To appear in: *Proceedings of the 1993 American Society for Quality Control Measurement Quality Conference*

TESTING OF COORDINATE MEASURING SYSTEM SOFTWARE[†]

Cathleen Diaz and Theodore H. Hopp
National Institute of Standards and Technology
Gaithersburg, Maryland 20899-0001[‡]

ABSTRACT

Coordinate measuring systems (CMSs) determine dimensional (length-based) characteristics of mechanical parts by analyzing 3-dimensional point data acquired on part surfaces. Data analysis software can contribute significantly to the total measurement error of a CMS. Factors affecting software performance include the choice of analysis method, the quality of software implementation, and characteristics of the specific measurement task. However, there are no standard mechanisms to evaluate the effects of these factors on the quality of the measurement results.

NIST is developing a Special Test service to evaluate the performance of Gaussian curve- and surface-fitting algorithms that are at the core of most CMS software. This service is being defined in coordination with an ASME standards committee developing a new national standard, Performance Evaluation of Coordinate Measuring System Software. The NIST service will be based on this emerging standard, an Algorithm Testing System currently being developed at NIST, and test procedures specific to coordinate measurement tasks. This paper discusses the background, design philosophy, and status of the new service.

INTRODUCTION

Data analysis software has become increasingly important in modern dimensional measurement systems. This is particularly true of coordinate measurement systems (CMSs) such as vision systems, theodolites, photogrammetry, and coordinate measuring machines. Software computations to convert raw data to reported results can be a major source of error in a

---

[†] Contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

[‡] Factory Automation Systems Division, Metrology Building, Room A127

measurement system. Yet there are no standards or accepted methods for evaluating the effects of software on the overall uncertainty of measurements. This paper presents current work at NIST to develop an Algorithm Testing and Evaluation Program (ATEP) for testing such data analysis software. ATEP will be made available through a NIST Special Test service offered through the Office of Measurement Services and will be coordinated with emerging U.S. and international standards for performance of CMS software.

This paper is organized as follows. The next section provides the background and motivation for this work—what software is being tested and why testing is important. Following that is a discussion of the design philosophy and theory of software testing on which this work is based. Next is a description of how the Special Test service—which we have called the NIST Algorithm Testing and Evaluation Program, or ATEP—will work. The paper concludes with the current status of ATEP and the related standards work.

BACKGROUND AND MOTIVATION

The data analysis function with which we are concerned is geometric fitting. This function, illustrated in Figure 1, lies at the core of most inspection tasks. The role of geometric fitting is to reduce measured point coordinates to curve and surface parameters. The resulting curves and surfaces are called the *substitute geometry* for the feature. In further processing, these parameters are compared to the tolerance limits for the part. Most CMSs are characterized in terms of how accurately point measurements can be made. However, it is the uncertainty of the computed substitute geometry that determines the quality of a measurement. There are currently no standards or methods for estimating the uncertainties of computed feature parameters.

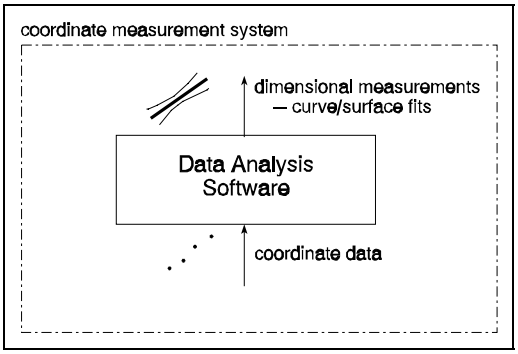

Figure 1 Geometric fitting software converts coordinate data to geometric parameters.

The phrase *computational metrology* refers to the study of the effects of data analysis computations on the performance of measurement systems. Computational metrology involves the application of core concepts of metrology[12] to the computational components of a measurement system. Not everyone shares the view that computational metrology is a significant area of study. For instance, an ASME standard on measurement uncertainty[1] states:

> Computations on raw data are done to produce output (data) in engineering units. Typical errors in this process stem from curve fits and computational resolution. These errors are often negligible.

The standard does deal with the *propagation* of errors through computations, but the above is the only mention of computations as a *source* of errors. Other practitioners, who accept that software can be a significant source of error, believe that software is not amenable to characterization in metrological terms.

During the last few years, much evidence has been discovered that data analysis can be a significant source of errors in CMSs. Estler[5] analyzed a measurement device for inspecting the casings of the solid rocket boosters for the NASA space shuttle. He reported that the data analysis software was the single largest source of error in the entire system. In the mid-1980s, Germany began a program of testing coordinate measuring machine software, with the express purpose of improving what was perceived as low quality of commercial fitting algorithms.[10] And in 1988, Walker issued an advisory[15] in which he reported the results of experiments with commercial inspection systems:

> Certain algorithms . . . are capable of stating that the measurement is worse than the actual data gathered up to an error of 37% and that the measurement is better than the actual data gathered up to an error of 50%.

There is considerable ongoing research on CMS algorithms. (See reference 6 for a survey.) There is also growing interest by government standards laboratories in testing the performance of CMS software, particularly in Great Britain[2,3] and Germany[14]. Both countries offer services to test CMS software by comparing results for test data sets to results obtained from reference software. In the U.S., the NIST ATEP is intended to provide a similar service[4]. In the next section, we present the design philosophy for the ATEP, which differs from other proposed approaches to software testing.

DESIGN PHILOSOPHY

The end goal of testing is to be able to evaluate how well software will perform in specific applications, or at least to be assured of a particular level of performance. The ATEP is being developed with a three-step process for achieving this goal. The first step is to identify the performance factors—what conditions or influences affect the quantities of interest. The second step is to hypothesize a performance model that relates the performance factors to levels of performance. The third step is to develop tests by which the parameters of the model can be evaluated. The model, when evaluated for a particular software package through testing, can then be used to estimate measurement uncertainties of computed quantities. We will first discuss the performance factors and models. We then discuss the design of tests.

Performance Factors and Models

Geometric fitting can be viewed as an optimization problem: find the parameters of substitute geometry that optimize a particular fitting objective. Under this view, performance factors for geometric fitting software fall into two broad categories: the choice of fitting objective and the implementation of that objective in software. We discuss each of these in turn.

Choice of Fitting Objective

The fitting process is illustrated in Figure 2. The choice of fitting objective is an important determinant of CMS performance. Part tolerances are generally interpreted in terms of extremal fits. That is, the objective is to find the geometry that fits the extremes of the part: the largest inscribed, smallest circumscribed, or minimum separation geometry. Also, simulation of

functional gages (sometimes called soft functional gaging) can be interpreted as finding the maximum clearance (or minimum interference) solid model fit to the part. Measurement practice, on the other hand, most commonly involves averaging fits, typically orthogonal distance regression† but also objectives such as least median of squares[9,11] and other robust techniques for outlier detection. When the fitting objective is anything other than the extremal fit of tolerance theory, the computed fits will be biased. That is, in the limit as the point density goes to infinity and the measurement error for each point goes to zero, averaging fits will be different from extremal fits.‡

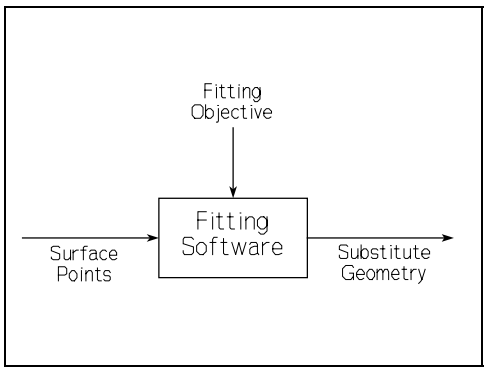

Figure 2 Ideal model of fitting.

The points being analyzed are not free of error; each point has an uncertainty of measurement. Also, the points are often not very dense on the part, so there can be significant sampling noise. (That is, the points may not accurately represent the part surface.) Averaging fits appear to be less sensitive to measurement errors than are extremal fits[13].

It is very difficult to develop general guidelines for the best choice of fitting objective for a practical CMS. There has been considerable debate over the widespread use of orthogonal least-squares fitting, with many claiming that extremal fitting is better because it "conforms" to tolerance theory. The truth is that the best choice of fitting objective is that which produces the smallest combined uncertainty in the result, and the best choice is not at all clear.

We can modify the ideal model of fitting of Figure 2 to include bias and sensitivity effects. The new model, drawn in Figure 3, shows that the observable values—the data points being fit and the reported fit—are combinations of inherently unobservable quantities. The observed data points always include some element of measurement error. The reported fit is a combination of the "true fit" (the fit to the surface) and two additional errors: the bias introduced by the choice of fitting objective and the sensitivity of the fit to the measurement error. Bias is the deviation between the mathematical fitting objective and the requirements of tolerance theory. Sensitivity is the effects of point measurement errors on the reported fit.

Objective Implementation

The second major category of performance factors deal with implementation. There are four factors to consider: 1) the optimization method chosen to compute the fit; 2) the computing environment (word length, rounding method, dynamic range, etc.); 3) handling of extreme cases; and 4) code correctness.

---

† Orthogonal distance regression is also called Gaussian fitting and, for lines and planes, total least squares. In the metrology community, it is often—but inaccurately—called least squares.

‡ Depending on one's viewpoint, bias might be considered to be an error in the model. We reserve the term *model error*, however, to indicate the error resulting from picking a particular geometric form to fit to the data.

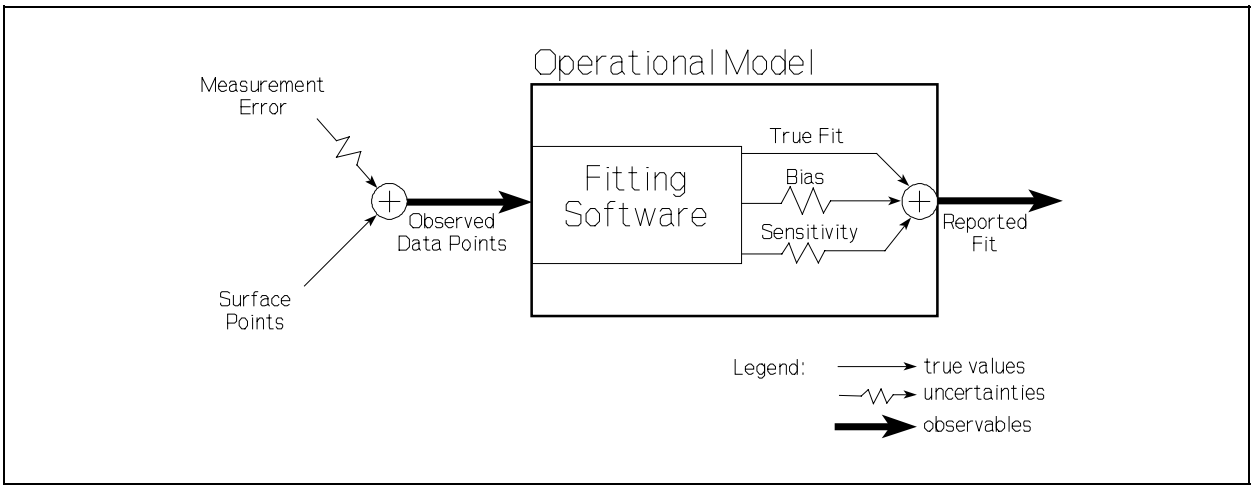

Figure 3 Operational model of fitting software, showing bias and sensitivity errors.

The choice of optimization method greatly affects how the computing environment impacts measurement uncertainty. In particular, the effects of rounding due to machine precision can be much greater for some optimization methods than for others, even for the same objective function. Analyses of rounding errors are often done in terms of a quantity called the *unit roundoff error*, which we shall denote by **u**. **u** is a property of the computing environment; it is the largest number that, when added to 1 in floating point arithmetic, produces an answer of 1. (Typically **u** is about 10^{-7} for single precision and about 10^{-15} for double precision.) Some iterative algorithms rely on convergence tests to end. The effects of the convergence tests can sometimes be modeled as an increase in the value of **u** for the computing environment.

The study of how optimization methods and rounding errors interact can draw on an extensive body of results in linear algebra perturbation theory. Although fitting problems are often non-linear in nature, most optimization algorithms work by repeatedly solving a linear approximation to the fitting objective. So the results of linear algebra can potentially be applied to a much broader class of optimization methods than might be apparent. All results that follow are based on linear algebra theory found in reference 7.

One example of how the choice of optimization method can dramatically affect the impact of rounding errors is simple linear regression. Suppose we wish to compute the regression line to the three points (-100,-100), (0,0), and (100,100). Clearly, the regression line has a slope of 1 and a y-intercept of zero. Denote the regression line by *v*. Then the fitting objective is to find the smallest magnitude *v* that minimizes the sum of squares of residuals. This can be modeled as solving for *v* in the matrix equation $Av \approx y$, where

$$A = \begin{pmatrix} -100 & 1 \\ 0 & 1 \\ 100 & 1 \end{pmatrix}$$

and $y=(-100, 0, 100)^T$. Generally, the computed value of *v* (by any method) will be different from the exact solution v_{LS}. The most common solution method is the use of the *normal*

equations for the model. The normal equations are obtained by multiplying the original matrix equation on the left by A^T and solving for v using the Choleski decomposition of A^TA. The result is $v=(A^TA)^{-1}A^Ty$. If v is obtained by evaluating the normal equations with roundoff error **u**, then the error of the solution is $\|v-v_{LS}\| \approx 6{,}700\mathbf{u}$. This is in contrast to a more accurate method, described next.

The same objective can be evaluated using *singular value decomposition*. Matrix A can be decomposed into the product $U^T\Sigma V$, where U and V are orthogonal matrices and Σ is a diagonal matrix. Let Σ^+ be the diagonal matrix obtained from Σ by setting each element of Σ^+ to zero if the corresponding element of Σ is zero, or else to the inverse of the corresponding element of Σ. Then $v=V\Sigma^+U^T$ is the least-squares solution. If v is obtained by evaluating the singular value decomposition with roundoff error **u**, then the error of the solution is $\|v-v_{LS}\| \approx 82\mathbf{u}$. Thus, for this very simple problem, the effects of rounding can be changed by nearly two orders of magnitude by the choice of optimization method.

The above is an example of *forward error analysis*. It says that the computed fit is the exact fit plus some error. This is illustrated graphically in Figure 4. General results for forward error

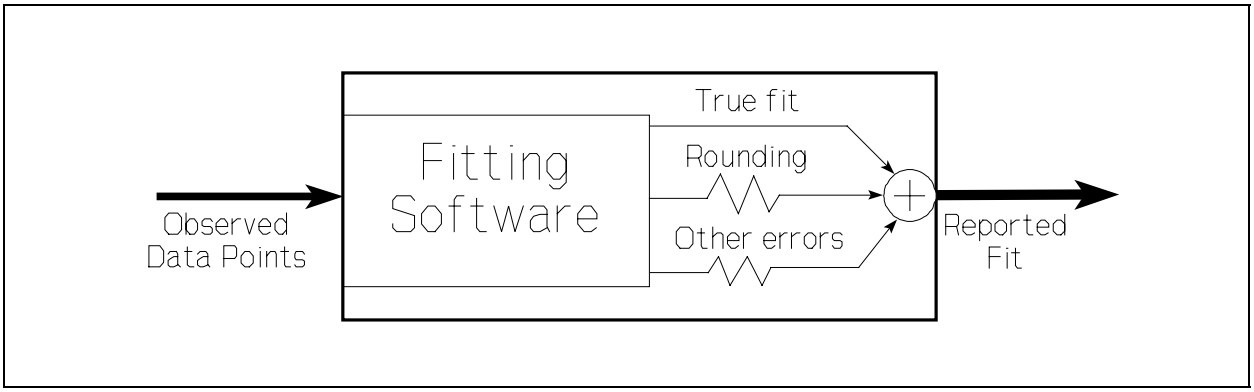

Figure 4 The forward error analysis model. The other errors include bias and sensitivity.

analysis can only be obtained for some algorithms. Forward error analysis has the additional shortcoming that the relative error can usually only be characterized in terms of the same quantities that are used to compute the fit. Thus, it is very difficult to write software for estimating the fit uncertainty that will work reliably in the same computing environment that is being used to calculate the fit.

A different approach, called *inverse error analysis*, is to view the computed solution as the exact solution to a "nearby" problem. Consider again orthogonal distance regression computed by the singular value decomposition of the data matrix D. The computed fit is the exact fit to some other data set $D+\Delta$. Inverse error analysis is illustrated in Figure 5. The effects of rounding are modeled as adding additional noise to the data *before* the fit is computed. The computation, however, is then viewed as being done in exact arithmetic. It is often much easier to characterize the effects of rounding in terms of inverse error analysis. The effects of the inverse error on the reported fit are determined by the sensitivity of the objective function.

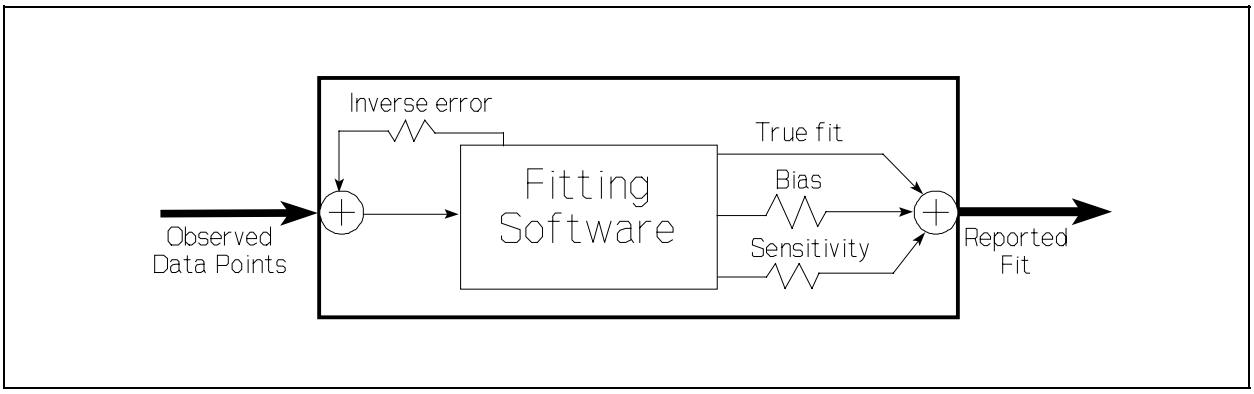

Figure 5 The inverse error analysis model.

A comprehensive test will check the behavior of the software for extreme cases. These cases fall into three categories: pathologies, degeneracies, and extreme values. Pathologies are data sets for which no solution is possible. These include data sets with too few points and problems with unbounded solutions (such as fitting a circle to collinear points). Degeneracies are configurations that should pose no problem but often do in practice. These include data sets with zero residuals (perfect geometry), "vertical" slopes, and similar configurations. Extreme values include number of points (minimum required; maximum for the implementation), distance from the origin, narrow geometry bounds (small arcs, etc.), and large residuals.

The last implementation factor, code correctness, is difficult to evaluate. Code errors can only be found if a test case exercises the relevant code. This is done in the ATEP by comparing test fits to corresponding reference fits. This, in fact, is a primary purpose of computing reference fits. However, it is essentially impossible to design tests that will find all code errors in a black box. The best one can do is to generate data that are representative of the intended application.

Test Design

Our test design approach is based on three assumptions. One assumption is that tests should be designed and interpreted according to a well-defined error model. The models presented above identify four error sources: bias, sensitivity, rounding error in the fit, and induced error in the data from rounding. Two other sources of error, handling of extreme cases and code correctness, will be discussed below. These are the error sources for which the ATEP is being designed. A second assumption is that test results must be interpreted and reported in terms directly related to inspection tasks. Part tolerances generally require conformance of the part to tolerance zones. These zones are volumes or areas for which the part features or substitute geometry elements must observe certain constraints. This means that test results should quantify the uncertainties of geometric relationships computed by the software. Thus, representation-specific test results (e.g., parameter values) should be avoided[†]. A third assumption is that the software to be tested is a "black box." That is, the internal structure of the software—the optimization method, code structure, computing environment, etc.—is unobservable. Thus, the testing method is limited to

[†] This is not to say that parametric representations cannot be used in obtaining the test results.

supplying the software with fitting problems and analyzing the fit results. The problem set and analysis are designed to identify the elements of the error models.

The assumption of black-box testing leads to the now-common architecture of a testing system, shown in Figure 6. The system consists of three components: a data generator, a set of reference

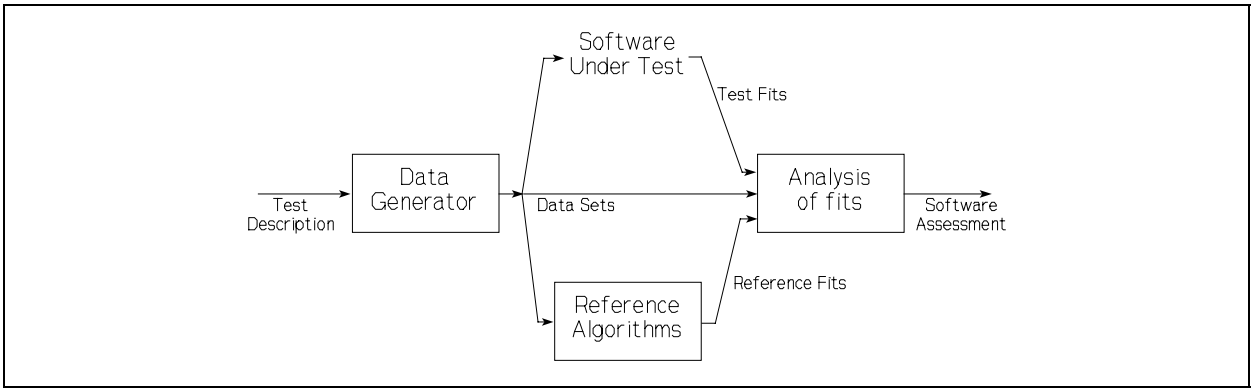

Figure 6 Architecture of the NIST Software Testing System.

algorithms, and fit analysis. The data generator is driven by a test description—a high-level specification of the tests to be performed. A test description may define hundreds of data sets; these would be generated automatically from the description. The data sets are processed by the software under test to generate test fits. At the same time, the data sets may be processed by fitting software supplied with the testing system. The data sets, test fits, and reference fits are used to assess performance measures for the software under test.

A data set is generated from four classes of information: the nominal (ideal) geometry of the feature, the form error(s) of the feature being simulated; the sampling plan (distribution of points on the feature) for the data set; and the (random) measurement error distribution for the points. A test description consists of ranges of instance values for each of these information classes. This provides the flexibility to study the behavior of the software in a controlled manner. For instance, a test description that varies the simulated measurement error alone can be used to assess the sensitivity errors of the software for the particular combination of other factors. Varying the sampling plan for a fixed simulated form error allows the study of the effects of sampling noise on the fit result. Similar variation of other factors can be used to study other aspects of the error model.

As mentioned before, fits are analyzed in terms of geometric differences among fits, not parameter values. In our system, all fit geometries are trimmed by the projection of the data onto the fit geometry. Comparison of test fits to reference fits is used to detect code errors and to identify systematic bias. Bias can arise when an implementation optimizes an approximation to the stated fitting objective. (Commercial software sometimes uses linear approximations in place of non-linear objectives to gain a speed advantage.)

The main tool for evaluating software is measures of variation in fit geometry in response to varying test conditions. These measures are independent of reference fits. A different analysis

is used for each type of geometry. For instance, cylinder fits are analyzed as follows. The axis of each cylinder is trimmed by the orthogonal projection of its corresponding data set onto the axis.[†] The smallest cylinder that encloses the axis segments is then computed. The diameter of this enclosing cylinder is a measure of dispersion for tolerances such as position, orientation, and parallelism. Depending on which parameters of the test description are involved, this dispersion is an estimate of sensitivity, bias, or effects of rounding errors. Similarly, the spread of radii of the cylinder fits measures dispersion for size. The spread of angles between the axes is measured as the angle of the narrowest cone that encloses all the direction vectors of the axes. This measure does not relate directly to common tolerance applications, but can be used for diagnostic purposes.

Comparison of fits to a reference is done on a data set-by-data set basis. (That is, a different reference is used for each data set.) Differences between test fits and the reference for each data set are computed in a way similar to that described above. Thus, for instance, the maximum deviation of axis segments from the reference axis measures axis location differences. Difference measures can be accumulated across data sets to study the sensitivity of the differences to test parameters. A detailed description of analysis methods used at NIST will be reported elsewhere[8].

THE ALGORITHM TESTING AND EVALUATION PROGRAM (ATEP)

This section describes ATEP, the NIST service currently under development to provide tests of CMS fitting software used in part inspection. The goals of ATEP are: first, to provide a mechanism for evaluating CMS software; second, to reduce measurement uncertainties associated with CMS software; and third, to respond to U.S. industry needs for software testing. The intended customers of ATEP include: vendors of CMSs such as vision systems, coordinate measuring machines, theodolites, and photogrammetry; CMS users for applications such as mechanical parts, electronics, geodesy, and anthropometry; and calibration services.

The organization of ATEP is shown in Figure 7. A customer desiring a software test arranges to receive test data from NIST on a computer disk. The test data may be a standard suite of data sets or may be a suite designed for a specific application. The customer then returns fit results to NIST for each data set. (The contents of the data sets and the format of fit results are defined in documentation accompanying the data.) Following standard test procedures, NIST uses its own reference algorithms and the testing system described earlier to prepare an evaluation of the software. The evaluation will report the analysis results in terms of error model parameters. It will also characterize the ability of the test software to handle extreme cases, and, through comparison to reference algorithm results, will report on code correctness.

One of the key features of the ATEP is that it is being coordinated with emerging U.S. and international standards. Within the U.S., relevant standards are being developed by the American Society of Mechanical Engineers in three areas: tolerancing theory, measurement methods, and CMS software performance evaluation. The designation of these standards is:

[†] Fits from more than one data set may be analyzed together.

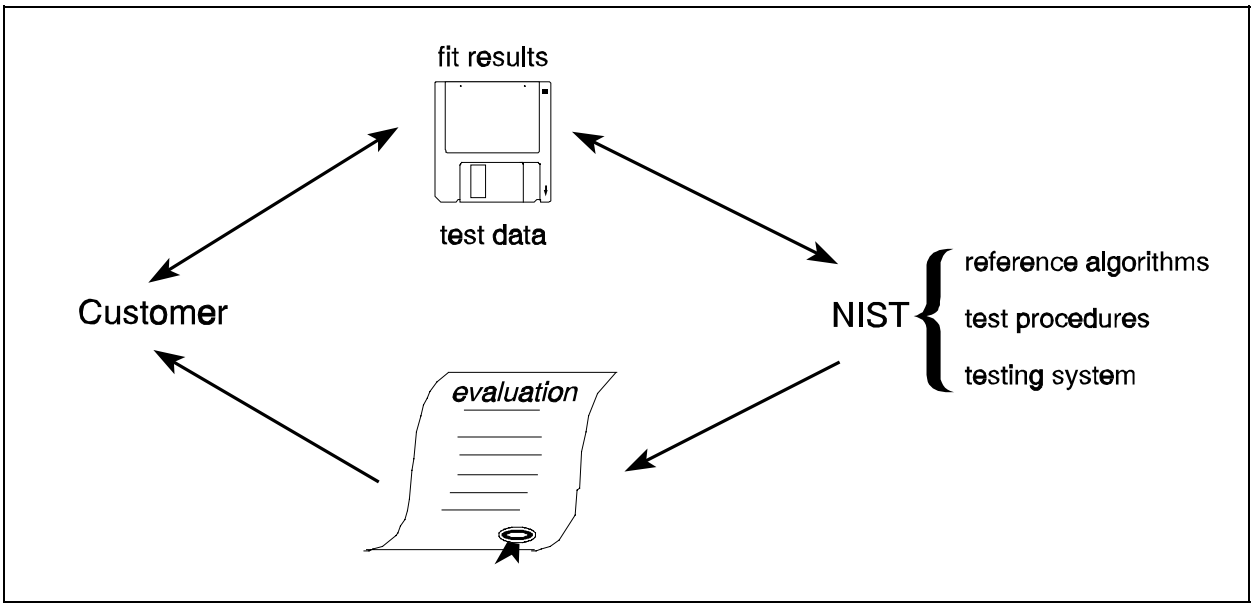

Figure 7 Organization of the NIST Algorithm Testing and Evaluation Program (ATEP)

ASME Committee	Standard Under Development
Y14.5.1	Mathematical Principles of Dimensions and Tolerances
B89.3.2	Dimensional Measurement Methods
B89.4.10	CMS Software Performance Evaluation

The standard most closely related to the ATEP is ASME B89.4.10. This standard is being developed in conjunction with NIST work. The ATEP test procedures, by which NIST evaluates fit results, will be based on this standard.

STATUS

As mentioned earlier, the ATEP is currently under development. Current plans are for it to be available in 1994. It will be catalogued as a Special Test Service under NIST's Calibration Program. The testing system used within the ATEP is also currently under development. Initially, the ATEP will support testing of orthogonal distance regression software for simple geometries (line, circle, plane, sphere, cylinder, cone, and torus). Future plans will include support of other fitting objectives, more complex geometries, and higher-level functions (such as tolerance verification) of inspection systems.

An earlier version of the testing system has been in use for over a year by members of the standards community and other interested parties. The purpose of distributing an earlier version of the system was to provide independent evaluation of the reference algorithms embedded in the system. These algorithms have performed well. They have been used to identify problems in third-party fitting software and have even been incorporated into commercial products.

The International Organization for Standardization, under its working group ISO/TC 3/WG 10 is also working on a standard for software testing. The ATEP program will support this standard as it is developed. There is major technical difference between the U.S. and ISO efforts. The U.S. standard envisions reporting fit results in terms of fit parameters. Under the current ISO proposal, the fit itself will not be reported, instead the customer will report the residuals corresponding to the points in the data set. Furthermore, under the ISO proposal fit results will be compared on the basis of differences between residuals computed by the software under test and corresponding residuals computed by reference software. It is unclear how differences in residuals relate to the fit uncertainty concepts that lie at the core of the U.S. approach. NIST and the U.S. delegation are working with the ISO group to resolve this issue.

The ATEP will be the first calibration service in the U.S. for dimensional metrology software. NIST has been working with standards groups and other interested parties to develop the principles and operation of this service. Any suggestions on how ATEP should work or on future directions are most welcome and should be directed to either of the authors.

REFERENCES

1 ASME, *Measurement Uncertainty*, ANSI/ASME Standard PTC 19.1-1985, American Society of Mechanical Engineers, New York, 1985.

2 Cox, Maurice G., "Improving CMM Software Quality," NPL Report DITC 194/92, National Physical Laboratory, Middlesex, U.K., 1992.

3 Cox, Maurice G., and Forbes, Alistair B., "Strategies for Testing Form Assessment Software," NPL Report DITC 211/92, National Physical Laboratory, Middlesex, U.K., 1992.

4 Diaz, Cathleen, "Concept for an Algorithm Testing and Evaluation Program at NIST," NISTIR 5366, National Institute of Standards and Technology, Gaithersburg, MD., January 1994.

5 Estler, Tyler, "Accuracy Analysis of the Space Shuttle Solid Rocket Motor Profile Measuring Device," NISTIR-89/4171, National Institute of Standards and Technology, Gaithersburg, MD, 1989.

6 Feng, Shaw C., and Hopp, Theodore H., "A Review of Current Geometric Tolerancing Theories and Inspection Data Analysis Algorithms," NISTIR 4509, National Institute of Standards and Technology, Gaithersburg, MD, 1991.

7 Golub, Gene H., and Van Loan, Charles F., *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1985.

8 Hopp, Theodore H., "Requirements for a Production Algorithm Testing System,", National Institute of Standards and Technology, Gaithersburg, MD, *in preparation*.

9 Mamileti, L., Wang, C.M., Young, M., and Vecchia, D.F., “Optical Fiber Geometry by Gray Scale Analysis with Robust Regression,” *Applied Optics*, Vol. 31, 1992, pp. 4182-4185.

10 Porta, C., and Wäldele, F., “Testing of Tree Coordinate Measuring Machine Evaluation Algorithms,” Technical Report BCR EUR 10909 EN, Commission of the European Communities, Luxembourg, 1986.

11 Rousseeuw, P., “Least Median of Squares Regression,” *J. Amer. Stat. Assoc.*, Vol. 79, 1984, pp. 871-880.

12 Taylor, Barry N. and Kuyatt, Chris E., “Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results,” NIST Technical Note 1297, National Institute of Standards and Technology, Gaithersburg, MD, January, 1993.

13 Vecchia, Dominic F., Wang, C.M., and Young, M., “Outlier-Resistant Fitting of Gray Scale Images Illustrated by Optical Fiber Geometry,” Proc. 1993 Meas. Sc. Conf., Los Angeles, 1993, Jan. 21-22.

14 Wäldele, Franz, Bittner, B., Busch, K., Drieschner, R., and Elligsen, R., “Testing of Coordinate Measuring Machine Software,” *Precision Engineering*, Vol. 15, 1993, pp. 121-123.

15 Walker, Richard, *GIDEP Alert No. X1-A-88-01*, Government-Industry Data Exchange Program, Washington, DC, 1988.