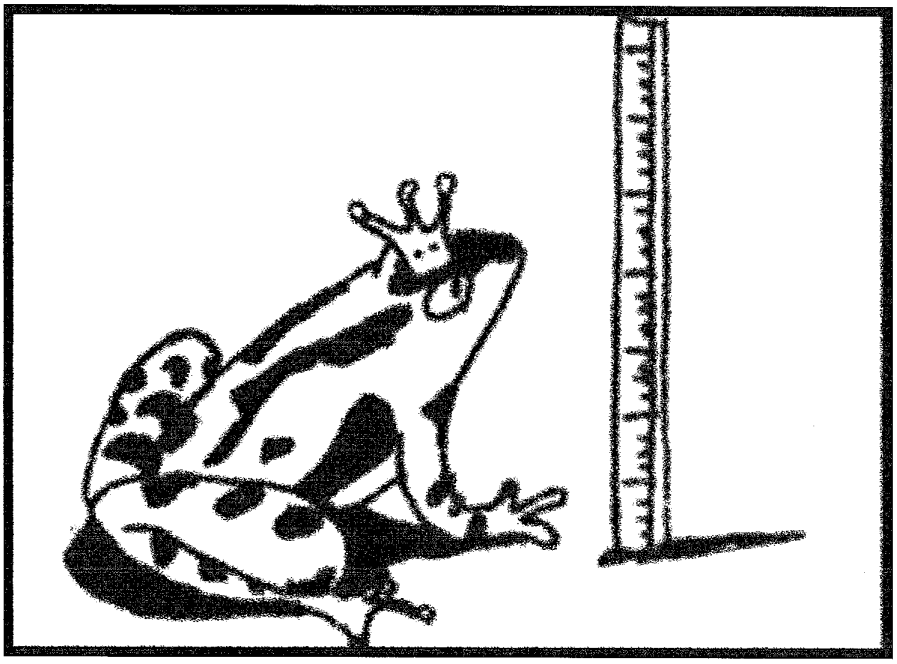
Measuring Reuse: A Cautionary Tale

Many organizations look to reuse for large improvements in quality and productivity. But measuring reuse is difficult, to be attempted only after careful analysis of measurement and management goals. This article describes how to measure reuse and provides warnings on how some numbers can be misleading.



SHARI LAWRENCE PFLEEGER,
Systems/Software

Reuse offers great potential for improving software development productivity. As reuse becomes a mature, tested approach supported by automated tools and repositories, organizations are establishing reuse programs to support their business needs. Recent issues of *IEEE Software* have explored the state of practice and research,[1,2] and Martin Griss and Marty Wasser have reminded us that effective reuse requires neither widespread implementation nor sophisticated automation.[3]

Measurement is important in any reuse program, both to assess its success and to guide its progress. To tell whether reuse is successful, potential reusers want to measure the amount of reuse and its effect on software products and processes. Based on my experiences with several reuse projects, I suggest that you consider several important problems when you plan and implement reuse measurement. These lessons, pointed out as key considerations for managers and technical staff, are accompanied by several cautions; without care, reuse measurement can be misleading, derailing a reuse program rather than guiding it.

0740-7459/96/$05.00 © 1996 IEEE

ONCE UPON A TIME

I begin this tale of reuse by describing the Amalgamated Company, a fictional corporation that is a composite of the real-life organizations, government agencies, and private companies whose lessons this story reflects. I've hidden the identities of the key actors, but every activity and decision described here is real. I present these events so that future reuse programs can learn and grow.

Amalgamated is a very large organization, composed of many divisions, each of which has its own culture. In this regard, Amalgamated is not unlike many large companies that expanded by acquiring other companies. In the past, Amalgamated has attempted to unify its disparate computing organizations by imposing standards for process, programming language, and purchasing. Some parts of the company have interpreted these standards liberally, though, so the unification is only partial.

Amalgamated's main business function is not information technology, but most of the business depends heavily on computing for support. In the early 1990s, Amalgamated hired a new vice president of information technology to complete the unification and save the corporation several billion dollars by the turn of the century. The vice president noticed a lot of duplication in the company: Each division had its own payroll scheme, its own purchasing organization, and its own programming staff, for example. He decided to focus on reuse as the primary method by which to streamline and save the company money.

Definitions and goals. His first two steps were in the right direction: He established an organization to oversee the implementation of reuse, and he set targets for the turn of the century. In briefings held throughout the company, he promised that Amalgamated would eventually have 80-percent reusable code and 100-percent reusable data. At these briefings, employees raised many questions, such as what he meant by "100-percent reusable data." He responded that various parts of the company used different terminology for the same item; payroll was called "pay" in one program, "salary" in another, and "wages" in a third, for instance. By consolidating and using consistent terminology, many programs could reuse the same data. Someone else noted that targets of 80-percent reuse were rarely met, and then usually in organizations that always built versions of the same kind of software. The vice president's response: They would meet the target because their jobs were on the line.

Similarly, employees wanted to know what he meant by "reuse." Did it mean code reuse, or did it include requirements, designs, documentation, test data, or even experience? The vice president decided to focus initially on code reuse, with a longer-term goal of broadening the definition once this was clearly established. But still, people asked, did reuse mean reuse in entirety, or could they modify the code for new applications? Here, his answers were never clear.

The company was already doing some reuse on a limited basis. In addition to informal reuse, where programmers borrowed code or test data from each other, several divisions had set up small reuse libraries, complete with automated support. But each library focused on the particular needs of its division, and no one tried to coordinate. As a result, you might find a bubble sort in several libraries.

The vice president established a group to measure the amount of reuse, so that he could determine progress against his stated goals. When the measurement group asked for more specific goals, his staff provided four:

- improve the quality and reliability of software,
- provide earlier identification and improved management of software risk,
- shorten system development and maintenance time, and
- increase productivity.

But when the group questioned further about how these goals related to reuse, the real goals of the program turned out to be

- save as much money as possible,
- expand the use of reuse libraries, and
- increase the overall level of reuse in the company.

Notice the difference between these two sets of goals. Each was measurable, but the second set reflected the expectations of the board of directors and was expressed in terms that they could understand.

The vice president's strategy for implementing reuse involved four activities:

- produce reusable components,
- create a mechanism for storing and retrieving reusable components,
- use reusable components in new or changing programs, and
- increase the use of reusable components.

Each library focused on the particular needs of its division, and no one tried to coordinate.

Each activity can be interpreted in light of the goals. For those trying to embed measurement in the reuse process, the revised goals were much more straightforward and easy to interpret as measures.

Managerial lesson 1. Express goals clearly, realistically, and in an organized way so that your staff knows which goals come first. Describe the political context for business decisions

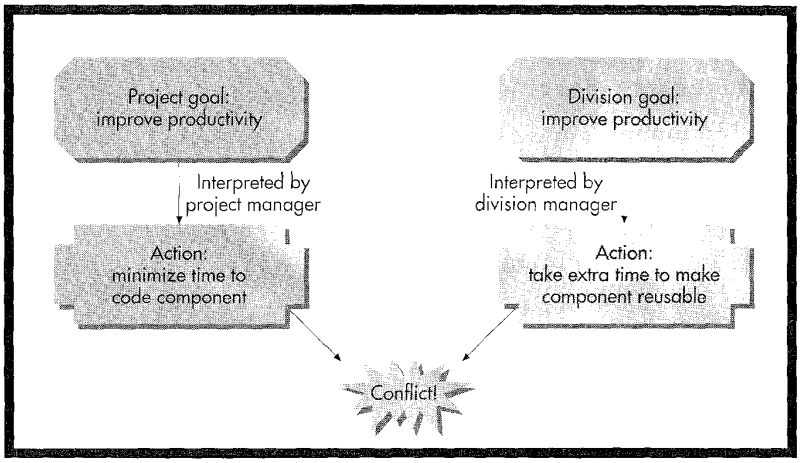


Figure 1. *Example of conflicting interpretation of goals.*

that measurement will support, so that you can devise a strategy for meeting the goals and for presenting the results both to the people who set the goals and to those affected by the goals.

Technical lesson 1. Express the goals in a measurable way so that you can gauge where you are and where you want to be. Understand the political context of technical decisions so that you are prepared to make compromises or weigh alternatives.

Eye of the beholder. Many measurement programs follow the Goal-Question-Metric (GQM) paradigm first introduced by Victor Basili and David Weiss.[4] While goals are necessary, the GQM process notes that stating goals is not enough to start a measurement program. Figure 1 shows how different perspectives can sometimes lead to conflicting interpretations of the same goal. Thus, you must associate with each goal the point of view of the person or organization setting it. Ultimately, the measurements will support actions that are driven by goals, perspectives, and priorities in concert, not by the goals alone.

Suppose our goal is to evaluate the reuse that Amalgamated is already doing, and we start by examining a division's reuse library. If we ask a programmer to tell us how good the library is, chances are that she will respond in terms of the retrieval time, the variety of functions available, and the number of languages supported. But a corporate executive will answer the question very differently, talking instead about increases in productivity and the cost to maintain the library. This difference in perspective, caused by the differing goals and motivations of the participants, is important, and any reuse measurement program must consider it.

We must remember that different perspectives can generate different questions. Whereas programmers ask specific technical questions, such as "Is this sort routine written in C++?", a vice president focuses on the corporation as a whole and its position in the marketplace, asking, "Are we saving money yet?" Even when the question is the same, we may get different answers. As we saw in Figure 1, when a programmer asks how she should write a module the project manager, driven by project schedules, might emphasize speed of coding; yet the division director, driven by corporate goals and personal bonus incentives, might prefer the programmer to spend extra time to make the module reusable. Different points of view reflect the different priorities of the participants in the reuse program.

What points of view do we need to consider when creating a reuse program? Most organizations have at least five: the programmer, the project manager, the department head, the division director, and the corporation. Their motivations and typical activities differ.

Programmers work on a specific project as members of a development team, motivated by the project goals, budgets, and schedules. Here, "programmer" is a generic term encompassing all development or maintenance functions affected by reuse: specifying, designing, coding, and testing a system until it is ready for release. It is the programmers who create components for reuse by others, and who use components developed for previous applications.

Project managers supervise a single project. They integrate reuse into the development and maintenance processes, explaining to the programmers where reuse fits in their day-to-day activities. At the same time, project managers identify components as candidates for reuse on subsequent projects. In addition to the project goals, project managers must consider department goals.

Department heads administer a collection of similar projects. Each department head is familiar with all these projects, and devises a strategy for tool and technology support. This support could include a reuse library, if such repositories are not organized at the division level, and education and training needs for the people in each project. Department heads might set programming and reuse standards, and usually assess adherence to the standards.

Division directors oversee a collection of departments that support a particular business niche. Division directors are often the ones who define reuse strategies as part of a broader business investment scheme. They see information technology in terms of profit and loss, and they value computers in the larger context of providing or supporting products and services. Conventional domain analysis for vertical reuse usually addresses the division or the department level.

Finally, *corporate executives* examine activities across and within divisions. They define a model for business domains, suggest criteria for reuse success, and create guidelines for reuse incentives and component ownership. Their goals and questions address not language and hardware, but the effectiveness and efficiency of techniques and tools in the context of business needs.

These roles can differ from one corporation to the next, and might not map easily to your company's structure. Regardless of the particular roles and names, however, every corporation must decide what level of staff asks and answers certain key questions. For example, who pays for reuse? When one project incurs expenses in the hope of recouping them on future projects, the tradeoff must reflect corporate choice and intent. And who owns the reusable components? Some level of management determines who owns them, and who is responsible for their maintenance. Who builds the components? Another level must be responsible for the construction, population, and support of reuse libraries, as well as the measurement data that describe their contents and related activities. Answers to these questions reflect the corporation's organization and priorities, just as profit-and-loss considerations reveal investment and leveraging strategies. You must support these questions and answers with measurement data, so you need to know who is asking and who needs the answer.

A consultant suggested these five perspectives to Amalgamated, with goals and questions reflecting each point of view. Many in the company recognized that these perspectives were distinct, and that omitting one would mean leaving out information about whether certain goals were satisfied. Nevertheless, Amalgamated chose to exclude programmers and department heads from their consideration of goals—a grave error. As Stan Rifkin and Charlie Cox noted in their analysis of successful measurement programs, the people doing the measuring must understand how it will help them.[5] This buy-in is crucial, not only for reflecting the needs of all participants, but also for giving participants a stake in the success of the reuse program. By ignoring the programmers, Amalgamated risked the loss of meaning in their reuse data: Programmers who are not convinced of the necessity of measurement or who feel as if they do not "own" the data will often supply incomplete or meaningless data so that they can get back to their "real" work!

Managerial lesson 2. The reuse measurement program must reflect the viewpoints of all participants in the corporation. When conflicts arise, they must be acknowledged and resolved, especially when they give mixed signals to the workers they affect.

Technical lesson 2. Reuse measurements must reflect the goals and perspectives that generated them, so that participants can understand and analyze the measurements, and get answers to their questions.

Deriving questions and metrics. Once we know the goals and players, the next step is to derive questions that reveal whether the goals are being met. For example, the goal of producing reusable components suggests several questions, such as:

- What are the target size and quality characteristics of a reusable component?
- How much longer does it take to make a component reusable than to make it for one-time use?
- How does the extra time to make a component reusable compare with the likely time saved for the subsequent uses of the component?

Then, we analyze each question to determine what must be measured so that we can answer the question. For instance, the third question could require several measures for understanding the answer, such as

- the time it takes to design, code, and document the component;
- the component domain(s);
- the probable number of uses; and
- likely maintenance costs.

Ultimately, managers must choose a metric based on what they can observe in the development or maintenance processes. As the box on page 122 describes, you can measure only what is visible. If you need to know something that is not yet visible, you must measure what you can while improving your process to enhance visibility later. Process visibility also affects the richness of what you measure, so that the more mature your process, the more information you can extract. For Amalgamated, the visibility issue raised questions about whether the company was ready for a reuse program—that is, whether they understood the development process enough to enhance it effectively with reuse activities.

Managerial lesson 3. Derive from goals the kinds of questions you need to answer, then the measurements that will help determine the answers. Your analysis can be generic (we will measure "size"), and the technical staff can determine the appropriate metrics (we will measure "size" as lines of code). Make sure you can interpret and present the measurements to each audience in a way that makes reuse success or failure visible.

Like anything else important to a company, the reuse program requires strategic planning.

Technical lesson 3. Determine what you *need* to measure before you decide what you *can* measure. Then fix any process problems to let you measure what you need. Be sure that you can present the measurement data to everyone who requires it, and that you can do so clearly and unambiguously.

How you slice it. To make them easier to handle, Amalgamated associated each measurement with three major activities of

PROCESS MATURITY AND MEASUREMENT SELECTION

Clem McGowan and I described how process visibility and maturity can affect your organization's or project's choice of measures.[1,2] Although not strictly derived from the Software Engineering Institute's Capability Maturity Model, our notion of increasing visibility allows you to decide what makes sense based on what you can see in a picture of the process.

To see how this works, consider an example. Suppose your organization is concerned about requirements volatility. The goal of understanding the reasons for and effects of a requirements change suggests that your development group measure both number of requirements and changes to those requirements. At the lowest level of visibility (akin to the CMM level 1), the requirements are ill-defined. Here, you measure the number of and changes to each requirement as best you can. At the next-higher level (similar to CMM level 2), the requirements are well defined, but the process activities are not. At this stage, you can measure the number of requirements by type, as well as changes to them. Now, not only do you know how many requirements are changing, but you can also tell whether the changes occur primarily in the interface requirements, the performance requirements, or the database requirements, or are distributed throughout the system specification. Actions taken based on these measurements have a more solid foundation and are more likely to be effective.

Similarly, at a higher level still (much like CMM level 3), the process activities are clearly differentiated; the project manager can tell when design is finished and coding starts, for example. Here, the number of and changes to requirements can be measured, but now you can also trace each requirement change to its corresponding design, code, or test component, enabling you to analyze the impact of the change on the rest of the system. The increased maturity of the process gives you more visibility, yielding a much richer body of information and a better understanding of the system than you had at lower levels.

REFERENCES

1. S.L. Pfleeger, "Lessons Learned in Building a Corporate Metrics Program," *IEEE Software*, May 1993, pp. 67-74.
2. S.L. Pfleeger and C.L. McGowan, "Software Metrics in a Process Maturity Framework," *J. of Systems and Software*, July 1990, pp. 255-261.

reuse: producing components, consuming components, and managing the library. Each question generated from high-level goals had a three-part identifier:[6]

- the activity to which it pertained: P for producer, C for consumer, or L for library;
- the perspective it reflected: PM for project manager, BM for business manager, or CR for the corporate view;
- its technical or managerial (T or M) bent.

The GQM analysis produced over 100 possible items to measure, organized by these perspectives. Figure 2 lists some of the questions raised and the metrics derived from them. But no organization can begin a measurement program with 100 measures! Instead, the consultant placed the measures in a large matrix, indexed by measure and question. Amalgamated managers could select the most pressing questions, then use the matrix to determine the associated measures. Or they could examine the measures engineers were currently collecting and use the matrix to determine which questions they could already answer in whole or in part. Either way, managers could use the framework of questions and metrics to define a small starter set of measures and expand it as needs dictated.

The process of deriving the questions and metrics raised many interesting issues about the nature of reuse at Amalgamated. For example, until the company conducted this analysis, no one knew who was responsible for a component once it was placed in a library. Similarly, the criteria for library acceptance were unclear. Amalgamated needed to decide when to deem something "reusable." And it had to devise a scheme for estimating how many times a candidate component was likely to be reused.

Amalgamated had hoped to impose fees for using the reuse library, and one goal of the measurement program was to help determine what those fees should be. Likewise, the program was supposed to suggest incentives for creating reusable components and placing them in the library. But many of the economic issues required study far beyond a simple metrics analysis. The library science community, where fees are usually *not* charged for borrowing a book, provided some lessons. Many researchers have studied the financial implications of reuse, emphasizing how this investment must be considered in the context of other corporate business decisions.[7-10]

Managerial lesson 4. Measurement is an integral part of managing software development and maintenance, as well as software reuse. You should make reuse decisions only after carefully exploring the lessons from related fields, such as economics and library science.

Technical lesson 4. Measurement is essential to understanding what is reusable, how best to reuse, and what to put into and take out of the library.

Modeling the process. Business goals and questions suggest why you should measure the process, and process maturity suggests both what you can measure and where in the process you can capture it. But a metrics plan requires more than a simple list of what to measure.[11] In particular, we must find out

- When in the process do we measure, and how often?
- Who does the measuring?
- How is the measuring done? Is it automated or manual?

A process model providing a context for measurement lets us answer these questions. We can represent this model in a variety of ways, but it must include major activities, resources, sequences of events, input and output for each activity, and constraints such as budget and schedule.

Questions for the producers of reusable components

What are the target size and quality characteristics of a reusable component? (P-PM-T1)
- Size characteristics
- Quality characteristics
- Component type
- Certification level
- Component dependencies

Are there financial or other incentives for creating reusable components? (P-PM-M1)
- Incentive paid for building them
- Percentage of system required to be reusable

What training is required so that software developers and project managers understand reuse and related tools? (P-BM-T1)
- Amount of effort needed to train
- Amount of time needed to train

How will reusable components require different care from nonreusable ones? (P-BM-M2)
- Effort to prepare component for library
- Time to prepare component for library
- Amount of effort needed for training
- Amount of time needed for training

What percentage of the components being produced does the producer consider reusable? (P-CR-T1)
- Portion that is reusable
- Portion that is not reusable

How can the process be improved? (P-CR-M4)
- Size characteristics
- Quality characteristics
- Component dependencies
- Incentive paid for building a reusable component
- Number of uses of component
- Tag to denote evaluated/included/excluded
- Number of examinations of component
- Whether component is product of domain analysis
- Number of extractions of component

Questions for the end users of reusable components

How does the evolution of reused components affect system design? (C-PM-T9)
- Size characteristics
- Quality characteristics
- Component dependencies

How much effort is required to modify a reused component? (C-PM-M5)
- Size characteristics
- Quality characteristics
- Actual effort to reuse
- Estimated effort to build new
- Component dependencies
- Portion of component modified for reuse
- Effort to integrate with existing system

Other questions about reuse

How is the domain changing over time? (C-BM-M2)
Does being a product of domain analysis increase the likelihood of component reuse? (C-CR-T3)
How effective is the library system? (L-PM-T3)
How much effort is involved in certifying, maintaining, and cross-referencing components? (L-PM-M1)
What is the usage profile of components, and how can it be improved? (L-BM-T1)
What fees should be charged for using the library? (L-BM-M4)
What is the most efficient retrieval technique? (L-CR-T1)
What trends of component characteristics promote reuse across domains? (L-CR-M6)

Figure 2. *Sample questions to consider before implementing reuse. The three-part identifier is in parentheses, and the derived metrics are listed under each question.*

The Amalgamated process model included four key activities: domain evolution (DE), component production (P), library management (LM), and component use (C) (see Figure 3). Domain evolution involves identifying domains likely to provide reusable components. The model emphasized *evolution* rather than *analysis*, because the domains ripe for reuse can change with the company's business focus and priorities.

This process model raised many questions. Notice, for instance, that the process model is completely separate from day-to-day development and maintenance. For this model to show a programmer how reuse should work, it must integrate regular development activities. When are programmers supposed to fit in reuse? Especially in an organization with a standard process, the reuse managers must demonstrate how reuse relates to development. Without such integration, reuse will likely never become a natural part of development, and reuse activities will stop when management withdraws incentives or changes priorities.

Nevertheless, the reuse process model is useful for understanding where to make measurements. For each metric identified and derived from goals and questions, an analyst can

- find the place in the process where the measurement should be made,

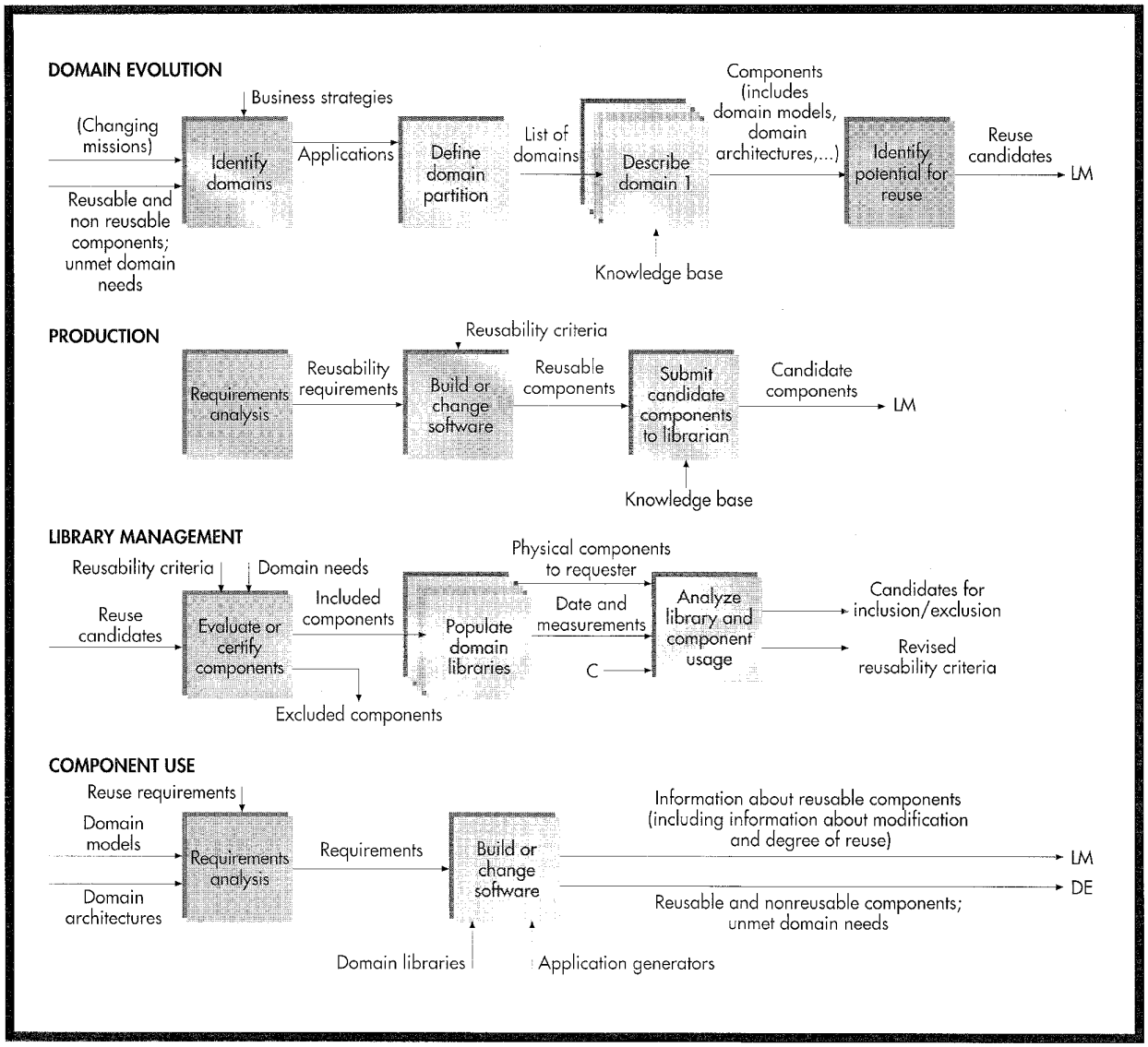


Figure 3. *Reuse process model: domain evolution, component production, library management, and component use.*

- record who should make it,
- record when and how often to measure, and
- record where to store the result and who can access it.

The last bullet implies building a metrics database. Figure 4 shows how Amalgamated mapped measures to the library management process. The consultant assigned each metric a number, and annotated the process model to show where to make the measurement. However, the model did not specify the who, when, and how.

As before, associating metrics with the reuse process led management to revise the process, questions, and measurements. The mapping demanded clarifying the source of component quality information and redefining the nature of the quality measures, for example. Moreover, the mapping led to questions about responsibility: Who fixes a broken component? Can there be more than one version of a component? Once again, modeling and measurement worked hand in hand to improve the overall process.

Managerial lesson 5. Measurement requires an understanding of the process as well as of the business needs.

Technical lesson 5. You must fully integrate reuse and reuse measurement into daily development activities.

CONTINUOUS PROCESS IMPROVEMENT

Amalgamated's reuse metrics plan is not yet complete; measurement activities do not end with data capture. The plan must describe how the data will be used and presented. That is, we must tie the metrics back to the goals and questions, so that we understand who will see the data and what decisions that data will support. For example, it is not enough to collect data on component quality. The plan should describe how to track quality (daily? monthly? by component? by project? by component type?) and who will use the results (librarians? producers? programmers? managers?).

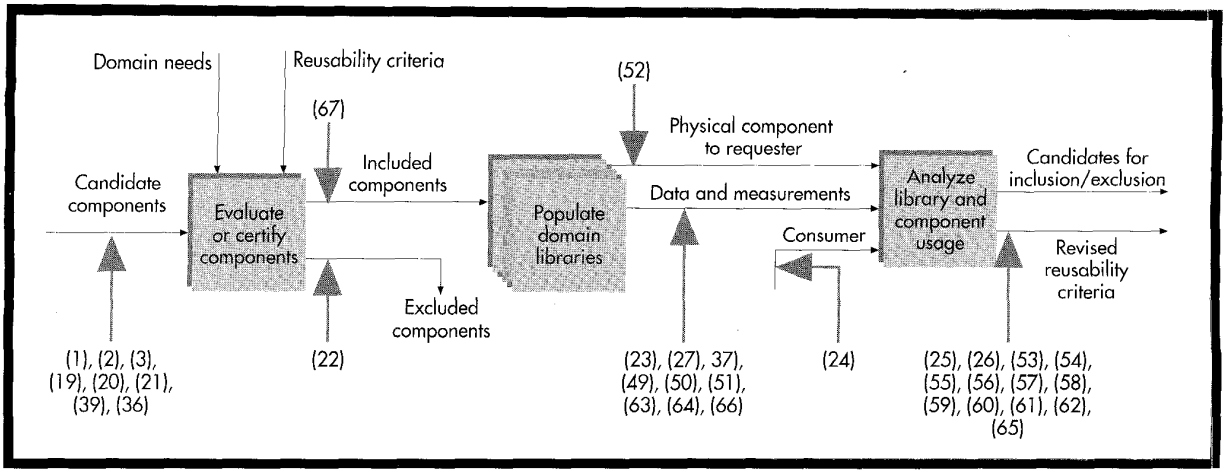


Figure 4. *Partial process mapping: library management. The numbers name the metric to be measured at each step indicated by a shaded arrow.*

Periodic evaluation. To stay successful, businesses should be flexible and grow. So too should metrics and reuse programs. Like anything else important to a company, the reuse program requires strategic planning. In particular, management should revisit the reuse measurement plan periodically to decide if changes are in order, asking a number of questions.

♦ *Are the goals the same?* If business goals change, the reuse goals might need to change with them. Initial goals might involve establishing a manual or automated component repository and making it available to developers. Once that is accomplished, goals relating to increased productivity could supersede availability goals. Similarly, once productivity rises, the company or division might want to focus on quality.

♦ *Are the priorities of the goals the same?* As the company implements one type of improvement successfully, it might select other types for the next initiative. For example, goals involving code reuse might initially receive high priority. However, once code reuse becomes part of the corporate development culture, the program might focus on requirements and design reuse.

♦ *Are the questions the same?* Questions relevant for the first stages of a reuse program (such as how much we should invest in reuse technology) might be replaced by questions reflecting the maturation of reuse (such as how much money and time we are saving by reusing code). Similarly, reuse projects that start with pilot projects might eventually generate new questions related to technology transfer (such as what costs from the pilot project will we incur again when we introduce reuse company-wide).

♦ *Are the metrics the same?* As the process matures, the richness of measurements increases. At the same time, some metrics might become obsolete, replaced by new ones. For instance, we might replace the initial collection of size measures with some kind of functionality measure. Similarly, as reuse becomes widely accepted in the corporation, we can start collecting measures of reliability, availability, and maintainability to determine the effect of reuse on corporate product quality. Metrics related to library quality and customer satisfaction might also grow, from measuring the number of components extracted and used to measuring interface and customer service quality.

♦ *Is the maturity the same?* The maturity of the reuse process could improve, and with it the visibility needed to measure new items. For example, initial attempts at reuse might not be automated. The catalog of available, reusable components could consist simply of a deck of index cards. But as reuse grows and with it the body of components, the company might need to develop an automated system to support it. An automated system allows us more easily to measure how often a component appears in a match of search criteria, how often it is examined for possible use in a new product, and how often it is actually extracted and used. These new measurements help library managers to know which components they should remove from the library (because no one searches for them) and which they should modify to make more suitable for reuse (because people search for them, but rarely use them in their current form).

♦ *Is the process the same?* The reuse process itself can change dramatically over time. Domain analysis can grow or disappear, depending on the corporation's business plans. Library management can change as librarians automate or connect repositories throughout the corporation. And production or usage can change as reuse grows to incorporate specifications, designs, test suites, documentation, and more. Each change has important implications for measurement, since we might want to quantify the type and magnitude of the change, as well as its impact on reusable components and the surrounding technological structure.

♦ *Is the audience the same?* Many reuse programs start small, often as a pilot in a department or division, and expand to the corporation slowly and carefully. As they do, the audience for measurement changes from programmers, managers, and department heads to division heads and corporate executives. That is, the audience changes with the impact of the reuse. The measurements must change, too, to reflect the audience's questions and interests.

Other issues. The above considerations help us decide if our measurements correctly reflect the concerns of those involved in reuse. They also help ensure that we have the numbers we need to support our decisions. But Amalgamated must

address other issues—issues that measurement cannot solve but which profoundly affect the success of the reuse program.

♦ *Do we have the right model of reuse?* Amalgamated gave a great deal of thought to parts of the reuse process. It decided to develop a front-end user interface for its several existing reuse libraries and planned to implement a fee schedule for borrowing components from a library. But Amalgamated never considered the reuse process as a whole, so it never understood the potential impact of reuse on the corporate culture.

The criterion for success should have been increasing usage of components, rather than number of components.

♦ *How does reuse fit into development and maintenance processes?* Unless reuse is integrated with standard practice, it will never become widespread, because engineers won't know when to do it. Amalgamated should have augmented its standard development process with reuse activities. Instead, the corporation continued to support separate task forces for each technology innovation it was pursuing, without mapping the technology to the day-to-day activities that the developers perform.

♦ *Do we have the right criteria for success?* In spite of his stated goals, Amalgamated's vice president ultimately chose as his key measure for reuse success the number of components in the libraries. As a result, developers were encouraged to shovel as many components into existing reuse libraries as they could. The vice president then told the board that the reuse program was successful, because the libraries held thousands of components; the potential savings were astounding. However, there were relatively few uses of those components, so the actual savings were very small indeed. Clearly, the criterion for success should have been increasing *usage* of components, rather than number of components.

♦ *How can we adjust current cost models to examine collections of projects, not just single projects?* Many decisions about potential cost savings require cost estimation techniques that involve several projects. But most commercial cost models focus on single projects, not on the collections of projects we need to justify reuse investment. New cost models should integrate aspects of multiple projects to support reuse decisions.

♦ *How do regular notions of accounting (such as return on investment) fit with reuse?* Those high in corporate management view reuse investment in the same terms as any other business investment. They want to discuss reuse options, indicators, and return on investment, not number of components, languages, or search techniques. Software engineers must translate the language of reuse into the language of accounting, so that financial managers can compare reuse investment with other possible corporate investment initiatives.

♦ *Who is responsible for component quality?* When the public library purchases a book, the library is usually responsible for maintaining the book: protecting it with a dustcover, checking for damage upon return, and replacing or repairing it when it wears out. But Amalgamated had different ideas for its reuse libraries. It decided to make a component's author continue to be responsible for quality after placing the component in a library. Such a scheme might work, but only if it fits in with the corporate culture. What happens when an author leaves the company? What happens when the component evolves or spawns multiple versions? What are the criteria for component quality, both for initial acceptance in the library and for maintaining the component as it changes over time?

♦ *Who is responsible for process quality and maintenance?* Reuse entails more than setting up libraries and filling them with components. Someone has to monitor the level of reuse, making sure that good components are identified, labeled properly, and used often. Someone has to cull the unused components and analyze them to determine why they are not used. Someone must evaluate the domain analysis process to determine if the targeted components are in fact highly reusable. And someone has to study the effect of reuse on the corporate bottom line to determine return on investment and future reuse policy. These ongoing activities occur at various levels of the corporation, and are necessary parts of a reuse program.

Final lessons. These observations lead to the final set of lessons learned.

Managerial lesson 6. Measurement makes visible the characteristics of process, product, and resource that management needs to support key business decisions. But measurement will not answer tough questions about business goals, process structure, and corporate direction. You must answer these questions before putting a reuse program in place.

Technical lesson 6. Reuse is not entirely a technical issue. The success of reuse depends as much on good business decisions as on good components in the library.

The Amalgamated reuse program still exists. Its project managers sometimes use high-level cost models

to make extravagant claims about costs the company avoided by reusing code. However, actual levels of code reuse do not seem to justify the claims, in part because Amalgamated's measurement set is not complete enough to address the questions that its management asks. When the vice president requested a reuse metrics plan, he expected a half-page list of several items to measure and report. But "what should we measure for our reuse program?" is easy to ask and difficult to answer.

Understanding measurement for reuse depends on understanding organizational goals, project goals, and individual goals. It also depends on how the reuse process fits in with development and maintenance processes. Many reuse efforts fail not because of technological insufficiency, but because of analytical insufficiency: The organization must think and plan before acting. So the bottom line is this: You cannot do useful measurement without understanding the process issues, and you cannot do effective reuse without proper measurement and planning. ◆

Shari Lawrence Pfleeger is president of Systems/Software, where she consults with industry, academia, and government on software engineering and technology evaluation. In the past, Pfleeger has been principal scientist at the Contel Technology Center and at MITRE Corporation's Software Engineering Center, and a visiting professorial research fellow at City University of London's Centre for Software Reliability.

Pfleeger received a PhD in information technology and engineering from George Mason University. She is a member of IEEE, IEEE Computer Society, and ACM. She is associate editor-in-chief of *IEEE Software* and an advisor to *IEEE Spectrum*.

Address questions about this article to Pfleeger at Systems/Software, 4519 Davenport St. NW, Washington, DC 20016-4415; s.pfleeger@ieee.org.

REFERENCES

1. R. Prieto-D'az, "Status Report: Software Reusability," *IEEE Software*, May 1993, pp. 61-66.
2. W. Frakes and S. Isoda, guest eds., *IEEE Software* special issue on reuse, Sept. 1994.
3. M. Griss and M. Wasser, "Making Reuse Work at Hewlett-Packard," *IEEE Software*, Jan. 1995, pp. 105-107.
4. V.R. Basili and D.R. Weiss, "A Method for Collecting Valid Software Engineering Data," *IEEE Trans. Software Eng.*, Nov. 1984, pp. 728-738.
5. S. Rifkin and C. Cox, "Measurement in Practice," Tech. Report CMU/SEI-91-TR-16, Software Eng. Inst., Pittsburgh, 1991.
6. M.F. Theofanos and S.L. Pfleeger, "CIM Reuse Metrics Plan, Version 2.3," Martin Marietta Energy Systems Tech. Report K/DSRD-1231, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1992.
7. B. Barnes and T. Bollinger, "Making Reuse Cost Effective," *IEEE Software*, Jan. 1991, pp. 13-24.
8. T.B. Bollinger and S.L. Pfleeger, "Economics of Reuse: Issues and Alternatives," *Information and Software Technology*, Sept. 1990, pp. 643-651.
9. S.L. Pfleeger and T.B. Bollinger, "The Economics of Reuse: New Approaches to Modelling and Assessing Cost," *Information and Software Technology*, Aug. 1994, pp. 475-484.
10. W.C. Lim, "Effects of Reuse on Quality, Productivity, and Economics," *IEEE Software*, Sept. 1994, pp. 23-30.
11. S.L. Pfleeger, "Maturity, Models and Goals: How to Build a Metrics Plan," *J. Systems and Software*, Nov. 1995, pp. 143-155.