Application of Statistical Process Control to the Software Process

Mark A. Lantzy
BTG, Inc
1945 Old Gallows Road
Vienna, Virginia 22182

Introduction

The increasing demand for software continues to outpace our production capabilities in terms of both quantity and quality. Proposed solutions to this dilemma range from the creation of the informal environments of garage-shop start-ups to the implementation of the formalized processes of software factories. IBM recently announced its long-awaited OS/2 Version 2.0 amid confessions of a purposely unstructured development environment intended to "replicate the spirit of the small start-up companies from which so much successful software has emerged." [1] Meanwhile, the Department of Defense (DoD) has adopted a process maturity framework [2], [3] that supports a highly structured approach to software production. The framework advocates the application of the principles of statistical process control to a software process that builds products according to a plan while simultaneously improving its capability to produce better software [2]. The reported IBM approach to the development of OS/2 Version 2.0 stands in sharp contrast to the DoD's process maturity framework, rekindling a long-running argument regarding whether software production is an art reserved for talented virtuosos or an engineering discipline that, with time, can be subjected to the rigor of a product manufacturing environment.

One element of this debate is the issue of whether or not statistical process control can be applied to the software process. The software engineering process can be defined as a set of software engineering activities needed to transform user requirements into software [2]. As with any process, the application of statistical process control to the software process is a desirable goal. In the statistical sense, process control refers to the ability to bring measurable characteristics of process outputs within control limits. If we view software as a manufactured product then, theoretically [8], we can apply the same principles of statistical process control used to manufacture telephones, personal computers, camcorders, and other mass-produced commodities to the production of software. Unfortunately, production of software is inherently different than the production of these commodities. Unlike a manufacturing process, the inputs into and the outputs from the software process are different for each instance of the process. More importantly, the transformation of user requirements into software is dominated by cognitive activities, many of which encourage multiple outputs permitting the selection of an optimal choice. A goal of most manufacturing activities is minimization of cognition. Higher levels of cognition increase variances in productivity and quality making the application of statistical process control more difficult. For the software process, a process dominated by cognitive activities, how can we apply statistical process control in a meaningful and economically useful manner?

Many existing applications of statistical process control to the software process suffer from a bias toward the application to manufacturing processes [4]. Theoreticians and practitioners do not sufficiently consider the fundamental differences between the software process and manufacturing processes. This has resulted in the propagation of some potentially dangerous axioms such as "if you can measure it, it must be good." A more appropriate analogy is that of the software process to the design and implementation of the manufacturing operation rather than the execution of it. This analogy more accurately reflects the cognition-dominated activities or design risk [5] of the software process.

Despite the difficulty, the software process is a process that can be brought under control in the statistical sense. This paper examines some of the principles of statistical process control that have been successfully applied to a variety of manufacturing processes and offers a set of transformations on these principles that permit their application to cognition-dominated processes such as the software process. In many cases, the transformations are not always as clean as we would like them to be. But fuzzy transformations are not new to the software community. A case in point is the application of hardware reliability measures such as mean time between failure (MTBF) to software.

The remainder of this paper is divided into four sections. The first section, Statistical Process Control, introduces some of the fundamental concepts of statistical process control. Of particular importance in this section is the

concept of variation and its inevitable presence in processes. The second section, Quality Characteristics, defines quality characteristics and the importance of their specification to process implementation and measurement. The third section, Software Processes, examines the effects of statistical process control principles on the development, specification, assessment, implementation, and auditing of the software process. This section introduces an implementation strategy to process development that supports the application of statistical process control principles. The fourth section, Process Specification and Management, addresses the importance of process specification and management to the software process in terms of assessment and measurement. The fifth section, Next Steps, enumerates implementation steps and describes an example application.

Statistical Process Control

A *process* can be defined as a series of activities that transform inputs into outputs in accordance with customer specifications. The definition of process used here is more restrictive than other definitions in two ways. First, customer specification of the end product or the process is required. Second, there is more than one activity required to transform the inputs into outputs, and each activity is the responsibility of an organizational element within an enterprise. Semantically, an organizational element performs an *activity* within a process. The set of organizational elements, the activities they perform, the interfaces between them, and the end-item or process specification compose a process. This does not necessarily imply that a craftsman who single-handedly makes a wooden rocking chair does not use a process. The definition used here simply implies that the craftsman plays the role of many organizational elements, each performing a discrete activity that when combined represent the process of building the rocker.

The customer specifications referred to by the above definition of a process may take one of two alternative forms. A customer may specify the process and, as a result, be responsible for the outcomes of the process. More typically, the customer provides a specification of an end-item in terms of quality characteristics and transfers responsibility for the process to the developer. For example, a threaded fitting would have a set of quality characteristics that would include pitch diameter, length, and strength, just to name a few. Customer specifications for the fitting would include values for each of these quality characteristics. The developer is responsible for the implementation of a manufacturing process that produces fittings according to the customer specification. However, specification of the values is not sufficient. Why? If the customer specified that the fitting have a pitch diameter of 0.2500 inches, how many fittings with pitch diameter of exactly 0.2500 inches would a high-quality manufacturing process produce out of a lot of 1000? Theoretically, the answer is zero because the pitch diameter is what is referred to as a continuous variable. In other words, the manufacturing process is subject to *variation* causing some fittings to have pitch diameters slightly greater than 0.2500 inches and other fittings to have pitch diameters slightly less than 0.2500 inches. Consequently, the customer must provide a specification in terms of tolerances. For example, a fitting may be viewed as acceptable or conforming if the pitch diameter is between 0.2492 and 0.2508 inches or stated another way, 0.2500 ± 0.0008 inches. A fitting is viewed as defective or nonconforming if the pitch diameter is less than 0.2492 or greater than 0.2508.

Statistical process control is a methodology used to determine if a process is *in control* in the statistical sense of the word. For a process to be in control, variation in the quality characteristics of the end-item must be predictable. Shewhart [6] established that variation in a quality characteristic has two types of causes.

- Common (chance) causes. Causes that are always part of the process.
- Special (assignable) causes. Causes that arise due to special circumstances. Causes that are not always part of the process.

As an example of the difference between common and special causes of variation, let's look at a software development effort that has fallen behind schedule during the implementation activity. Upon investigation, management determines that the schedule slip is due to a decrease in programmer productivity. Programmer productivity is affected by a variety of causes that are common to the process such as programmer capability, programming standards, language, machine availability, and tools. Other causes that affect programmer productivity are special to particular instances of the process. For example, excessive computer down time due to power outages, inadvertent use of the wrong programming standard, outbreak of the flu, and others may be considered special causes of variation in programmer productivity. Processes that suffer from these special causes of variation do not produce predictable results and are said to be *out of control* in the statistical sense. Identification of the cause of the decrease in programmer productivity will determine the action that management takes in response to the variation.

In order to examine the results of special causes of variation on the outputs of processes more formally, let's turn our attention back to the fitting discussed above. We have already stated that the pitch diameter of the fitting is a

continuous variable. The pattern of variation or frequency distribution of the pitch diameter and many other variables of end-items of manufacturing processes are modeled or represented quite nicely by common probability distributions such as the normal distribution. Variation within the defined distribution results from common causes. Special causes of variation result in a fundamental change to the frequency distribution that represents the behavior of the variable. Figure 1 contains an example of just one of the ways the frequency distribution may change. The graph in the foreground represents the frequency distribution of the variable when the process is subject to only common causes of variation. The frequency distribution in the background is shifted to the right. This shift is caused by the introduction of a special cause of variation into the process. One result of the shift is that there is an increase in the area of the distribution that is greater than the upper tolerance limit. This means that the probability of creating a nonconforming end-item has increased. Thus, we see the effects of special cause variation on the predictability of processes. For completeness, it is worth noting that Figure 1 represents just one possible effect of a special cause of variation to the distribution, a shift in the mean of the distribution. The point to be made is that special causes of variation result in fundamental changes to the frequency distribution for a variable result of a process.

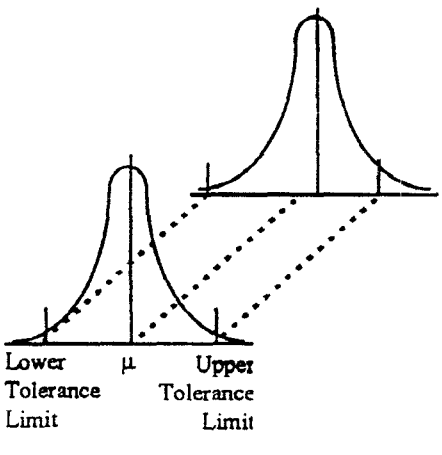

Figure 1. The Effect of Special Causes on Frequency Distributions

The enlightened manager understands the distinction between common and special cause variation. When variation results from special causes, management works with workers in the process to eliminate the cause. The elimination of special causes of variation is not a process panacea. The graph in the foreground of Figure 1 represents a process dominated by only common causes of variation. This does not mean that we are happy with the process. For example, the area of the distribution that is outside of the upper and lower tolerance limits may represent a probability of nonconformance that is unacceptable. When variation is the result of common cause, management works with the workers in the process as well as outside experts to identify process defects and *reengineer* the process to eliminate common causes of variation and narrow the distribution around the mean until the results are considered acceptable. Unenlightened managers make decisions based on the effects rather than the cause. Decisions that are based on effects can result in [7]:

- Blaming people for problems beyond their control
- Spending money for new technologies or new people that are not needed
- Wasting time looking for an explanation of a perceived trend when nothing has changed
- Taking other actions when it would have been better to do nothing

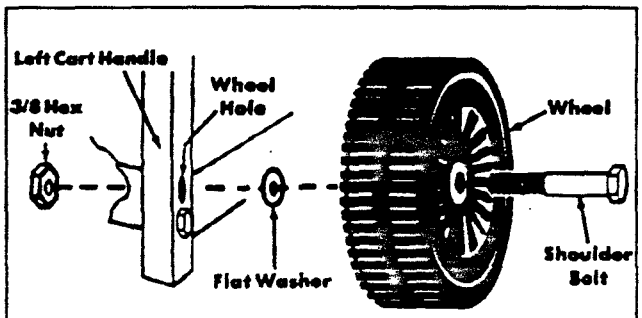

Figure 2. Shoulder Bolt for Wheel Assembly

Shewhart's [6] method for determining whether a process was subject to special cause variation was the control chart. The purpose of the control chart is to identify unstable processes. Unstable processes are those processes that are subject to special causes of variation. Figure 2 contains a drawing of a wheel assembly for a manufactured item. A variable of interest in the machining of the shoulder bolt is the diameter of the portion of the bolt that serves as the wheel axis. If the diameter is too large, the fit of the bolt into the wheel will be snug and the wheel will not roll as freely as desired. If the diameter is too small, the wheel will wobble. Figure 3 contains an example of the Shewhart control chart for the diameter of the shoulder bolt. Points on the chart represent the average diameter of the bolt in the nth sample. The points that are circled are those points that are outside of the upper and lower control limits which are represented by the dashed lines. These points indicate that the process is out of control, that is, subject to special causes of variation. (A detailed discussion regarding control charting including calculation of upper and lower

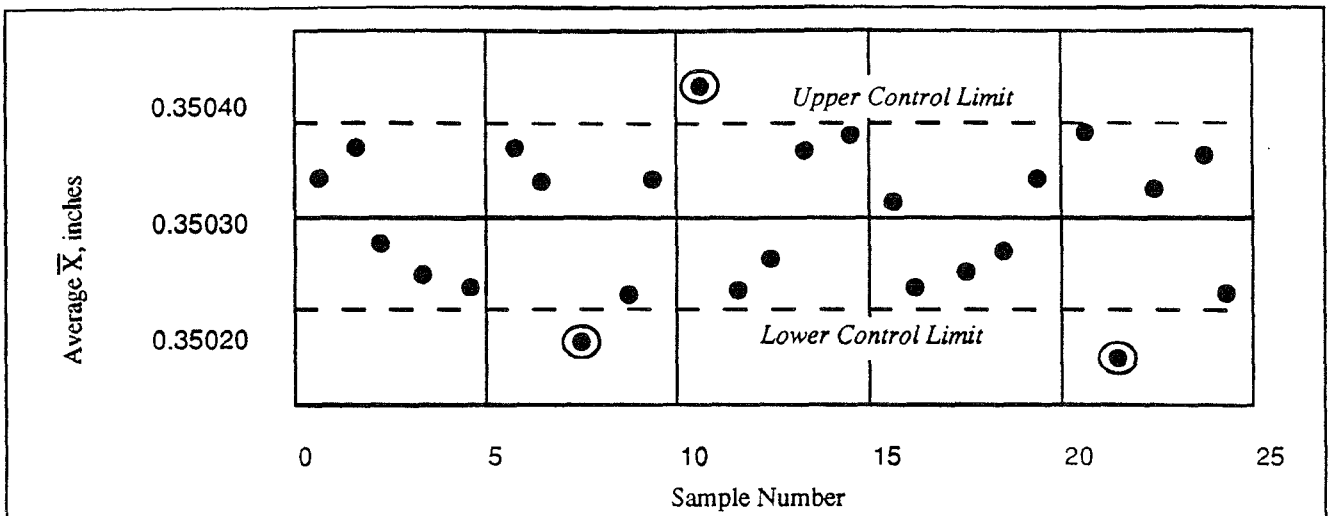

Figure 3. A Control Chart

control limits is beyond the scope of this paper. For a detailed discussion see [6], [8].)

The elimination of special cause variation results in a stable process for which, according to Deming [9], performance is predictable, costs and quality are predictable, productivity is at a maximum, and the effect of changes is measurable. Again, this doesn't necessarily mean that we are happy with the results of the process. It does imply that variation in the process is the result of common causes and the process, as it currently exists, cannot perform any better.

Software Quality Characteristics

The translation of the principles of statistical process control, as summarized above, to the software process is not trivial. The software process has been identified as a process that is dominated by design risk [5] or cognition because each instance of a software process produces a unique product for which quality is determined by conformance to customer requirements measured in terms of software quality characteristics. Software quality characteristics are those attributes of software that include correctness (conformance to functional requirements), reliability, understandability, portability, maintainability, testability, robustness, usability, cost-to-develop, and time-to-develop. In the past, many of these characteristics have been referred to as software quality attributes [10]. Adherence to customer requirements refers to the elimination of variance in quality characteristics, in other words, eliminating differences between customer expectations and the delivered software product. Because these requirements vary for each customer and often times conflict, even within a given instance of the software process, practical application of statistical process control requires thoughtful action on the part of the developers. Effective application depends on the ability of managers to negotiate a prioritized list of quality characteristics and acceptable tolerances with their customers and apply statistical process control principles in a manner that assures conformance of the software product to that prioritized list.

Resolution of conflicting software quality characteristics has already been identified as a major concern of the software development manager [11]. Successful resolution of these conflicts depends on goal setting prior to the start of the software process. More importantly, the process should be designed based on the product goals. For example, one would expect the process of developing the avionics software for the next generation passenger aircraft to be significantly different from the process of developing the application software for a military intelligence workstation designed to automate the manual activities of correlating and fusing information from sources of intelligence data. For the avionics application, reliability, correctness, and efficiency are the high-priority quality characteristics. For the intelligence application, usability, maintainability, and portability are the high-priority quality characteristics due to the uniqueness of the product and the more than likely evolutionary nature of the development life cycle. Obviously, the differences in quality characteristics between these applications are due to the respective problem domains. However, even within the same problem domain, different software products require a different set of prioritized software quality characteristics depending, of course, on customer requirements.

In the language of statistics, an important distinction is

made between the terms *variables* and *attributes*. A variable is an actual measured quality characteristic such as width, height, or diameter. An attribute refers to the conformance or nonconformance of a product typically by visual inspection rather than measurement. For example, a crack in a ceramic mug results in a nonconforming product. The property of not being cracked represents an attribute of the mugs. Statistical process control has to do with using the Shewhart control chart to examine the performance of a variable or an attribute over some time. Application is dependent upon the ability to specifically define conforming items versus nonconforming items. This has not proven trivial in the production of software. Too often, the acceptance (as conforming) of a software product is based on two attributes, reliability and "goodness." Reliability refers to the failure rate of the software. If the failure rate does not exceed a customer's tolerance, which is very often not quantified, then the software is viewed as conforming in terms of reliability. Goodness is almost impossible to quantify and refers to the customer's perception of the software—does it work the way the customer thinks it should [12]? Software is accepted—viewed as conforming—if it does not exceed the customer's tolerance for failure rate and is perceived as "good."

Specification of software quality characteristics requires a quantifiable and measurable agreement between customer and developer. These agreements can be difficult and time consuming, and like any software requirement, subject to change without notice. The investment can be worthwhile if executed properly. From the customer's perspective, benefits of the agreement include:

- The ability to control the development by specifying end-item objectives rather than process constraints
- Inclusion of time-to-develop and cost-to-develop as quality characteristics, equal in importance to functional and performance requirements
- A signed-up developer who will work harder to meet the goals when the agreement results from fair negotiations between customer and developer.

Software process control requires periodic monitoring of the progress being made toward the achievement of the quality characteristics agreed to between customer and developer. Figure 4 contains a graph of the expected progression of a specific quality characteristic over time. At time t the process owner develops an estimate-at-completion of the quality characteristic and finds that it falls short of the requirement. This may be an indication of a process failure, an unrealistic requirement, or a flawed monitoring process. Regardless of the cause, the immediate effect is an investigation into the process and a discussion with the customer about the potential ramifications. The investigation may result in changes in the development process, the monitoring process, or both, or a renegotiation of the requirement with the customer.

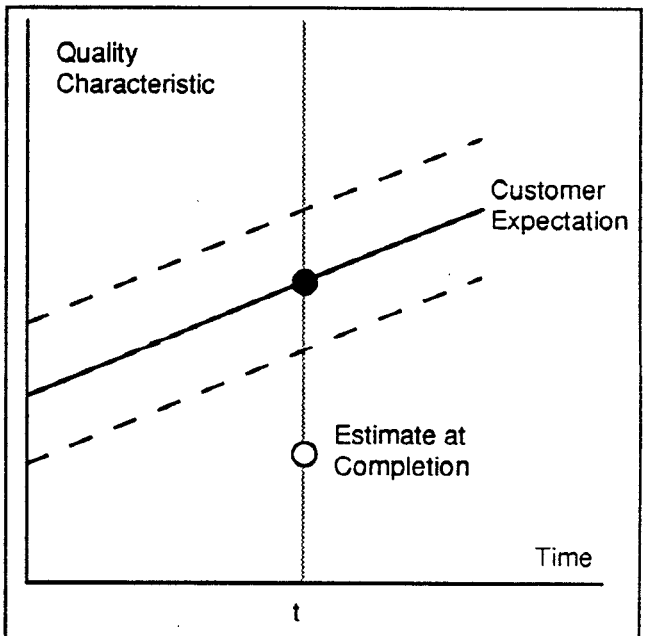

Figure 4. Keeping Quality Characteristics Within Customer-Defined Tolerances

Software Processes

A popular semantic in the software community is the phrase *the software process*. Of interest is the use of the word *the* which implies that there is a software process that is the correct software process for an enterprise. As discussed above, it is probably unrealistic to expect a single software process to satisfy the requirements of every customer of an enterprise. An important element of statistical process control is the explicit relationship between product and process.

In a manufacturing environment it is well understood that quality characteristics specified for an end-item have considerable effects on the process used to produce that item. Consequently, design-for-manufacturability has become a key attribute of the product design process. Creating product designs that allow for more relaxed tolerances for product variables can result in dramatic improvements in manufacturing quality. These improvements come from more easily satisfied tolerances rather than improvements in the manufacturing process. The result, none the less, is fewer nonconforming end-items and improved quality. For example, Figure 5 contains a diagram of an apparatus that requires a user to insert a handlebar into a carriage and secure it with a locking rod.

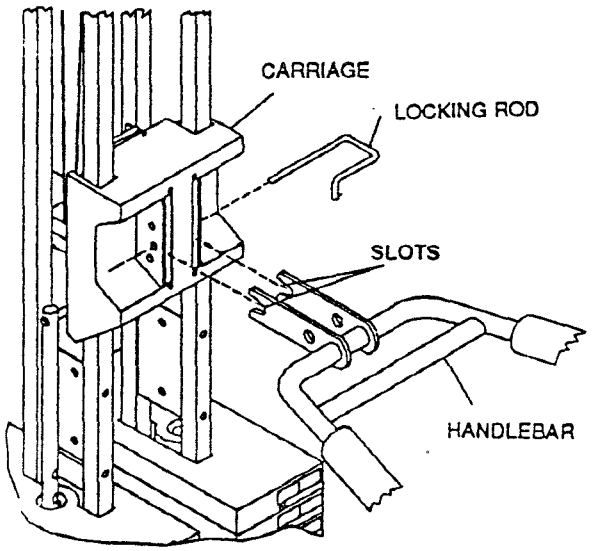

Figure 5. A Design Requiring Limited Manufacturing Tolerances

This design requires that the holes in the carriage and the handlebar be drilled within very limited tolerances if the user is to secure the handlebar with the locking rod easily and have the handlebar fit securely. Alternative designs are available that would ensure ease of use for the user, a secure fit, and less demanding tolerances. An alternative design has the potential to reduce costs in terms of production costs, nonconformities, and user satisfaction. This example illustrates the close relationship between product specifications and the production process. An understanding of this relationship provides the basis for a strategy for process development, implementation, and measurement that is applied to manufacturing processes and can be applied to software processes. The fundamental axiom supporting the strategy recognizes the importance of tailoring a process to the customer specifications of the product in terms of quality characteristics. All aspects of the process including measurement and reporting are designed to ensure delivery of what is important to the customer. Implementation of the strategy follows these four steps:

1) Identify the variables that correspond to the customer specified end-item quality characteristics. Variables in the statistical sense refer to quantifiable metrics that unambiguously measure satisfaction of quality characteristics.

2) Design and specify a production process tailored to satisfaction of the quality characteristics.

3) Identify process activities that influence the product variables identified in step 1.

4) Establish metrics and a measurement process for each activity identified in step 3.

Existing software process assessments [2] assume correlation between process maturity level and the ability to produce quality software. This approach may be overly simplistic. A software process must be assessed based on its capability to develop software that conforms to the quality characteristics specified by the customer for that instance of the process. This includes the establishment of tolerances for each variable and application of metrics to ensure compliance. Metrics applied to the process and its individual activities should measure progress against quality characteristics ensuring a software product whose quality characteristics are within customer specified tolerances. The degree of correlation between an activity metric and a customer-defined quality characteristic for the product determines the effectiveness of a given metric. Metrics that do not correlate to the quality characteristics defined for the end product are considered ineffective and unproductive. Where does the process owner find process metrics that correlate to end-item quality characteristics?

Processes, like the products they build, are systems composed of people, hardware, software, and operating procedures that together accomplish a set of specific functions. A software process, viewed as a system, has a development life cycle consisting of activities that include requirements definition, design, implementation, and test. The requirements for an instance of a software process are specified in terms of quality characteristics specified by the customer. Process design is established by partitioning the process into meaningful activities that can be defined in terms of inputs, outputs, and constraints. Implementation is accomplished by specifying the procedures for each activity and assigning responsibility for each procedure. Activities that produce tangible items are initial candidates for the application of metrics. Process owners identify quality characteristics for each tangible item that correlate to the quality characteristics specified for the software product. Tolerances are defined for each quality characteristic and metrics are put in place that ensure compliance. Documentation of the software process in terms of activities, inputs, outputs, constraints, quality characteristics, and metrics is contained in a process specification. This specification should be complete enough to permit assessment of the process and detailed enough to permit in-process audits. The purpose of the assessment is to ensure the organization's capability of producing a quality software product as defined by the customer of that instance of the process. The purpose of the in-process audit is to ensure process compliance.

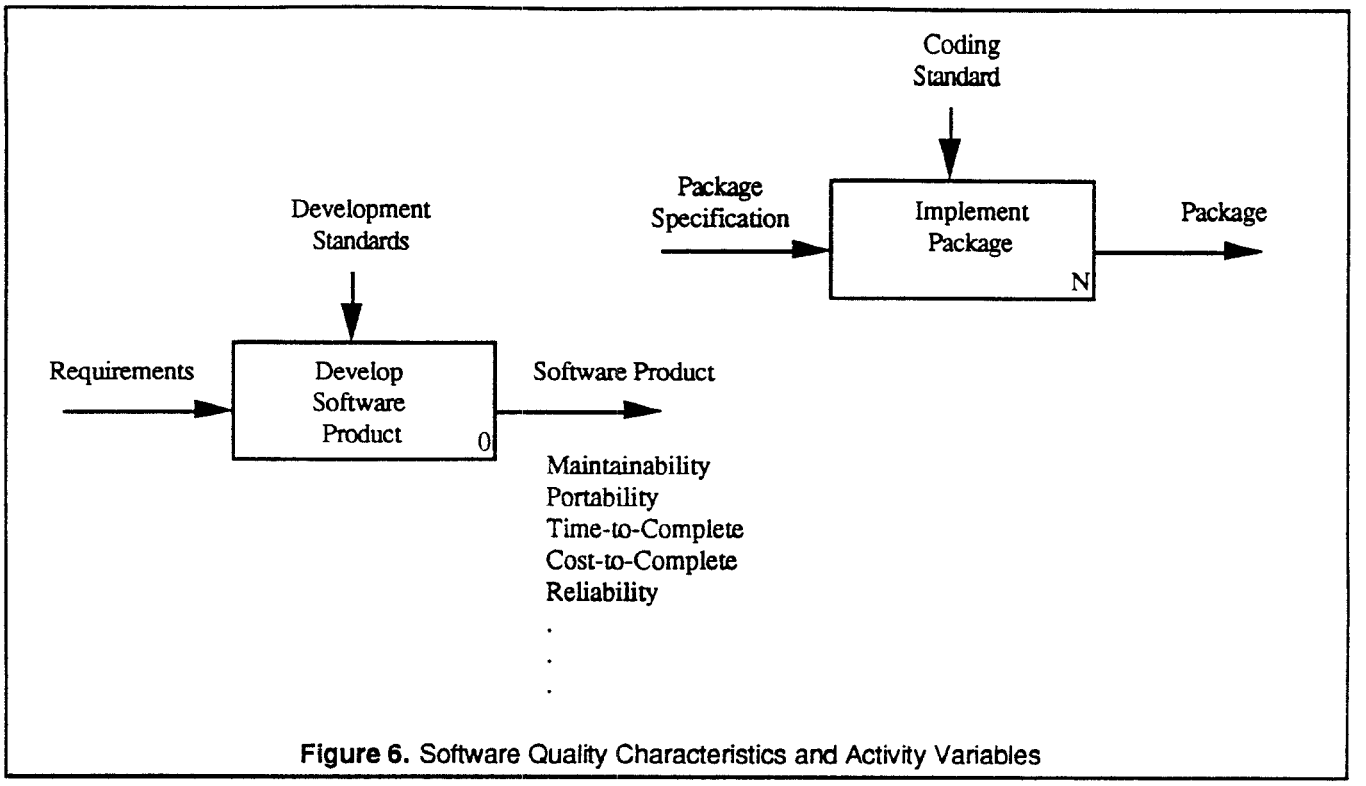

Figure 6. Software Quality Characteristics and Activity Variables

Figure 6 contains an example that demonstrates some of the concepts discussed above. The diagram on the left in Figure 6 is a model of a software process which represents the process as a single activity entitled *Develop Software Product*. Inputs to the activity include functional requirements. The constraint for the activity is the development standards chosen for this particular effort. The output, of course, is a software product. The diagram is annotated with a partial list of the prioritized set of quality characteristics identified by the customer and negotiated with the developer. The right side of Figure 6 contains a model of a discrete activity within the software process represented by the diagram on the left. This particular activity is entitled *Implement Package* and is responsible for transforming a package specification into a package with the constraint being the project's coding standards. The output of this activity, a package, is an ideal candidate for the application of a metric. A variety of metrics have been identified that could be applied here. However, our interest is in the application of metrics that correlate to the quality characteristics identified for the software product by the customer. Since the customer identified maintainability as the high-priority quality characteristic, a candidate variable for measurement is the number of coding standard violations. We assume here that past experience has shown a correlation between this variable and the quality characteristic maintainability. Without this past experience, managers must make assumptions of correlation and verify the results. With a variable identified, use of control charting or a comparable technique can point out whether or not this process is in control.

Applying statistical process control principles to processes dominated by cognitive activities is not an exact science. The frequency distributions of variables from manufacturing processes tend to fall neatly into the bell curves of the normal distribution permitting simplifying assumptions and straight-forward calculations to develop equations for control limits. Determining whether or not a process is in control merely requires establishment of a measuring process for the variables and charting over time. Even the variable, number of coding standard violations, described above represents a relatively trivial aspect of the package implementation activity. We would expect this variable to be brought under control quickly once errors are detected and the source of the errors corrected. We base our expectations on the fact that adherence to coding standards is not an activity that requires a high-degree of cognition. A variable from the package implementation activity that requires a higher degree of cognition is correctness. The term correctness here refers to compliance with the package specification. This implies that the package functions reliably and does the right thing. The activity associated with this variable requires a higher degree of cognition. Consequently, the behavior of the variable will not be as nice as those in manufacturing processes. Measurement, establishment of control limits, and identification of special cause variation, will be more difficult requiring solutions

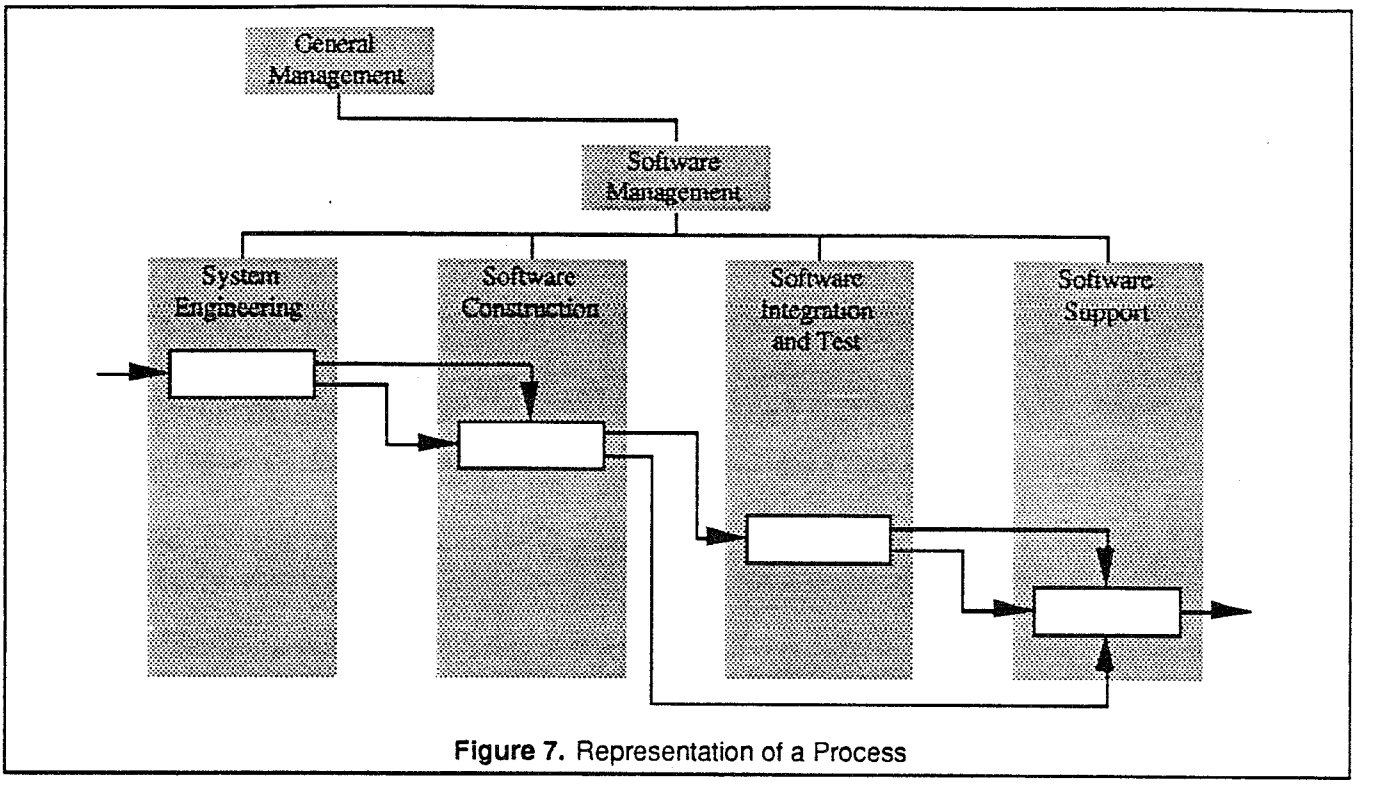

Figure 7. Representation of a Process

that are more intuitive than scientific. This does not eliminate the usefulness of the principles. However, it does modify the implementation significantly but not to the point of corrupting the basic principles.

Process Specification and Management

Earlier, this paper addressed the close relationship of product and process. The relationship formed the basis for a strategy that when implemented tailors a development process to the desired product. This tailoring is necessary to ensure satisfaction of the product quality characteristics agreed to between customer and developer. Failure to tailor the process results in static processes that tend to serve the developer's goals at the expense of the customer's goals. As discussed in the previous section, the tailoring of the process includes the tailoring of process metrics to the quality characteristics of the product specified by the customer.

An effective software process is engineered based on the requirements for the process and specified in a language that unambiguously conveys the functions that make up the process. Currently, many of the software development plans that we write do not provide a dynamic representation of the software process. Software organizations are typically composed of vertically aligned functional groups that need to interact effectively to succeed. The static representation of the organization in the form of an organization chart and a prose description of the role each functional area plays in the process do not sufficiently document the software development process. What is needed is a more precise method for documenting the dynamic nature of the software organization. The tool of choice for the documentation of people-intensive processes like the software process is Doug Ross's Structured Analysis and Design Technique or a subset of that technique called IDEF (Integrated Computer-Aided Manufacturing Definition Language). There are many advantages to the use of this technique.

- Each function is described in terms of its inputs, outputs, constraints, and mechanisms.
- Functional blocks can be assigned to vertically aligned functional areas in the software organization. Inputs, outputs, constraints, and mechanisms define the interfaces between functional areas.
- The model is simple to build and simple to read.
- The model is a living model that can be modified easily to reflect desired changes to the process.
- Unlike flow-charts, it is not burdened with a negative bias from the programming staff.

Figure 7 demonstrates the use of an IDEF model to describe

the dynamic behavior of a software organization. The model defines the interface between functional areas and allocates specific activities to each area. More importantly, the model defines an interacting set of activities that make a horizontal cut through the vertically aligned organization. The success of the software process is dependent upon the success of these horizontal cuts because the results of these cuts are the end-items of the process. Management must understand the importance of this dynamic view to the success of the software process and to the success of the enterprise as a whole. In the past, management has focused on the control of the vertical functional areas and has burdened functional area managers with activity management and employee supervisory duties. This problem is compounded by conflicting incentives that force managers to make unattractive decisions regarding employee growth versus activity accomplishment. It also promotes inefficiencies by rewarding managers for discrete activity accomplishment often at the expense of process efficiency. It also fosters staff growth resulting in decreased productivity per labor hour as managers work to build their functional areas and increase their status within the organization. The new model of management assigns process owners to each individual horizontal process and makes them responsible for the success of that process. This encourages a focus on the end-item and the satisfaction of customer requirements. Most importantly, it focuses managers attention on process efficiency and thus creates an environment that encourages the use of statistical process control principles. Without a process focus, an enterprise finds it difficult to align the goals of the functional area managers with the goals of the enterprise in a way that guarantees success.

An effective management program identifies those processes that are critical to the organization's success, assigns a process owner, establishes performance and cost objectives based on product quality characteristics, empowers the process owner to accomplish the goals, and evaluates the process owner and employees in the process on their progress toward the goals. Written agreements that are reached by consensus between the process owner and management, and subsequently between process owner and employee in the process, become the basis for performance evaluations. This provides confidence that employee goals are aligned with enterprise goals.

Next Steps

This paper has examined some of the basic principles of statistical process control and their application to the software process. Statistical process control has its roots in the manufacturing environment of the early 20th century, however, some simple transformations are available that allow the application to cognition-dominated processes. Variation in the quality characteristics of the end-items of processes dominated by cognition will continue to be much greater than those of manufacturing processes. However, recognition of special versus common causes of variation will enable process owners to make better decisions and as a result bring processes like the software process under control in the statistical sense of the term. In general, the application of statistical process control principles to the software process involves the following steps:

1) Negotiate a set of prioritized software quality characteristics with the customer.

2) Design, specify, and implement a software process capable of producing the desired software product.

3) Establish process owners and empower them.

4) Establish metrics for processes that correlate to the quality characteristics established for the end-item software product.

5) Employ control charting or comparable techniques to determine the stability of each process.

6) Bring processes in control by eliminating all special causes of variation.

7) Continuously improve processes in order to bring control limits within tolerances so that the end-item software product meets customer requirements.

The steps enumerated here closely parallel those developed by product manufacturers to describe a design-for manufacturability program [14]. The application to software is not immediately apparent because of the lack of similarity between the model inherent in these steps and existing planning models. However, some similarities can be denoted. For example, our current approach to steps 1 and 2 typically entails the selection of an existing software process model (rapid prototyping, evolutionary development, or waterfall) based, ideally, on problem domain characteristics but more often based on the model that the developer is most comfortable with. Actually, specific enumeration of steps 1 and 2 encourages the development of new software process models. An actual case study describing the results of following steps 1 and 2 in an unbiased manner is well-documented in [15]. The case study describes the development of the Prototype Ocean Surveillance Terminal (POST), an intelligence gathering and display system used by Navy intelligence analysts.

Intelligence analysts based on U.S. Navy vessels are responsible for tracking and reporting on enemy vessels in

order to provide ship-based officers strategic information vital to the survival of the fleet. Prior to POST, analysts manually correlated intelligence data from shore-based processing systems to create the tracks of enemy vessels. This correlation process was time-consuming and error-prone because much of it was based on intuition and induction rather than straight-forward deduction. In the early 1980's, with the advent of reasonably-priced, rugged, high-performance workstations on the market, a group of individuals, familiar with the problem of ship-based intelligence analysts, began to envision a potential solution. At the time, neither the system developer (BTG) nor the customer (the U.S. Navy) could visualize—to the point of writing a requirements specification—the type of system required by the intelligence analyst. This eliminated the use of the classic waterfall process model. Software process models based on prototyping were available but did not meet the need [15]:

- There was a need to deploy a working system early.
- Only a minuscule subset of overall system functions could be articulated at project initiation.
- The customer wanted the users to play with a working system, provide feedback, and quickly see a new desired capability.
- Product accuracy, robustness, user-friendliness, maintainability (in fact, operational maintainability or the ability to modify while deployed) were the critical quality characteristics.

Stepping back and applying step 2 in an unbiased manner to the quality characteristics negotiated as part of step 1, resulted in the development and implementation of a new software process model later named *operational prototyping* [15]. The operational prototyping process can be summarized by the following steps.

1) Construct an evolutionary prototype using a conventional development (waterfall) process. Specify and implement a small, well-understood, set of the requirements.
2) Deploy copies of the prototype to operational sites.
3) Deploy prototypers with the prototypes to sit with the user and learn the application. The prototypers fix problems that arise and add new functionality that the users request.
4) Record changes that users view as useful and role them into a next generation evolutionary prototype.
5) Go to step 2.

So far in this case study, we have seen the negotiation of quality characteristics (step 1 of applying statistical process control principles to the software process) and the design, specification, and implementation of a software process capable of building the desired software product (step 2). We have also seen a subset of the process owners and the empowerment of these process owners (step 3). We did not see this empowerment explicitly, however, implicit in the operational prototyping process is the empowerment of the prototypers deployed with the evolutionary prototypes. These prototypers are not encumbered by formal change control procedures, management indecision, or resource limitations. They are empowered to serve the user. This empowerment is necessary for the on-site prototyping process to succeed.

Metrics established for POST development did not include the classic metrics of lines-of-code (LOC), errors-per-thousand LOC, or others that did not correlate to the quality characteristics specified as part of step 1. Instead, a limited set of metrics that, over time, showed correlation (step 4) to the quality characteristics were used.

This case study has shown the feasibility of steps 1 through 4 in the approach to the application of statistical process control to the software process. Design, specification, and implementation of a software process model tailored to produce a software product possessing the desired quality characteristics has proven, from experience, to be a valuable part of the approach because statistical process control is not just a measurement discipline. It is a product planning and assurance philosophy that recognizes the variation inherent in all processes.

The high degree of correlation between the software process and the end-item software product results in a process that is considered to be correct. In other words, a correct process is a process that when adhered to will increase the probability of developing an end-item software product that satisfies customer requirements. The prudent enterprise considers some additional process attributes when designing a software process. These attributes include [13]

- Responsiveness. This attribute refers the ability of the process to react quickly to changes in external conditions (e.g., a change in the set of prioritized quality characteristics).
- Flexibility. This attribute is similar to responsiveness except that flexibility refers to the process's reaction to internal factors.
- Cost-Effectiveness. Is the process efficient enough to

be competitive? Does the process minimize direct labor hours?

- Auditability. Schedule and cost pressures tempt circumvention of the process and the potential for introduction of special causes of variation. Process audits ensure that the process remains in control.

The flexibility of a software process affects the competitiveness of the process. Flexibility refers to the ability to adapt and implement new technologies as they become available, to correct flaws by allowing process path modifications, to maintain production even when some part(s) of the process fail, and to adapt to new applications without major modifications.

References

[1] Burgess, John, "IBM Finishes One Race, Starts Another," *The Washington Post*, Washington, D.C., March 31, 1992.

[2] Humphrey, Watts S., *Managing the Software Process*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[3] Humphrey, Watts S., et al., "Software Process Improvement at Hughes Aircraft," *IEEE Software*, July 1991, pp. 11-23.

[4] Cho, Chin-Kuei, *Quality Programming: Developing and Testing Software with Statistical Quality Control*, John Wiley and Sons, Inc., New York, 1987.

[5] Bollinger, Terry B. and Clement McGowan, "A Critical Look at Software Capability Evaluations," *IEEE Software*, July 1991, pp. 25-41.

[6] Shewhart, Walter A., *The Economic Control of Quality of Manufactured Product*, D. Van Nostrand Company, New York, 1931, reprinted by ASQC Quality Press, Milwaukee, Wisconsin, 1980.

[7] Nolan, Thomas W. and Lloyd P. Provost, "Understanding Variation," *Quality Progress*, May 1990, pp. 70-78.

[8] Grant, Eugene L. and Richard S. Leavenworth, *Statistical Quality Control*, Sixth Edition, McGraw-Hill Publishing Company, New York, 1988.

[9] Deming, W. Edwards, *Out of the Crisis*, MIT Center for Advanced Engineering Study, Cambridge, Massachusetts, 1986.

[10] Bowen, Thomas P, Gary B. Wigle, and Jay T. Tsai, *Specification of Software Quality Attributes*, Volume I (of three), Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, New York, 1985

[11] Weinberg, G. M. and E. L. Schulman, "Goals and Performance in Computer Programming," *Human Factors*, 1974, 16 (1), pp. 70-77.

[12] Potosnak, Kathleen, "Mental Models: Helping Users Understand Software," *IEEE Software*, Human Factors, September, 1989, pp. 85-86, 88.

[13] Kurtz, Robert B., et al., *Toward a New Era in U.S. Manufacturing: The Need for a National Vision*, Manufacturing Studies Board, Commission on Engineering and Technical Systems, National Research Council, National Academy Press, Washington, D.C., 1986.

[14] Harry, Mikel J., *The Nature of Six Sigma Quality*, Motorola, Inc., Government Electronics Group.

[15] Davis, Alan M., "Operational Prototyping: The POST Story," to be published in *IEEE Software*, August, 1992.