--	--	--	--	74	69	66	57	--	--	--	--	--	**%Moval**
87.4	85. 4	--	87.4	--	--	--	72.8	--	--	55.3	--	82.5	--	**%Remov.**
79.6	80.6	--	80.6	--	--	--	70.9	--	--	58.3	--	72.8	--	**%Moval**
88.4	86.4	88.4	88.4	--	--	--	72.8	--	--	--	--	82.5	76.7	**%Remov.**
77.7	79.6	77.7	80.6	--	--	--	70.9	--	--	--	--	72.8	68.9	**%Moval**

* see characteristics of N2, N5 in table 8

The results for %Moval in the middle part of Table IX show a significant improvement in accuracy using the stacked generalization in comparison to the best results of each classifier individually. That is, three 0-level classifiers with accuracy of 58.3%, 70.9% and 72.8%, when applied individually, have resulted in accuracy of 80.6% when used in a stacked generalization architecture.

Stacked Generalization-3: In this experiment, a neural network, a decision tree, and a classifier based on the reconstructability algorithm have been combined by utilizing the stacked generalization architecture (due to space limitations we exclude the description of the reconstructability algorithm, see [23] for details). The combining algorithm is a neural network. The results of the composite classifiers for %Removal and %Moval are summarized in the bottom part of Table IX. As high as 15% improvement in prediction accuracy has been demonstrated in combining the three different algorithms. These results illustrate once more that an intelligent combination of known learning algorithms, even weak ones, can create a strong and accurate classifier.

Conclusions: The experiments primarily show that the accuracy of a classification algorithm (even if very strong on its own) can be increased, merely by combining its predictions with those made by another classifier (even if a weak one). The best results have been achieved in *Stacked Generalization-2* and *Stacked Generalization-3*, where different kinds of component classifiers have been combined. On the other hand, the boosting algorithm, which combines different decision trees, has displayed little or no improvement in accuracy. This may be explained as follows: A decision tree classifier performs best when trained with a large data set. Therefore, little advantage has been gained in combining decision trees that have been trained with relatively small volume of data. All four experiments have used different techniques in order to create component classifiers that are diverse. The results of these experiments show that the strategy of building a diverse set of classifiers by using classifiers from *different* model classes performs better than other strategies. Thus, by combining different kinds of component classifiers, high prediction rates may be obtained.

The results for %Removal data were better than for the %Moval data in all of the experiments. However, since the main concern is to *remove* the particles from the wafer and not merely to *move* them, the %Moval prediction accuracy is less cardinal than the %Removal prediction accuracy.

VIII. SUMMARY

Die yield is one of the most important elements for the competitiveness of semiconductor manufacturing companies. Since contamination can lead to drastic yield loss, better gains can be achieved by employing micro-contamination control and wafer cleaning processes.

In this paper, we have presented a comprehensive and successful application of data mining methodologies to an emerging new dry cleaning technology called *Advanced Wafer Cleaning*. The new dry cleaning technology utilizes the interaction between a laser beam and a wafer in a gaseous atmosphere for the removal of micro-contaminants. Three classification-based data mining methods have been employed. The composite classifier architecture has been shown to yield higher accuracy than the accuracy of each individual classifier on its own. This is particularly important for semiconductor manufacturing environments where various physical and chemical parameters that affect the process exhibit highly complex interactions, and data is scarce and costly for emerging new technologies. In addition, the component classifiers can use data sets of a relatively small volume, and need only reasonable amount of time and memory for training and application. Thus, a composite classifier may be highly effective when embedded in real-time monitoring of semiconductor manufacturing processes.

The large amount of yield data generated during daily semiconductor manufacturing operations makes it almost impractical to manually analyze the data for valuable decision-making information. In semiconductor manufacturing environments, this situation calls for new techniques and tools that can intelligently and (semi)automatically store and manage large amounts of yield data, which will be turned into high-level and useful knowledge. This knowledge extraction process should be supported by powerful data acquisition systems such as computers, microprocessors, transducers, and analog-to-digital converters for collecting, analyzing, and transferring data. The Integration of real-time data-mining methodologies with closed-loop process control may become a critical ingredient of future yield management, which will be integrated, agile, and capable of continuous prevention and improvement.

Beyond the above outcomes, it is possible that the classification-based methods like those presented here may enhance the accuracy of traditional yield modeling [1], [24]. It can relate the

underlying manufacturing process parameters to the number of particles existing on a wafer—which are related to the average defect density.

ACKNOWLEDGMENT

The authors would like to thank Oramir Semiconductor Equipment Ltd., for exposition of its dry cleaning process and for letting us use its experimental data sets.

REFERENCES

[1] S. P. Cunningham, C. J. Spanos, and K. Voros, "Semiconductor yield improvement: Results and best practices," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 103–109, May 1995.
[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery: An overview," in *Advances in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, S. P. Amith, and R. Uthurusamy, Eds. Cambridge, MA: MIT Press, 1996, pp. 1–36.
[3] S. S. Anand and A. G. Buchner, *Decision Support Using Data Mining*. New York: Pitman, 1997.
[4] R. Brachman and T. Anand, "The process of knowledge discovery in databases: A human-centered approach," in *Advances in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, S. P. Amith, and R. Uthurusamy, Eds. Cambridge, MA: MIT Press, 1996.
[5] M. J. Berry and G. Linoff, *Data Mining Techniques*. New York: Wiley, 1997.
[6] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[7] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
[8] T. Kohonen, "Self-organizing map," *Proc. IEEE*, vol. 78, pp. 1464–1480, Sept. 1990.
[9] J. A. Benediktsson, J. A. Sveinsson, P. H. Swain, and O. K. Ersoy, "Parallel consensual neural networks," in *Proc. 1993 IEEE Int. Conf. Neural Networks*, 1993, pp. 27–32.
[10] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 66–75, 1994.
[11] R. T. Clemen, "Combining forecasts: A review and annotated bibliography," *Int. J. Forecasting*, vol. 5, pp. 559–583, 1989.
[12] X. Zhang, J. P. Mersirov, and D. L. Waltz, "A hybrid system for protein secondary structure prediction," *J. Molecular Biol.*, vol. 225, pp. 1049–1063, 1992.
[13] D. B. Skalak, "Prototype Selection for Composite Nearest Neighbor Classifiers," Dept. Comput. Sci., Univ. of Massachusetts, Amherst, 95-74, 1995.
[14] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.
[15] L. Breiman, "Bagging Predictors," Dept. of Statistics, Univ. of California, Berkeley, 421, 1994.
[16] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, pp. 197–227, 1990.
[17] C. E. Brodley, "Recursive Automatic Algorithm Selection for Inductive Learning," Dep. of Comput. Sci., Univ. of Massachusetts, Amherst, 96-61, 1994.
[18] M. Genut, B. Livshits, Y. Uziel, O. Tehar-Zahav, and E. Iskevitch, "Laser removal of foreign materials from semiconductor wafers," in *Proc. SPIE*, vol. 3274, 1998, Paper 3274-19.
[19] B. Livshits, O. Tehar-Zahav, E. Iskevitch, and M. Genut, "Laser, dry and plasmaless, photoresist removal," *Solid State Technol.*, vol. 197, pp. 197–202, July 1997.
[20] ORAMIR Semiconductor Equipment LTD, Yokneam, Israel.
[21] A. C. Tam, W. P. Leung, W. Zapka, and W. Ziemlich, "Laser-cleaning techniques for removal of surface particulates," *J. Appl. Phys.*, vol. 71, no. 7, pp. 3515–3523, 1992.
[22] D. Wolpert, "Combining generalizers using partitions of the learning set," in *Lectures in Complex Systems*, L. Nadel and D. Stein, Eds. Reading, MA: Addison-Wesley, 1993.
[23] G. J. Klir, *Architecture of Systems Problem Solving*. New York: Plenum, 1985.
[24] W. Kuo and T. Kim, "An overview of manufacturing yield and reliability modeling for semiconductors products," *Proc. IEEE*, vol. 87, pp. 1329–1344, Aug. 1999.
[25] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley, 1991.
[26] D. Braha, Ed., *Data Mining for Design and Manufacturing: Methods and Applications*. Boston, MA: Kluwer Academic, 2001.

Dan Braha is currently at the MIT Center for Innovation in Product Development (CIPD) as a Visiting Professor. He has a position at the Department of Industrial Engineering at Ben-Gurion University, Israel, and is affiliated with the New England Complex Systems Institute in Cambridge, Massachusetts. Prior to that he was a Research Associate in the Department of Manufacturing Engineering at Boston University. One of his primary areas of research is engineering design and manufacturing. His research within engineering design focuses on developing methods to help the designer move from the conceptual phase to the realization of the physical device. He has developed a mathematical theory—the Formal Design Theory (FDT). He has published extensively, including a book on the foundations of engineering design with Kluwer Academic Publishers, and an edited book on data mining in design and manufacturing; also with Kluwer. He serves on the editorial board of AT EDAM, and was the editor of several special journal issues. He has also served on executive committees and as chair in several international conferences. His research interests include knowledge-based systems, case-based reasoning, cognitive science, artificial intelligence, computational complexity, information theory, product design and development, and operations research.

Armin Shmilovici received the M.Sc. degree in electronics engineering and the Ph.D. degree in industrial engineering in 1991 and 1997, respectively, from Tel-Aviv University, Tel-Aviv, Israel.

He joined the Department of Industrial Engineering and Management, Ben-Gurion University, Beer-Sheva, Israel, in 1998. His current research interests include automated reasoning from high dimensional systems, with techniques such as fuzzy logic, multiresolution analysis, and signal pursuits.