Measurement in Everyday Language

Scott E. Donaldson and Stanley G. Siegel
SAIC

Measurement for measurement's sake is a waste of time and money. Measurements need to be expressed in everyday terms that are familiar to the organization; otherwise, they may be of little value. We present a measurement technique that enables you to measure software products (which can be extended to software systems development processes) in everyday terms familiar—and therefore meaningful—to your organization. This article presents a software product measurement concept that we label product integrity. We illustrate how to measure a software product in terms of attributes and attribute value scales. We briefly describe the application of the measurement approach in the real world.

A developer of software or software-related products wants to stay in business—which is strongly tied to customer satisfaction. Customer satisfaction can be expressed in many ways. First and most important, a product should do what the customer wants it to do. In addition, when a customer pays a developer to develop software products, the customer wants these products to be delivered according to some established schedule and for some established amount of money. In this article, we fold considerations such as these into our concept of "product goodness."

For us, then, product goodness is a multidimensional concept that in this article is called *integrity*. One dictionary definition of *integrity* is "completeness," which can be tied to multiple perspectives represented by product attributes. Often, people think of goodness from one perspective, e.g., manager or developer, or in terms of an attribute, e.g., budget, schedule, or requirements. Our integrity concept allows for blending of multiple perspectives. For example, a manager may think of product goodness as the product being delivered on time or within budget or both. A developer may think of product goodness as the product doing what the customer wants. We think of product goodness as product integrity that folds in all the perspectives.

As shown in Figure 1, we use the notion of displacement (length of a line) in space to derive the idea of a product integrity index that combines all perspectives into a single number. The space of interest is product attribute space; that is, the axes in product integrity space are product attributes. Figure 1 illustrates this notion of displacement for the case of three attributes.

The measurement concept presented is a general approach to quantifying almost any object. We call this approach Object Measurement®, which is a registered trademark owned by Scott E. Donaldson and Stanley G. Siegel.

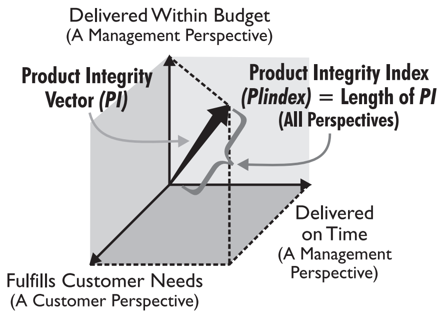
Delivered Within Budget
(A Management Perspective)
Product Integrity Vector (PI)
Product Integrity Index (PIindex) = Length of PI
(All Perspectives)
Delivered on Time
(A Management Perspective)
Fulfills Customer Needs
(A Customer Perspective)

Figure 1. *Object Measurement® uses the notion of a vector length to combine multiple perspectives into a single quantity called a Product Integrity Index.*

We recognize that the attributes we may choose to fold into our notion of "product integrity" would not necessarily be the same as the attributes you would choose. Consequently, the measurement approach we discuss in this article allows you to mold the product integrity concept to the needs of your organization.

This article provides an example set of attributes to define an example product integrity index, a general formula to compute the index, and a summary of how the integrity index concept has been applied in the real world of software systems development.

Measuring Product Integrity

To illustrate the generality of our measurement approach, we now consider a product integrity space that consists of the following five attributes (vs. the three attributes already introduced):

- at_1 *Fulfills specified customer needs.* The product does what it is supposed to do as recorded and agreed to by the customer and the seller.
- at_2 *Can be easily and completely traced through its lifecycle*. The product is "maintainable"—it can be easily updated to incorporate new things, revise existing things, and get rid of things no longer deemed needed by the customer.
- at_3 *Meets specified performance criteria.* How many? How often? How long? These criteria are sometimes considered special cases of the first attribute (at_1).
- at_4 *Meets cost expectations.* The product costs what the customer and the seller agreed.
- at_5 *Meets delivery expectations.* The product is delivered in accordance with schedules agreed to by the customer and the seller in a project plan or updates to the plan.

To understand how we can use the idea of a length of a line in space to quantify the concept of product integrity as a means for measuring customer satisfaction, consider the following five-dimensional vector.

Equation 1. *Five-dimensional measure.*

$$PI = \frac{\begin{bmatrix} at_1 \\ at_2 \\ at_3 \\ at_4 \\ at_5 \end{bmatrix}}{N} = \frac{\begin{bmatrix} CustNeeds \\ Traceable \\ PerfCrit \\ WithinBudget \\ OnTime \end{bmatrix}}{N}$$

In Equation 1, product integrity (*PI*) is a vector in five-dimensional product integrity space whose components, at_i, are the example product integrity attributes defined earlier. The quantity *N* is a normalization factor that establishes a "product goodness scale." As we subsequently explain, we choose *N* so that the length of *PI* is restricted to the range from zero to one.

To turn Equation 1 into a measurement tool, consider the following questions:

- How can a five-dimensional quantity be converted into a single quantity to simplify measurement interpretation?
- What scales should be established for the attributes?
- What relative weights should be assigned to the attributes?
- How can a scale of values be assigned for the single quantity?

Clearly, there are many sensible ways to address the above questions. The first question deals with simplifying measurement. As Equation 1 indicates, multidimensional expressions of product integrity are possible. Each product attribute dimension contributes to the "length" of the vector *PI*. To convert the five-dimensional quantity in Equation 1 into a single quantity (to represent "quality" or "completeness"), we calculate the "length" of the vector. We call the length of *PI* the Product Integrity Index, or *PIindex*. As subsequently explained, this product integrity vector length, *PIindex*, is simply the square root of the sum of the weighted (w_i) squares of the attributes at_i divided by the normalization factor *N*.

The second question deals with attribute value scales. Many people find it useful and convenient to quantify things in terms of percentages. Thus, a convenient range for an attribute value scale goes from zero to one. Again, for simplicity, we take the approach of limiting the attribute value scales to the range zero to one.

The third question deals with relative weights for product attributes. If we assign the same scale to each attribute (namely, zero to one), we are weighting each attribute equally. For simplicity, we will take this approach. However, you may wish to emphasize one attribute more than the others. For example, if you wanted to give "delivering on time" double the importance of any of the other attributes, you could set its scale to run from zero to two and set the scales of the other attributes to run from zero to one. Equivalently, you can keep all the scales the same and give prominence to selected attributes through the use of weighting factors (w_i).

The fourth question deals with establishing a value scale for the length of *PI*. We select a scale for the magnitude of this vector by choosing a value for the normalization factor *N*. Arguing as we did before, we merely select a scale that ranges from zero to one. For equally weighted attributes, the value of *N* then becomes the square root of the sum of the squares of the maximum values that the attributes at_i can take on. For the case where a product has five attributes each with a maximum value of one, the value of *N* thus becomes the square root of 5.

Defining a PIindex

Based on the four questions introduced above, we can now define a product integrity index, *PIindex*, that ranges from zero to one as shown in Equation 2.

The *PIindex* is normalized to one, i.e., restricted to the range of zero to one. If you want to remove this normalization, remove the denominator.

To illustrate how Equation 2 works, we define value scales for each example software product attribute (at_i). There is a multiplicity of ways such assignments can be made. This example provides insight into ways that you can make such assignments that are relevant to your organization.

$$PIindex = \frac{\sqrt{\sum_{i=1}^{n} w_i^2\, at_i^2}}{\sqrt{\sum_{i=1}^{n} w_i^2\,(\text{maximum}[at_i])^2}}$$

where at_i = product integrity attribute
n = number of product integrity attributes
w_i = weighting factor for attribute at_i
maximum $[at_i]$ = maximum value of at_i.

Equation 2. *The PIindex.*

Fulfills Customer Needs (at_1)

For at_1, set up a three-value scale based on an Acceptance of Deliverable Form[1] as follows:

$at_1 = 1$ if the customer returns the form indicating "accepted as delivered."

$at_1 = 0.5$ if the customer returns the form indicating "accepted with minor changes."

$at_1 = 0$ if the customer returns the form indicating "changes to be negotiated."

If we wanted to provide more insight into the percentage of requirements fulfilled, we could count such requirements that appear in the product and compare them against some ground truth showing what this number of requirements should be ("shalls" in the language of requirements analysis).

Can Be Easily and Completely Traced Through Its Lifecycle (at_2)

For at_2, the situation can become complicated. Depending on the product, traceability may involve more than the product. For example, if the product is computer code, traceability involves the existence of predecessor products such as design and requirements specifications. If the product is a requirements specification, traceability typically involves documents that a customer may supply such as congressional legislation or corporate policies. More generally, traceability involves such things as product decisions recorded at change control board (CCB)[2] meetings, internal project meetings, and recorded seller and customer management conversations and E-mail between these two parties. To keep things simple, we set up a three-value scale based on the existence of records showing how the product evolved as follows.

$at_2 = 0$ if nothing other than a customer-prepared statement of work exists that calls for the development of the product.

$at_2 = 0.5$ if written records exist for some part of the project's lifecycle that show how the product contents are what they are.

at_2 = 1 if detailed written records exist throughout the life of the project showing how the product contents are what they are.

Meets Specified Performance Criteria (at_3)

For at_3, merely set $at_3 = at_1$ since performance criteria are often lumped together with customer needs. If this is not the case in your environment, follow the suggestions offered above for the attribute at_1.

Delivered within Budget (at_4)

For at_4, set up a three-value scale.

at_4 = 1 if the product was delivered for less than the cost specified in the project plan or as modified in CCB minutes.

at_4 = 0.9 if the product was delivered for the cost specified in the project plan or as modified in CCB minutes.

at_4 = 0 if the product was delivered for more than the cost specified in the project plan or as modified in CCB minutes.

Clearly, the above scale places a slight premium on delivering for less than planned cost. The scale also ranks a deliverable delivered for one dollar more than planned cost the same as a deliverable delivered for $3,000 more than planned cost. Again, in your environment you may not wish to place a premium on delivery below cost—but the above gives the idea for how you can establish such premiums (this also applies to the attribute at_5 below).

Delivered on Time (at_5)

For at_5, we set up a scale as follows.

at_5 = 1 if the product was delivered before the delivery date specified in the project plan or before the delivery date as modified in CCB minutes.

at_5 = 0.9 if the product was delivered with no more than a 10 percent schedule slippage. Here, "percent slippage" is calculated by taking the length of time allocated in the project plan for preparing the product or as modified in CCB minutes, dividing that time into the slippage time, and multiplying by 100. For example, if the product was scheduled to be delivered 10 weeks after project start but was delivered 11 weeks after project start, then at_5 = 0.9 because the slippage was (1/10) x 100 = 10 percent.

$at_5 = (1 - X)$, where X is the fraction of schedule slippage as calculated above. For example, if the product was scheduled to be delivered 10 weeks after project start but was delivered 13 weeks after project start, then at_5 = (1 - 3/10) = 0.7. For all schedule slippages greater than or equal to the original length of time to produce the deliverable, at_5 = 0 (for example, if a deliverable was to be developed over a 10-week period, any delays greater than or equal to 10 weeks result in at_5 = 0).

The above scale places a slight premium on delivering early. The above scale favors on-time product delivery while allowing for some planning leeway.

Example: *PIindex* Calculation for a Requirements Specification

The product is a requirements specification. After delivery, the customer sent back the Acceptance of Deliverable Form showing "accepted with minor changes." Thus, $at_3 = at_1$ = 0.5. The product was delivered on time, so at_5 = 0.9. The project plan called for 300 hours to be expended on the task to produce the document, but only 275 hours were expended. Thus, at_4 = 1. Written records consisting of CCB minutes that show decisions underlying the document's content exist for some part of the project. Thus, at_2 = 0.5. The *PIindex* for this requirements specification is therefore the following:

$$PIindex = \frac{\sqrt{0.5^2 + 0.5^2 + 0.5^2 + 1^2 + 0.9^2}}{\sqrt{5}} = 0.72$$

Benchmarks

It is easy to measure, for example, our weight. However, the resultant measurement is generally of little value if our objective is to gain or lose weight. We need weight benchmarks to know whether we are underweight or overweight. Similarly, we need benchmarks for *PIindex*. For example, we can use *PIindex* to establish norms for "product quality" or "completeness." As you gain experience with this index, you can establish goals for various types of products, projects, and seller periods of performance. For example, you can establish goals such as the following:

- Legacy system releases for which little or no documentation exists shall have a *PIindex* not less than 0.75.
- Deliverables for projects whose ultimate objective is to produce a new software system shall have a *PIindex* not less than 0.85.

For this reason and others, it takes time to institute a measurement process. Keep the measurement process simple—otherwise, it will die quickly. "Simple" means "easy-to-collect data and easy-to-interpret information resulting from these data."

Application in the Real World

In this section we highlight how the ideas described above have been applied in the real world of software systems development. This discussion may offer you some ideas for instituting a measurement program in your organization if you do not currently have one or for adjusting an existing measurement program. Reference [1] elaborates on these ideas.

For over five years, we have participated in the development and operation of a software systems development center that could be termed a "software factory." We use the notion of a factory here to suggest that the center manufactures, in a consistent fashion, software products for a diverse customer base. Some customers want minor repairs to existing legacy systems, some want major enhancements, while others require an entirely new software system. The center services from 30 to 50 customers. Customer project sizes range from one person to approximately 50. Typically, customers fund projects for a period not exceeding 12 months, and project budgets range from tens of thousands of dollars to several million dollars. The center produces approximately 100 contract deliverables a month. These deliverables span a broad range of software and software-related products and supporting services.

For each deliverable provided to a customer, we calculate the deliverable's product integrity using the following two attributes: at_1 = "fulfills customer needs," and at_2 = "meets delivery expectations." As each deliverable is provided to the customer, the center uses a Customer Acceptance of Deliverable Form to collect, in part, customer feedback on the product integrity attribute "fulfills customer needs," i.e., at_1. The customer fills out the acceptance form by selecting one of the following choices on the form.

- Deliverable is "accepted as delivered" ($at_1 = 1.0$).
- Deliverable is "accepted with minor changes" ($at_1 = 0.5$).
- Deliverable is "not accepted and changes need to be negotiated" ($at_1 = 0.0$).

In addition, as each deliverable wends its way through the center's product development process, it is tracked by a form that we call the *Deliverable Tracking Form*. This form has entries on it tied to the activities that make up the center's product development process. These activities include such things as peer reviews, product assurance support, technical editing (for deliverables that are documents), and project-level and organization-level management reviews and sign-offs. Two pieces of data that are captured on this form are the customer's required deliverable due date and the date the deliverable is provided to the customer. From this data, we can assign a value to the product integrity attribute for meeting delivery expectations as follows:

- Deliverable is "delivered on time" ($at_2 = 1.0$).
- Deliverable is "delivered late" ($at_2 = 0.0$).

The resulting *PIindex* provides insight into the fulfillment of a customer's needs and meeting a customer's delivery expectations. With a 100 deliverables a month, this simple *PIindex* helps point out areas where there may be (we stress, *may be*) potential problems.

To complement a deliverable's *PIindex*, we also calculate an integrity index for the product development process used to produce the deliverable. The process integrity index is an extension of the *PIindex* concept. Again, to keep our measurement activities simple, we use the back of the Deliverable Tracking Form to capture the data we need to calculate the process integrity index. The back of the form contains a set of six value scales that are tied to the product development process activities on the front of the form. The responsible person, e.g., principal author of a product, technical editor, or product assurance reviewer, circles the appropriate value on the value scale. These circled values are then used to calculate a process integrity index for the process used to produce the deliverable.

We store the product integrity data and process integrity data in a centralized data base. This data base is accessed by a tool that helps us answer questions such as the following:

What is the center's average PIindex for the last three months? The answer to this question provides global insight into how well the center is doing with respect to giving its customers what they asked for on time. An average product integrity value close to 1 means that the center is consistently delivering "good" products, where goodness means "on-time delivery" and "products that do what they are supposed to do."

STSC Measurement Assistance

The product integrity index (*PIindex*) can be useful to organizations that already collect and use measurements. However, if an organization does not have a measurement program in place and management merely wants to check a box, the *PIindex* should not be used as a "silver bullet." In this article, the authors did not discuss the importance of unfolding the data that make up the *PIindex* or how to unfold it because of space limitations (see reference [1] for a complete explanation). An organization may want to use the *PIindex* to see an overall trend or use the *PIindex* as a trigger for action, but to use the *PIindex* solely to analyze and interpret data hides information many organizations need to know. It is important that organizations continue to look at their raw data and the extremes in that data for the messages and trends they contain.

If you are just starting a measurement program, begin with simple measures that address your organization's issues and let the program evolve as you learn. Choose measurements that are important to you, that help you reach your immediate goals and address your issues. As Donaldson and Siegel state, the attributes they chose might not necessarily be the attributes you would choose. If you find it difficult to decide what you want from a measurement program, a good start might be the basics of size, effort, cost, schedule, defects, and rework. Once you find a set of measurements that work for you, you can then combine related information to create your composite metric.

If you want help to determine what measurements your organization needs or want help to get a measurement program running, the Software Technology Support Center (STSC) can help you. We can perform a measurement capability evaluation of your organization, then use the data to return information about your measurement strengths and weaknesses and recommend steps for improvement. We can also support you with training and hands-on coaching while you develop your organizational measurement program.

Beth Starrett, STSC
OO-ALC/TISE
7278 Fourth Street
Hill AFB, UT 84056-5205
Voice: 801-775-5555 ext. 3059 DSN 775-5555 ext. 3059
Fax: 801-777-8069 DSN 777-8069
E-mail: starretb@software.hill.af.mil

What has been the PIindex trend over the past two months? The answer to this question provides insight into whether the center's products are improving, staying the same, or declining with respect to on-time delivery and content.

Which projects have deliverables with PIindex values less than 0.5? The answers to this question help senior center management turn its attention to projects that may need extra attention.

What is the center's average process integrity index for the last three months? The answer to this question provides global insight into how well the center is following the systems engineering environment documented practices. An average process integrity value close to 1 means that the center is consistently following stated practices. An average process integrity value close to 0.5 means that the practices are not being done as documented. There are multiple reasons why this may be the case. For example, some of the practices just do not make good sense given the type of products being produced. Consequently, the practices may need to be rethought. Perhaps employees are simply ignoring the practices. Whatever the case, the process integrity index helps to highlight those areas of the product development process that may need to be improved.

What if the average product integrity score is near 0.5 and the average process integrity score is 0.95? The answer to this question may be that the project teams are following the documented practices but are producing products the customers are not accepting.

From a review of the above measurement observations, decision makers, product developers, and others can focus their attention (and potentially, resources) on those activities that may need improvement. The decision might be to take more measurements and review them carefully. Perhaps the software development process needs to be more closely followed, maybe the process needs to be changed, or maybe the management or the product development staff are overcommitted. Regardless, product and process integrity measurements can be expressed in everyday terms to help achieve customer satisfaction.

Summary

This article describes and illustrates an approach for measuring the “goodness” of software-(related) products. For us, “product goodness” is a multidimensional concept. The label we put on this concept is *integrity*. The following five steps capture the essence of our product measurement approach.

- Decide on the questions that you want or need to address, e.g., are we producing “good” products?
- Select the products from your software systems development process that you want to measure, e.g., requirements specification.
- Identify the product attributes that you want to measure, e.g., for a requirements specification, you might identify an attribute as “at_4, meets cost expectations.”
- For each identified attribute, e.g., at_4, define a value scale in everyday terms that are familiar to the organization e.g., delivered for more than the cost estimate = 0.0, delivered for cost estimate = 0.9, and delivered for less than cost estimate = 1.0.
- Using Equation 2, calculate the *PIndex* value. For simplicity, use the formula to restrict values between zero and one. Select weighting factors to reflect your perception of the relative importance of your product attributes.

The measurement approach described in this article can be extended to measure almost any object. In particular, we have introduced how it can be used to measure your software systems development process. This extension, as well as a more in-depth discussion of the ideas presented in this article, are given in reference [1]. ◆

About the Authors

Scott Donaldson, a corporate vice president with Science Applications International Corporation (SAIC), has been in the computer field since 1974. He has a broad range of software engineering experience in the public, private, and commercial industries, including the design, development, implementation, technical management, and evaluation of computer systems application. He is the director of an SEPG that helps a 350-person organization achieve SEI Level 3. He is also the deputy program manager for this more than $28 million business. He received a bachelor’s degree in operations research from the U.S. Naval Academy and a master’s degree in systems management from the University of Southern California.

SAIC
200 North Glebe Road, Suite 300
Arlington, VA 22203
Voice: 703-516-0603
Fax: 703-516-0618

Stanley Siegel, an assistant vice president with SAIC, has been in the computer field since 1970. Since 1976, he has specialized in the area of software product assurance and co-wrote a textbook on the subject that appeared in 1987. He is a co-author of the first textbook on software configuration management. Together with Scott Donaldson, he wrote a book on software process improvement that was published in March 1997. He is a member of an SEPG in an organization working on approximately 40 software projects of various sizes. He holds a doctorate in theoretical nuclear physics from Rutgers University.

SAIC
200 North Glebe Road, Suite 300
Arlington, VA 22203
Voice: 703-516-0608
Fax: 703-516-0618

Reference

1. Donaldson, S.E. and S.G. Siegel, *Cultivating Successful Software Development: A Practitioner’s View*, Prentice-Hall PTR, Upper Saddle River, N.J., 1997.

Notes

1. As part of our software systems development process, we use an Acceptance of Deliverable Form to obtain, in part, customer feedback. When the product is delivered to the customer, the customer reviews the delivered product, decides the product’s status, signs the form, and returns the form to the seller. For this example, we have assigned discrete values for the three possible customer evaluations.
2. The CCB is a management support tool that provides a forum for discussing management, development, and product assurance activities. Our concept of a CCB extends far beyond the traditional configuration management control board concept. Simply stated, no matter how well the customer articulates what is needed to be done, no matter how well the seller writes a corresponding project plan, and no matter how well the customer and the seller negotiate the final agreement, once the project begins, things start to change. Furthermore, changes persist throughout the project. Therefore, the CCB is a business forum where the customer and the seller can discuss how to deal with this unknown but anticipated change.