Silver Bullet: An Iterative Model for Process Definition and Improvement

Stacey Gelman

AT&T's competitive advantage depends upon its capacity to respond to customers' escalating demands for high-performance, high-quality, and timely software products. An adaptable, robust software development process is a key means to succeed. The Silver Bullet project was designed to show the Operations Systems Business Unit how to create and adapt processes that employ the most productive technologies to help project teams serve their customers. Our insight was to build upon the concept of a customer-supplier partnership between a small group of full-time process engineers (the supplier) and the product development teams (the customers). The iterative model for process definition and improvement, which is described here, emerged as we institutionalized processes to facilitate the creative dialog between these two groups.

Introduction to Silver Bullet

Silver Bullet[1] was launched in August 1990, in the Operations Systems Business Unit (OSBU), to decrease the time intervals in software development. Chartered with changing software development through a process-driven approach to product development, Silver Bullet has evolved from a concept to a fully operational stage.

Silver Bullet has resulted in a functional, customizable *process product* that describes to a specific customer a systematic series of steps, or processes, to organize the processes they use to accomplish their business objectives. The customized product includes an extensive and complete software package, plus detailed written guidelines, training, consultation, and other tools, on how users should follow the processes identified by the customer to develop an operations systems product.

One important aspect of Silver Bullet is the physical separation of the creators of the process product from those responsible for its application. This physical separation ensures that each organization can concentrate on its set of responsibilities, in order that objectives and lines of responsibility will not be diluted. Thus, the Software Technology Center (STC) is responsible for *developing* the process product for its customer, the Advanced Software Construction Center (ASCC), which, in turn, is responsible for *using* the process product to build operations systems products. This separation does not mean an "over-the-fence" hand-off of the process product. On the contrary, a continual, multi-level, two-way feedback between the process designers and the process users is integral to Silver Bullet.

The term *process product* includes the process definition and the description of how the customer does its business. It also includes the supporting elements needed to assure that the process or processes that have been defined can be effectively employed, not just by the customer's product developers, but by other functions, such as technical support and end-user customer feedback. While Silver Bullet was set up as an experiment, we wanted it to be a field experiment, not a theoretical exercise. As a result, we designed the process product in such a way as to encourage its use in a live, software development environment.

The essential component to achieving a sustained process application for an organization is the ability to return to and quickly change the process definition as the situation demands. Understanding that the initial

Panel 1. Acronyms and Terms Used in This Paper

ASCC—Advanced Software Construction Center
KLOC—Thousand lines of code
MR—Modification request
OSBU—Operations Systems Business Unit
SABLIME—Software tool used for tracking modification requests (MRs)
SB 1.0—Silver Bullet process generic Release 1, etc.
SDE—Software Development Environment
STC—Software Technology Center
T—Testing tool designed to improve the productivity of test-case design.

process definition would be neither complete nor static, our strategy was to get the process definition into use quickly, and take advantage of our customers' experiences to improve and extend it. Thus, an iterative model for process definition and improvement was developed, based on our understanding that:

- Feedback from the customer (in our case, the ASCC) is crucial to improving the process definition. While previous experiences are important when creating an initial process definition, a rigorous, complete, and effective process will result only through continual use.
- The ability to be responsive to process problems is crucial to success. We wanted to be able to maintain credibility by responding quickly to a customer's feedback, and to ensure that the processes are not bypassed by users when a problem or error is found.
- Close coordination between the project plan and the processes used to accomplish that plan is critical to the processes being used. Project decisions are made through project plans. A project manager must be able to use the process product to generate and update plans, and the process users need information to execute these plans.
- The fact that a process is defined does not mean it is static. The process needs to adapt to changes in the environment and to incorporate new technology to create a sustained advantage. Objectives, such as reducing the time interval in developing a product, may require complete shifts in process.

By treating process as a product, and the process users as our customers, we have been able to successfully work with the ASCC to employ our process product on software projects, gain feedback on the effectiveness of these processes, and substantially improve the processes. Indeed, we are looking at two sets of processes—those used to develop the Silver Bullet process product and those user processes that were identified by the customer in the process definition. The specific process-iteration model that we use has been developed, applied, and grown over the past several years in our work on Project Silver Bullet. The remainder of this paper addresses this model and some of our experience in using it.

Process-iteration Overview

One of the challenges in process engineering is to efficiently define, introduce, improve, and re-introduce process definitions to the customer. Just as a software product goes through stages in its life cycle, a process goes through a similar cycle. A product moves from concept to definition to implementation to deployment and on-going support. One aspect of deployment is the continuous evaluation of the product to understand what new or enhanced capabilities users need. Following this software product metaphor, there is a systematic way of developing a process that parallels the development and deployment of software products. To support the development of the process product, we developed the model for process iteration shown in Figure 1.

The five phases of the model represent the stages required for an organization to apply and sustain the use of process to develop products. This model is based on the continuous improvement of the process definition, and the ability to introduce a major change in it, if required. As in software products, a major release of the process is referred to as a generic (numbered generic 1.0, 2.0, etc.). Improvements made in a major release are point releases (numbered generic 1.1, 1.2, 1.3, etc.).

Major releases of the process are produced in the *define* phase. The process that is defined depends on the objectives of the organization and the environment within which the process will be used. Objectives could include interval, cost, and quality. The definition of the process must take these objectives into account and optimize the process to meet them. The environment variables include available methods, tools, the type of organization that will be using the process, and the type of products that will be built. A process defined for operations systems products, which are transaction-based, will

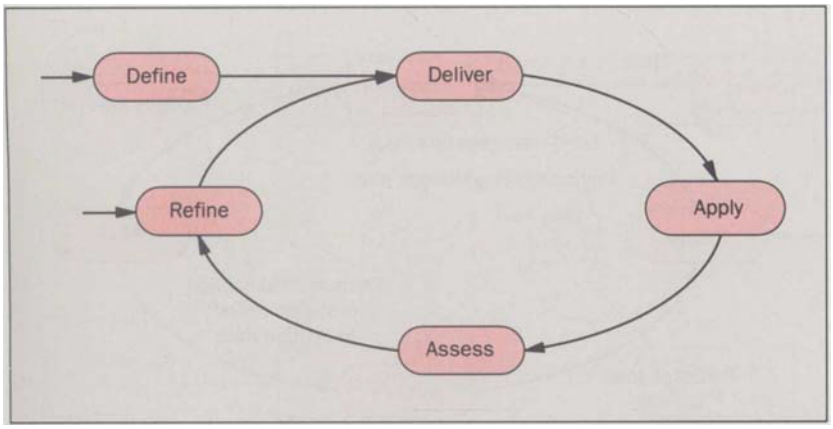

Figure 1. The five phases of the Silver Bullet model represent the stages that are required for an organization to apply and sustain the use of process to develop products. The *define* phase includes establishing the customer's objectives, and the processes used to accomplish them; the *deliver* phase prepares and packages a new Silver Bullet release for use by the customer; the *apply* phase involves using the process product; the *assess* phase evaluates the information gathered during the apply phase; and the *refine* phase includes improving the product.

differ from a process defined for building real-time systems. A change in the objectives or any of the environmental variables could lead to the development of not just an improvement (generic 1.1), but of a new generic (2.0).

The *deliver* phase prepares and packages a new release of the process for installation in a customer's design and development environment. As with a software product, the process definition must be "loaded" into a customer's work environment.

There are many challenges with delivering a process to an organization. The deliver phase must take into account that several versions of the process exist simultaneously, and that people must be able to determine the impact of a new release on their current project schedule. For example, Silver Bullet for the ASCC is operating on its ninth release of the process, SB 1.8.

The deliver phase also must take into account that there are multiple copies of the same release of the process in existence—one that the process engineers use for improvement, and others for product development. These process definitions must be synchronized to enable meaningful communication and improvement of the processes.

Getting the process into use to build software products, and gaining information about the use of the process, occurs during the *apply* phase. This phase is where choices are made by the development organization, in our case the ASCC. These choices have an influence on the use of the process on a particular product. It is our objective to influence the users of the process to make the right choices and to collect information needed to improve the process.

There are a variety of users within an organization. While people who execute the process require information about how to do their jobs, they also require schedules that tell when to do their jobs. Training, both formal classroom training and hands-on experience, is critical to a user's ability to execute processes efficiently. During the apply phase, training must be scheduled and jump-starts, that is, intense consultation of the users with appropriate experts, must be made available.

For continuous improvement, data must be collected to measure the application of the processes on specific products. The approach taken on Silver Bullet is to develop several feedback loops that enable the process execution to be examined in multiple dimensions, enabling cross-correlation to validate the data collected.

The *assess* phase is used to evaluate the information gathered during the apply phase to drive process-improvement activities. Such activities focus on improvements that influence the performance of a process user. This includes:

- Evaluating the appropriateness of training,
- Evaluating the usefulness of software platforms, and
- Improving the process or task flow.

On the Silver Bullet project, the software platforms in use are the BaseWorx™ Application Platform[2] and the Software Development Environment (SDE) from the Software Technolgy Center. During the assess phase, the data received on the processes are evaluated, work items are assigned to process engineers, and progress is tracked.

The *refine* phase enables the process developers to improve the current release of the process, based on user feedback during the apply and assess phases. These improvements may also incorporate changes to the

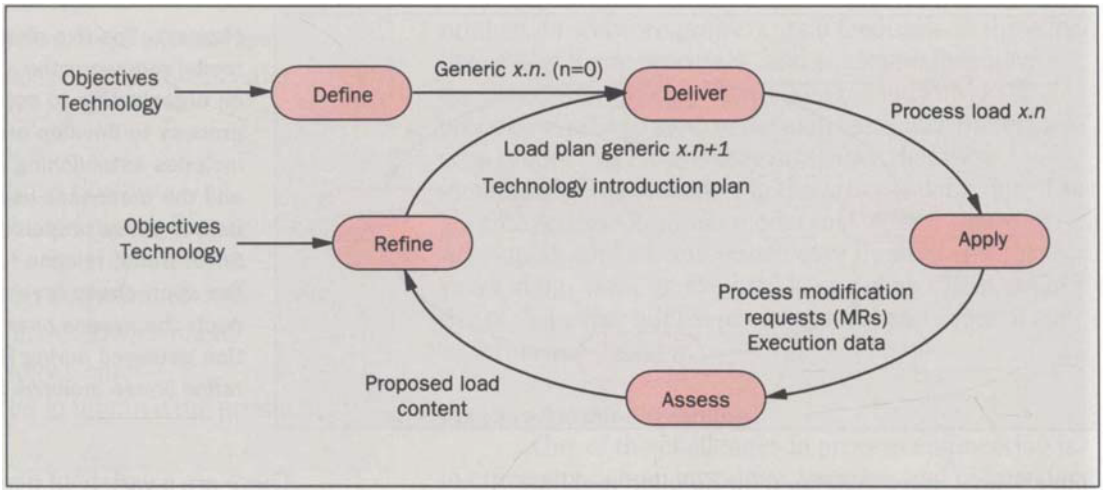

Figure 2. The process-iteration model includes steps that must be taken, called key work products, between each phase of the model. Once the initial release of the process product is made, all these activities go on simultaneously during the course of the product development life cycle.

software platforms. During the refine phase, process engineers operate at the process-definition level, making changes to the current generic based on the analysis of the information gained during the apply and assess phases. These changes to the generic are similar to point releases in software products.

Figure 2 shows the process-iteration model with the steps that must be taken—called "key work products"—between each phase of the model. Once launched with the initial release of the process, all these activities go on simultaneously during the life cycle of the product development.

Taking a Closer Look

In this section, we will discuss each of the five phases from a work-flow perspective, including the specific activities that take place during each phase.

Define The Process. The *define* phase of the process-iteration model uses as input:

- The objectives that one is trying to satisfy through a defined process, and
- The available technology, or emerging technology, that will help achieve these objectives and produce a defined process.

Before the process can be defined, the process representation must be specified.[3] The process representation describes the output, that is, the information to be presented to the process user.

To specify the process representation, the information needs of the process users are analyzed to understand how best to represent the process to meet these needs.[4] These users include:

- Process engineers who must define, track, and improve the processes,
- Managers who must hire and train staff,
- Project managers who must establish project plans and manage to meet the schedule and resource commitments,
- Staff members who must use the processes to do their jobs, and
- Providers of platforms and tools.

The core for the process representation is the *task definition*, which communicates what job needs to be done, and what output must be produced. This is represented in the process definition as data-flow diagrams, with supporting task specifications and templates. Figure 3 is a sample of the processes, showing the use of data-flow diagrams to represent information flow. In this case, the task "Test and debug unit" requires two inputs: "Inspect custom code" and "Unit test plan." Outputs include the test code and unit test release notes.

The task specification is the underlying information about each individual task. Table I shows an excerpt of the task specification for the "Plan unit test" function. Each element in the description is essential for the users of the process definition. Although the information that is represented in the task specification is detailed, our experience is that this level of detail is necessary for an organization to execute the process. It tells project managers what staff is required, what training they need, and

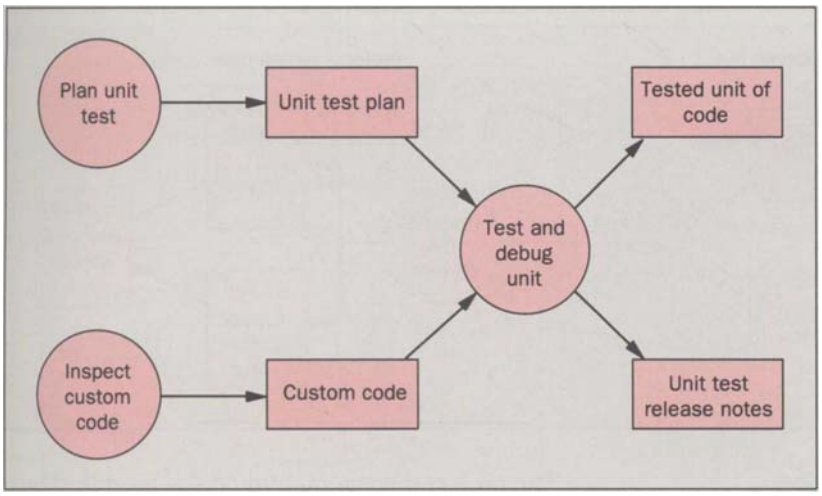

Figure 3. The *define* phase uses as input the objectives and the available or emerging technologies.

Table I: Task specification for "Plan unit test"

Enabler	Content
Task	Plan unit test
Training	Unit test workshop
Template	Unit test plan template
Architecture platform component	BaseWorX™ component: None applicable
Quality assurance	Entrance criteria: Complete subsystem design Exit criteria: Complete unit test
Staff	Skill/experience level specification: B.S. in computer science
Tool component	FrameMaker* (for template)
Formulas	Estimated duration: 1 person for 1 week per thousand lines of code (KLOC)

how long it should take to execute the task through specific parameterized formulas. It tells process users what training they need, what platform and tools are applicable, and what entrance criteria must be met for them to begin execution. The task specification also gives the users the target output in the form of a template, and it tells them what exit criteria must be met to satisfy the requirements of the next task(s).

From a process engineering perspective, each element of the process representation is a potential area for process improvement. This includes the addition of new tasks and the deletion or update of existing tasks. Changes that take place are at the level represented in Figure 3 (the data flow), Table I (task specification) and specific templates (such as the template for the unit test release notes, which is not shown in this paper). Data captured during the apply and assess phases determine the:

- Appropriateness of training and staffing,
- Applicability of templates,
- Usefulness of platform components,
- Accuracy of the formulas, and
- Ordering and appropriateness of particular tasks.

The need for this level of detail drives the need for precision in the process representation, as well as

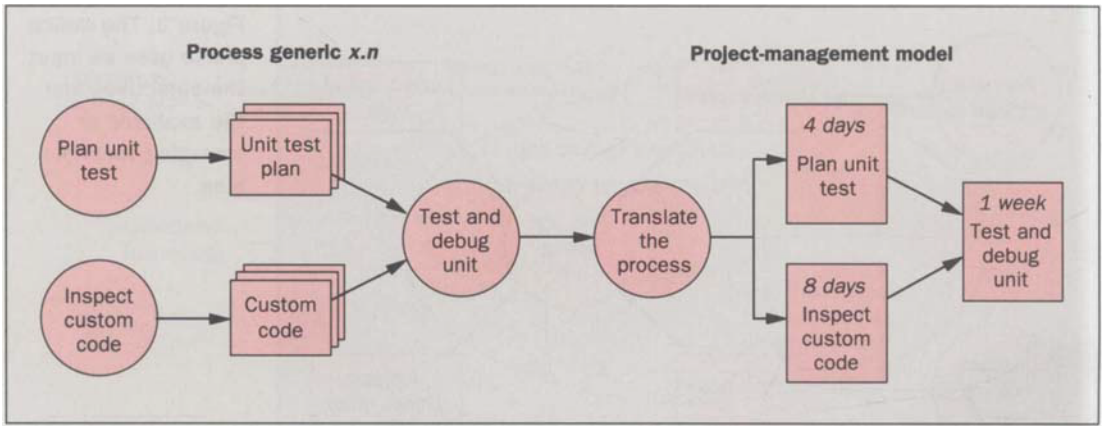

Figure 4. Here is a sample of a data-flow diagram representing information flow. In this case, the task, "Test and debug unit," requires two inputs: "Inspect custom code" and "Unit test plan." The output is translated into time parameters to create a schedule.

the choices of what data is collected to drive process improvements.

Deliver The Process. The *deliver* phase packages the process information for application by the users. This phase must take into account the fact that several versions of the process exist, and that users need to know the appropriateness of using a new generic relative to their own position on a particular product. For example, if a new testing process is introduced that requires information during the analysis phase, it cannot be used on a product that has already passed this phase. The deliver phase must be able to clearly communicate information that has changed, and the impact of these changes.

The package of information produced during the deliver phase is communicated to the process users through a *process load* that has a generic number associated with it (e.g. load 1.1). The load contains the current release of the processes (data flow diagrams, task specifications, and templates), and a clear statement describing the content of the release to enable users to understand the potential impact of changes on their project schedules. A specific problem with a process can readily be traced to the particular load in use.

In addition to the information given to the process users, the load also contains information that managers need to plan a project using the processes. It is during the deliver phase that the processes are put into a state that will enable them to be tracked within the context of project-management models. Each process step has a parameterized time-to-execute formula. These formulas, project parameters, and flow dependencies from the processes are used to generate a project-management model.

The project-management model is produced by applying appropriate time parameters to the tasks from the data-flow representation. The built-in process formulas use these parameters to create a schedule. For example, the task "Plan unit test" (shown in Figure 3) is translated from its data-flow representation, in process Generic *x.n*, to a four-day task in the "Project-management model" (shown in Figure 4). The project-management model is a part of the delivery to the process users that is included in process load *x.n*.

The deliver phase is an important quality assurance step for the processes themselves. The translation to a project-management model enables consistency checking to be performed.[4] It also enables the current generic to be evaluated relative to interval goals. Since one of the goals of Silver Bullet is decreasing the total software-realization time interval, each process generic is monitored to assess its impact on the time interval. The addition of tasks can result in a longer interval and, thus, must be evaluated within this context.

Finally, the training curriculum is a key part of the load delivered to the process users. In addition to communicating what has changed, there are times when retraining is necessary to ensure a successful process introduction. Potential retraining is considered when either a new technology, or a completely new process capability, is introduced. An example is the addition of reverse engineering processes that are not a part of the original process definition. While new technology is introduced in the context of a *technology introduction plan* (discussed in "Redefine The Process" below), the delivery mechanism must ensure that the training curriculum is updated to reflect the need for any additional training.

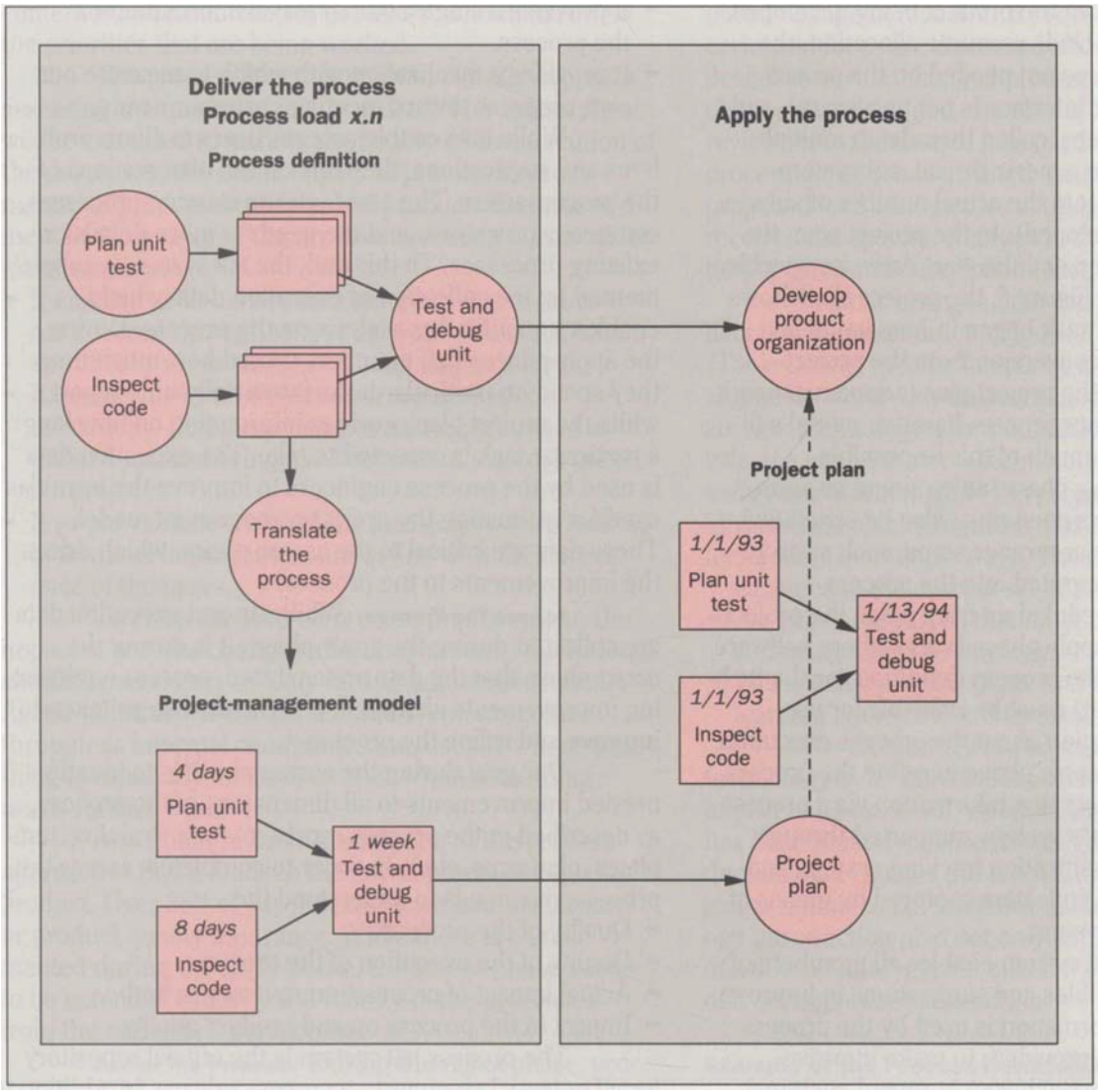

Figure 5. The *deliver* phase delivers to the customer the process load, which includes the initial product Generic *x.n* instructions for its use, and any information that has changed in subsequent generics. Inputs include the generic, the load plan, and the technology introduction plan.

Apply The Process. During the *apply* phase, the processes are used to develop software products and data is gathered to improve the processes.

Our direct work with the ASCC was essential to understanding how to provide processes that can be put into practice. Since project decisions are made through project plans, the closer the connection between the processes and the project plan, the higher the probability of the processes being used. A project manager needs to use the process to generate and update project plans, and a process user needs information to execute these plans.

Figure 5 shows the process users, represented as "Develop product organization," receiving two pieces of information from the deliver phase:

- The *process definition*, that is, the tasks, task specifications, and templates that indicate how a job is to be done.
- The *project plan*, which tells when a job is to be done and who is to perform it.

The project plan is produced during the apply phase by taking the project-management model produced during the deliver phase and customizing it to

meet the specific needs of the product being developed. These specific needs include resource allocation, the exclusion of process steps not needed on the project (e.g., if a graphical user interface is not necessary), and the binding of single steps, called threads, to multiple threads (e.g., taking the generic thread, subsystem design, and mapping it into the actual number of subsystems that are to be developed). In the project plan, the project start date is given and the start dates for specific tasks are calculated. In Figure 5, the project plan shows that the "Plan unit test" task began in January 1993.

Currently, the conversion from the project-management model to the project plan is a manual operation. Improvements to our process-iteration model will focus on automating as much of this as possible.

During the apply phase, any training or jump-starts that staff members need must also be scheduled. Specific product quality assurance steps, such as architecture reviews, are integrated into the process definitions, and are scheduled up front when the project plan is produced. The apply phase is also where software platforms specified by the process definition (or the technology introduction plan) must be available for use.

Critical information about the process execution is gathered during the apply phase to refine the process. We have chosen to collect that information via a process modification request (MR) system, supported through SABLIME, a software modification tracking system, and through execution-data collection captured by means of a daily time-reporting system.

The process MR system enables all members of the project to enter troubles and suggestions to improve the processes. This information is used by the process engineers and platform providers to make improvements. The MR system was chosen to record customer feedback for several reasons:

- It encourages customer involvement. The ASCC staff directly accesses the MR system for entry of troubles, status, and reports.
- It reduces the need to learn a new methodology and develop a new tool. Software developers are familiar with the use of MRs and SABLIME. The use of MRs to support process changes is an extension of the SABLIME application.
- It eliminates losing information, as all troubles and suggestions are invaluable to understanding how the process is working.
- It provides a single conduit for all information on the process.
- It provides a mechanism with which to measure our own progress toward meeting customer needs.

While MRs enable process users to share problems and suggestions, they rely on the observations of the process users. The MRs focus on missing processes, erroneous processes, and the need for more details on existing processes. To this end, the MR system is supplemented by the collection of execution data, which enables a quantitative analysis on the process. During the apply phase, staff members record how much time they spend on particular tasks into a daily time report, while the project plan provides information on how long a particular task is expected to take. The execution data is used by the process engineers to improve the formulas used for estimating the project-management model. These data are critical to the assess phase, which drives the improvements to the process.

Assess The Process. While MR and execution data are collected during the apply phase, it is during the *assess* phase that the data are analyzed, process engineering improvements identified, and the next steps taken to improve and refine the processes.

Our goal during the assess phase is to identify needed improvements to all dimensions of the process, as described in the process representation (training, templates, platforms, etc.). In order to completely assess the process, one needs to understand the:

- Quality of the process,
- Quality of the execution of the process,
- Actual impact of process improvements, and
- Impact of the process on end-product quality.

The process MR system is the official repository for all potential changes to a process release. In addition to the direct entry of MRs by process users, MRs are generated by the results of any process improvements identified during postmortems, and root-cause analysis of execution data. The MRs drive the priorities and work program of the process engineers. The priorities come as a direct result of input from the users who have initiated an MR, assessment by the process engineers in terms of impact, and input from managers in terms of project-level priorities. The MRs are also the basis by which process engineering work is communicated to the users. Once prioritized, the MRs are scheduled into loads, which mark when a process improvement will be available for use. The proposed load

content is communicated to the users, who then review the priorities that are being worked.

The process-iteration model currently supports assessing the quality of the processes. The assess phase will be extended to include the quality of the execution of the processes, the actual impact of process improvements, and the quality of the end-product. The assessment of the quality of the process execution is critical for many reasons:

- It enables us to understand other aspects of process not covered through MRs or execution-data collection, specifically areas such as training and skill level.
- It enables us to validate the correctness of the execution—that the processes are being applied as defined and are being used consistently across multiple projects.
- It would validate whether the decisions we are making in terms of improvement are in line with the experience of the users.

To close the loop on process improvement, the impact of process changes after they are fully operational should be assessed and any further adjustments made. At this point, any feedback on improvements comes through as informal communication, usually when something is especially liked, or as MRs, when something needs further work.

And finally, a basic assumption of the process approach is that it leads to the development of a quality product. Over 35% of our processes are tied to software or product quality assurance. While these are implemented during the apply phase, the assess phase needs to be extended to verify the validity of the processes from the end-user's perspective.

Refine The Process. During the *refine* phase, process engineers work on the prioritized MRs established during the assess phase, and communicated through the proposed load content. The refine phase takes the outputs of the assess phase, the objectives, such as interval reduction, and the available technology, and updates the current generic of the process. The updates made during the refine phase parallel point releases in software product development.

The process engineer begins by analyzing the available data to determine how best to make the improvement. The data analysis uses the information that exists about potential problems. This includes the specific MR, and any execution data that is appropriate. If the proposed change comes from a process user, the user's concerns and ideas are discussed and serve as input to the design of new or changed processes. As the process changes are designed, the user is again involved in the review and validation of proposed changes. As a result, the process definition is updated. As changes are made, the proposed load content also is updated. The delivery phase receives the specific changes in the form of process definition updates and descriptions that detail the changes.

Process changes take the form of updates to the data flow representation (Figure 3), task specifications (Table I), and specific templates and technology. Process engineers work with the technology providers to evaluate new technologies for potential impact on time intervals. If a need for a new technology is discovered, a *technology introduction plan*[5,6] is prepared. The technology introduction plan recognizes that a user is impacted by technology in multiple ways. These include:

- Changes to the current set of processes,
- Assigning resources that must be applied to learning and using a new technology.
- The specific product plans that must account for the learning curve of a new technology.

The technology introduction plan allows new technology to be introduced in a way that enables the impact to be assessed, isolated, and measured. If there has been limited experience with a technology, the introduction plan enables the establishment of a control project to minimize potential downside impact. The technology introduction plan not only supports the initial setup of a new technology, but also the ongoing management of it during process execution.

Example of the Process-Iteration Model

Experience to date with the ASCC has enabled us to go through the process-iteration model nine times. This example illustrates how the model facilitated the identification of a process bottleneck and helped us select a new technology to eliminate it.

In the midst of establishing the initial process generic for the ASCC, we evaluated a testing tool, called T,[7] that was designed to improve the productivity of test-case design. At that time, we decided not to use T, as:

- There was little experience using the tool within AT&T.
- We were evaluating a tool without the benefit of an existing generic.

The initial load SB 1.0 was delivered without the T tool. During execution of load 1.0 on the 5XPRESS™ project, an MR was entered against the processes, indicating that test-case design was an intensely manual operation, and that the problem could be solved with an appropriate technology. In addition, the execution-data analysis pointed out that more time was being spent on test-case design than was originally expected. This led the MR review board to recommend, within the context of the existing generic, a re-evaluation of the T testing tool.

The concept for a technology introduction plan grew out of this work. There was a desire to move forward with T, but there was also a recognition that, to be successful, the full impact must be assessed and managed. During the evaluation of the T testing tool, we found that, within AT&T, practical experience with the tool was still lacking. We determined that a controlled introduction with the use of a control project was the most sensible approach. The T technology introduction plan was produced, and the decision to move forward was agreed to by both the ASCC and the STC. We selected two projects under development at the ASCC as part of the technology introduction process. The first project is the application project (using T), and the second project is the control project (not using T). Both projects are being executed in parallel and, therefore, provide us with timely data to assess whether or not T should be widely deployed throughout the ASCC.

This example not only points out how we use the model to improve the processes and enabling technologies, but also how the same set of processes can be used on different projects in different ways. This is done during the apply phase, when the project-management model is translated to product-specific project plans.

Other Experiences

While the application of T illustrates a specific use of the process engineering model, the model itself has permitted the introduction of a multitude of process changes to the ASCC over the past two years.

Through the various iterations of the process-iteration cycle, there is a marked difference between the generic SB 1.0, introduced more than two years ago, and the generic (SB 1.8) that is operational today. The changes to the processes can be attributed to processes that were missing, processes that were corrected, and processes that expanded to better detail. All of the improvements can be attributed to the ASCC's use of the process-engineering model to identify potential problems as products were built, as well as to the process engineers who, working with the ASCC, assessed this input and refined the processes.

Without the five phases of the process-iteration model as it exists today, it would not have been possible to reach this level of iteration. The lapse between load SB 1.0 and load SB 1.0.1, which was 10 months, indicates that we didn't start this work with the correct mechanisms in place to promote fast iteration. The process MR system was first introduced, in January 1992, to ensure that all needed changes were captured and trackable. The project-management model was introduced next, in May 1992. The formal process-delivery mechanisms were introduced in June 1992. These capabilities enabled us to become operational, and to begin iterating quickly on the input that we were receiving from the ASCC. Once these capabilities were established, we were able to focus on improving such activities as employing new technology through technology introduction planning, which occurred in October 1992.

Summary

Silver Bullet seeks to achieve a sustained competitive advantage in the rapidly changing software product market through a process-driven approach to product development. One key to our approach was to recognize the distinction between designing a software process and the application of that process product to an organization that will use it to develop software products.

By treating the process as a product, with its users as our customers, we developed a process-iteration model to enable a systematic approach to the development and improvement of the process. Our work in this area not only focused on the design of mechanisms for improving the processes, but also on mechanisms to ensure that the processes can be readily used to build software products.

A vital element of our approach is the continuous dialogue between those who design processes and those who execute them, thus focusing process improvements on the real problems faced by a competitive organization. The process-iteration model is designed to support a high rate of process improvement.

As discussed in this paper, we can demonstrate the success of our approach through:

- Participation of all ASCC members in executing the processes to build software products;
- Participation of all Silver Bullet project members in either supplying, or effectively using, process feedback;
- The resulting high rate and quality of process changes made in response to the ASCC's needs; and
- The rate at which we have learned about the principles, practices, and pitfalls of software process engineering and management.

Acknowledgments

The work represented in this paper reflects the thinking and contributions of the Silver Bullet team, specifically Phil Brown, Bob Brownlie, Kathleen Culver-Lozo, Mark Dennis, Rhoda Gordon, Ken Hackbarth, and Jim Striegel. We also thank the members of the ASCC for their partnership on this work.

* FrameMaker is a registered trademark of Frame Technology, Inc.

References

1. Gelman, S. J., Lax, F. M., Maranzano, J. F., "Competing in Large-Scale Software Development," *AT&T Technical Journal*, November/December 1992, pp. 2-11.
2. Beck, R. P., et.al., "Architectures for Large-Scale Reuse," *AT&T Technical Journal*, November/December 1992, pp. 34-46.
3. Culver-Lozo, K. and S. Gelman, "A Process Definition Methodology For Software Development Organizations," *Proceedings of The Seventh International Software Process Workshop*, 1991, pp. 54–56.
4. Brown, P. E., and K. E. Culver-Lozo, "A Methodology and Tool for Supporting Process Driven Software Development," Interactive Development Environments (IDE) Users Group, 1992, Tab 5, pp. 1–22.
5. Poston, R. M. and M. P. Sexton, "Evaluating and Selecting Testing Tools," *IEEE Software*, May 1992, pp. 33–42.
6. Mosley, V., "How to Assess Tools Efficiently and Quantitatively," *IEEE Software*, May 1992, pp. 29–32.
7. Poston, R. M., "A Complete Toolkit For The Software Tester," *American Programmer*, April 1991, pp. 28–37.

(Article approved February 1994)

Stacey Gelman *is head of the Software Process and Ware house Application Process for the Network Systems Chief Information Office in Whippany, New Jersey. Prior to this, she was responsible for leading the team that engineered the process capability used on Project Silver Bullet. She has a B.A. degree in computer science from Queens College in Flushing, New York, and an M.S. degree in computer science from the University of Pennsylvania in Philadelphia. She joined the company in 1979.*