Closing the Loop on Improvement: Packaging Experience in the Software Engineering Laboratory

Sharon R. Waligora, Linda C. Landis, Jerry T. Doland

Computer Sciences Corporation
10110 Aerospace Road
Lanham-Seabrook, Maryland 20706

Abstract

As part of its award-winning software process improvement program, the Software Engineering Laboratory (SEL) has developed an effective method for packaging organizational best practices based on real project experience into useful handbooks and training courses. This paper shares the SEL's experience over the past 12 years creating and updating software process handbooks and training courses. It provides cost models and guidelines for successful experience packaging derived from SEL experience.

1. Introduction

The Software Engineering Laboratory (SEL) is a partnership among NASA Goddard Space Flight Center (GSFC), Computer Sciences Corporation (CSC), and the University of Maryland; it has received international recognition for its achievement in continuous, measurable improvement in software products and processes. The SEL supports the Flight Dynamics Division (FDD) at GSFC, which builds software systems for satellite ground support and spacecraft attitude control.

The SEL has forged a process improvement approach that identifies the goals of the organization, initiates process improvement initiatives based on those goals, and measures the impact of those initiatives on the products produced. This approach is based on the concept of organizational learning from project experience, similar to the way that successful people learn from their experience and apply new techniques to the way they do their jobs. For example, once an improved process or new technology has been used successfully by a pilot project, it must be shared with other projects to broaden its impact. This expansion of process improvements throughout the organization is accomplished in the SEL through packaging. Packaging is a structured mechanism for capturing the best practices, the most effective technologies, and the lessons of past experience and communicating that information throughout the organization. By making improvements part of the standard way of doing business, packaging closes the process improvement loop.

Recent SEL experience shows the benefits of packaging. In the late 1980s, the SEL began experimenting with several software engineering technologies, including object-oriented design, the Ada language, and the Cleanroom methodology. Around 1990, the SEL updated and improved its methodology guidebooks and developed new training courses. As a part of this update, beneficial parts of each of the experimental technologies were integrated into the methodology along with process improvements derived from best practices that had evolved since the guidebooks were last revised. Key product measurements from the 1990–1993 time period show dramatic across-the-board improvements over baseline measurements taken in the mid- to late 1980s: a three-fold increase in software reuse that resulted in significant cost and schedule savings, and a 75 percent decrease in software development

errors. These improvements can be traced to new techniques—such as the high-reuse process that grew out of the Ada/OOD experimentation and the consistent use of software inspections and code reviews that was introduced by the Cleanroom methodology—which were highlighted and stressed in the new guidebooks and training.

These standards are not just "shelfware;" a survey of the local software engineering staff indicated that users find the guidebooks relevant and easy to use. The survey results revealed that 95% of the software developers use the guidebooks, with software project leaders and managers using them most frequently. In addition, SEL guidebooks have been cited by industry publications, such as *The Software Practitioner*, as excellent examples of practical software engineering standards. The SEL's *Manager's Handbook* has been used as a textbook for software management courses at the University of Maryland, the Johns Hopkins University, and McGill University in Canada. The training courses also have been well received. Course evaluations consistently rate the SEL courses as highly relevant and informative, with 90% of the participants stating that the courses were well worth the time they had invested.

This paper describes the SEL packaging process—our approach to capturing and reusing experience. We discuss the methods used to synthesize experience into a standard software engineering process and to effectively communicate that process to the software engineers. We consider the needs of the audience; sources of information; issues of package scope, content, and format; and offer cost and schedule models for packaging. Finally, we summarize some of the key lessons learned and rules of thumb for packaging experience.

2. Background

2.1 SEL Process Improvement Paradigm

The SEL's process improvement paradigm is shown in Figure 1. The first and most important step is *understanding* how an organization currently does business and what it values. This is done by characterizing the products generated and the process that is used to produce them. In the second step, *assessing*, the organization sets goals for improvement, and experiments with process changes, such as a new technology, that might help achieve its goals. This is done by introducing a process change on pilot projects, assessing its impact on the product, and refining it if necessary before selecting it for use throughout the organization. The final step is *packaging*, where the successful new technologies and procedures are integrated into the organization's standards and training program so that all projects may benefit from the changes.

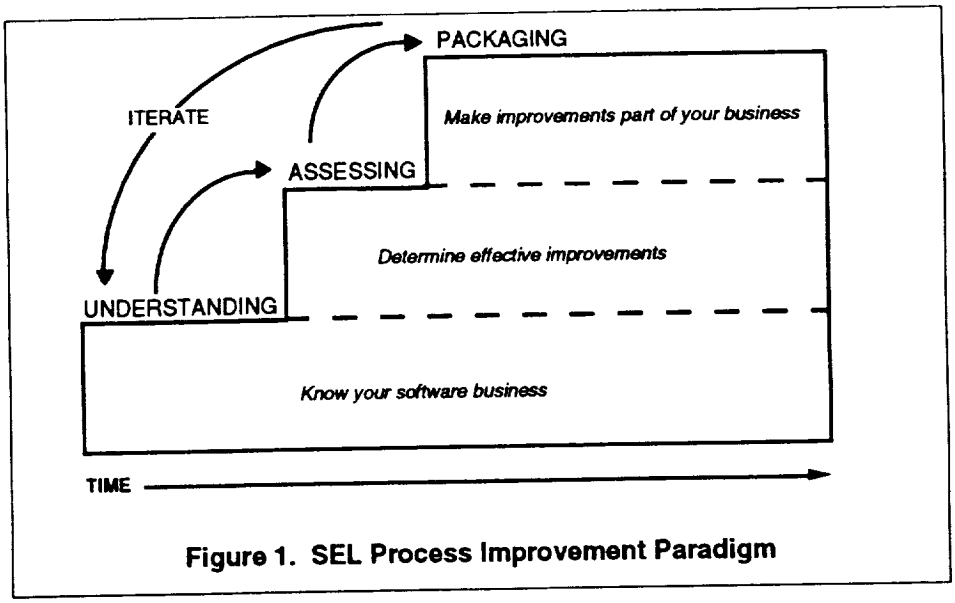


Figure 1. SEL Process Improvement Paradigm

Within the SEL, a group of researchers, analysts, and support personnel (separate from software developers) perform process improvement activities. They collect and analyze software project measurements to produce models and standards for use by the projects. They design and monitor experiments with new technologies and modified procedures to determine their applicability to the local environment and refine/tailor them for optimum use in the FDD. They package research results and local experience in process guidebooks, training courses, and tools.

2.2 Experience Packaging

The SEL relies on its measurement program to provide a view into actual product and process characteristics. Similarly, it uses experiments to gain additional insight into the effect of new or modified techniques, tools, and processes on the products. Based on this information, the SEL identifies and captures the most appropriate practices/technologies in "experience packages." These packages are in the form of standards, tools, and training that give practical guidance on how to apply the new techniques in the context of the local process. This guidance effectively captures the results of the understanding and assessment phases, packages them for "reuse" by subsequent projects, and integrates them into the routine software business. SEL packages are always designed with the local organization's needs in mind, but many have found broader applicability outside the SEL domain.

Packaging is performed by a team that is independent from the development organization, but whose members work closely with development personnel. Packagers talk with developers to learn about improvements made within the projects while using the standard process in a changing environment. They study project documentation and data to verify their findings. Using the current documented software development process as a reference point, packagers determine the evolved state of the practice based on current project experiences and create new baseline models. The packagers also integrate beneficial new methods derived from SEL experiments into the standard software development process. Working in consult with the project personnel who will use the materials, the packagers synthesize all of this information into an updated process. From there, they design the optimal presentation of the information, develop the package, and introduce it to the users through organized deployment and delivery.

2.3 SEL Experience Packages

The SEL has developed a primary set of guidebooks, training, and tools that document and support the evolving local process. In addition to these major packages, the SEL produces technology reports and interim packages. These products support the assessing step of the process improvement paradigm and have shorter life spans and are less extensively distributed. Technology reports record the results of SEL studies with specific technologies and techniques. They contain the recommendations and rationale for including all or parts of the subject technology in the standard local process or for abandoning the technology or process change as inappropriate for this environment. Interim packages fill the gap when a new technology is being assessed, while its applicability in the environment has not been determined or sufficiently refined for widespread local use. Some interim packages have been incorporated into later updates of the standard methodology, and others have been entirely superseded.

2.3.1 Guidebooks

The SEL has produced a set of guidebooks that defines the baseline development standard. The guidebooks communicate the rationale for the methods and offer guidance for applying them, rather than specifying detailed procedures. We have found that this level of detail allows each project the flexibility to define project-specific procedures as needed (based on those used by previous similar projects) to meet the needs of its current environment. Given that detailed procedures change as improvements are introduced and the organization evolves, segregating procedures from the formal documentation mitigates the need for project waivers and continual updates to the standards.

In addition to the baseline standards, several specialized documents have been developed to support tailored applications of the local process,

such as implementing in a particular programming language or using a tailored methodology. We have learned that it is best to document language-specific processes separately from the baseline methodology; this allows them to be modified as frequently as needed to keep pace with rapidly changing technology.

Baseline Standards

- *Manager's Handbook for Software Development*—Contains the models, guidelines, and acceptable processes for managing the development of flight dynamics systems. It provides specific guidance for using planning and performance models to successfully manage software engineering projects.
- *Recommended Approach to Software Development*—Presents guidelines and standards for developing software in the flight dynamics environment. Intended for developers and technical managers of software development projects, it describes the recommended practices for each phase of a software development life cycle, including key activities, products, measures, methods, and tools.
- *Cost and Schedule Estimation Study Report*—Presents planning models for cost and schedule estimation based on local project data. The planning parameters are built into spreadsheet tools for use by project managers and are updated yearly based on ongoing analysis.

Tailored Standards

- *Ada Developers' Supplement to the Recommended Approach*—Presents guidelines for programmers and managers who are developing flight dynamics software in Ada. Intended to be used in conjunction with the *Recommended Approach to Software Development*, it provides additional detail on reuse and object-oriented analysis and design.
- *C Style Guide*—Presents the recommended practices and coding style for programmers using the C language in the flight dynamics environment. The guidelines are based on generally recommended software engineering techniques, industry resources, and local convention. It offers preferred solutions to C programming issues and illustrates through examples of C code.
- *Cleanroom Process Handbook*—Presents guidelines for using the Cleanroom methodology in the flight dynamics environment. It describes the Cleanroom life-cycle model and the specific activities performed in each life-cycle phase. It also addresses pertinent managerial issues and highlights the key differences and similarities of the SEL Cleanroom process and the standard development approach. This handbook started out as an interim package and later became a tailored standard after the methodology matured in the local environment.

Figure 2 shows the development history for several SEL guidebooks. The *Manager's Handbook* and *Recommended Approach* were initially developed in the early to mid-1980s and then updated around 1990. The SEL developed a few interim guidebooks, a *Generalized Object-Oriented Development (GOOD) Guide* and an

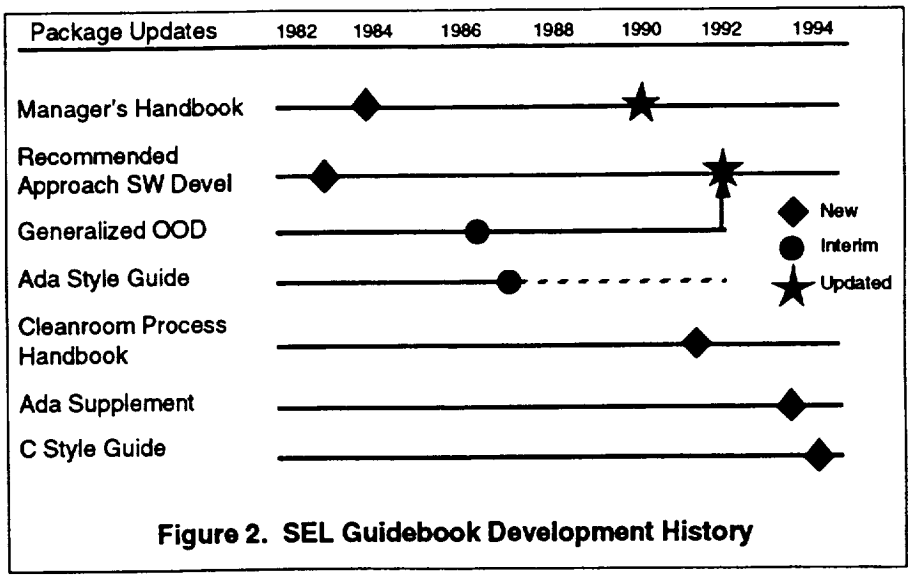


Figure 2. SEL Guidebook Development History

Ada Style Guide to support experiments in the late 1980s. The *GOOD Guide* eventually was absorbed into the next update of the *Recommended Approach* while the *Ada Style Guide* was replaced by an evolved industry standard.

2.3.2 Training Courses

SEL training packages include several core courses and a training plan that documents the goals of the training program, describes course content, and recommends the training sequence for project personnel. The core courses cover the SEL organization, methodology, and process improvement approach, and provide in-depth training in the application area and software development process. These courses are updated as needed to reflect changing process elements within the SEL. All staff (managers, developers, maintainers, testers) are expected to participate in the core set of training classes. Courses include:

- *Orientation to the FDD*—Orients the newcomer to the local environment, application/mission, organization, process improvement approach, and methodology; 6 hours–lecture.
- *Principles of Flight Dynamics*—Bridges the gap between academic mathematics and physics, and their application in flight dynamics software systems; 30 hours–lectures and homework exercises.
- *Recommended Approach to Software Development*—Illustrates the use of and rationale for applying the local software development methodology (based on the guidebook discussed above); 24 hours–lectures and workshops.
- *Task Leader/ATR Training*—Demonstrates how client and contractor project leaders work together within the context of the contract to successfully manage software projects; 12 hours–lectures, workshops, and interactive exercises.

2.3.3 Tools

An important aspect of packaging is the infusion of technology in the form of support tools for use by project personnel. The SEL developed a project management tool called the Software Management Environment (SME) that puts local experience at the fingertips of project managers. The SME provides access to the SEL's database of previous project information and baseline process models. Using the SME, a manager can, for example, compare the growth rate of source programs or the error rate of the current project against the models, or, using data from similar projects in the database, the manager can predict future trends on the current project. This tool has helped institutionalize the SEL process, because project managers can use it to gain insight into their software projects.

3. Packaging Guidebooks and Training

The SEL's current packaging process is based on the fundamental understanding that the local software engineers are the primary users of our products. When developing guidebooks and training courses, the SEL emphasizes user involvement to ensure that the documented process matches what is actually done, that recommendations are based on agreed-upon procedures, and that the end product will be useful to the software engineers.

This approach has evolved over the years, with the SEL learning from some missteps along the way. For example, the first issue of the SEL's baseline standard, the *Recommended Approach to Software Development,* was a classic case of the "typical" approach (described below). Originally conceived and published without much input from local, practicing software engineers, the standard was ill-received and almost immediately recalled for revision. At that point, early SEL "packagers" decided to take a new approach, and just write down exactly how the developers actually produce, test, and maintain software in this environment. Their new document, albeit rough, formed the basis of the current *Recommended Approach.* This guidebook has since been refined based on the experience packaging concepts discussed in this paper. At its core is the concept that the users know best how they do their jobs and what guidance they need to support them in their work.

3.1 Typical Industry Approach

The typical industry approach to defining process often results in the creation of "shelfware," i.e., standards and procedures that aren't used and wind up gathering dust on bookshelves throughout an organization. This commonly used process (shown in Figure 3) begins with managers or quality assurance personnel creating a list of topics to be addressed based on an external (industry) standard. The topics are then divided up and distributed to people who have expertise in the subject areas. These people do their best to draft the standards and procedures for their particular topic in their spare time—because rarely are there resources or time allocated to the effort—while also meeting their regular project responsibilities. The draft standards are subsequently distributed to a small group of reviewers and turned into "legalese" by incorporating everyone's review comments. They are then assembled by a coordinator, published, and distributed to the developers. When the standards are delivered, it may well be the first time that most of the developers will have seen them. They peruse them, often don't understand them or recognize their relevance, and so, they place them on the shelf, and continue following their current process.

3.2 SEL Approach

The SEL packaging process, illustrated in Figure 4, involves the users directly. Two separate groups each play an important role in this process: the *packagers* who document the process and the *software developers* who are the users of the process and the supporting packages. The packagers, who are usually dedicated full time to the effort, are responsible for gathering and distilling process information and then presenting it in a useful form. The experienced developers are one of their key sources for this information. SEL packagers also consider information from the SEL's metrics database and results from SEL experiments when defining the updated process. This information flow is depicted by the outer loop in Figure 4.

The inside loop in Figure 4 represents the iterative method used to refine the software development process and develop the package. Often, as the preliminary step in a packaging effort, project personnel are interviewed or invited to a brainstorming session to gather information and requests for package content. They explain to the packagers how the process is being applied in the current environment; they raise issues and problem areas; and they offer suggested improvements. Based on the identified strengths

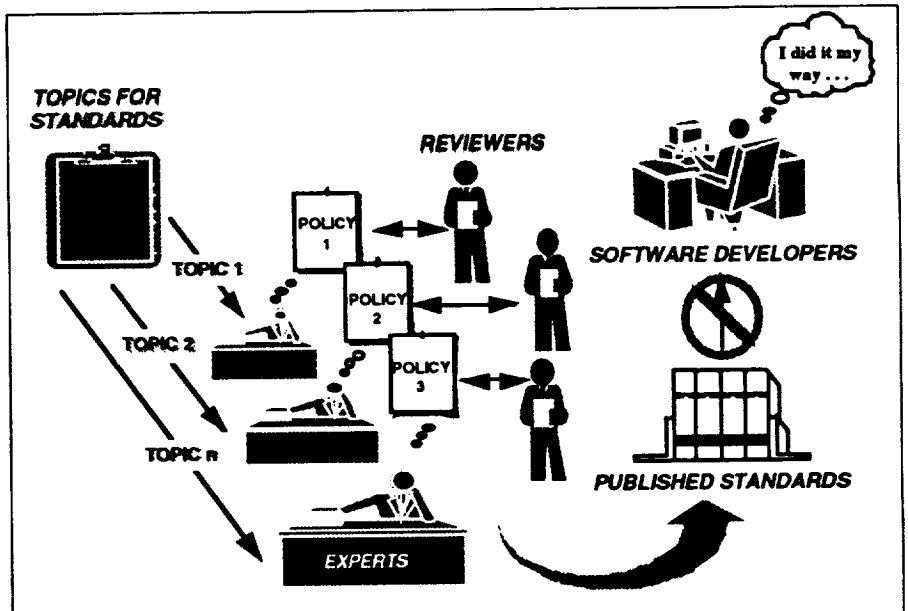


Figure 3. Typical "Shelfware" Development Process

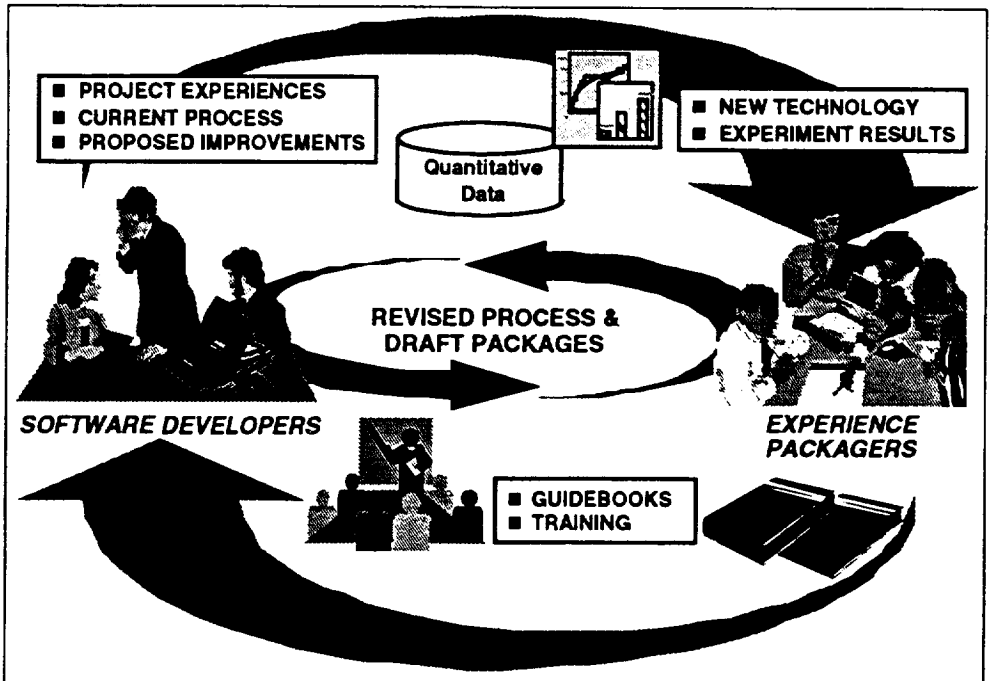


Figure 4. SEL Package Development Process

and weaknesses of current books and courses (if available), packagers design and prototype new packages using key developers as reviewers for both content and usability.

We find that this approach results in accurate, usable guidebooks and effective training courses. The developers feel connected to the products because they were involved in creating them, and they appreciate the fact that they were not burdened with producing them. The quality of the package is top-notch because the team that produced it was dedicated to the effort and skilled in communication, information analysis, and desktop publishing.

3.3 Packaging Activities

The basic activities in the experience packaging process include:

- Information gathering and synthesis
- Package development
- Package deployment

These activities are looked at in detail in the paragraphs that follow.

Information Gathering and Synthesis

Packagers first review any existing version of the standard or guidebook that is to be updated or, in the case of training, the information on which the course will be based. In essence, they apply step 1 of the process improvement paradigm, understanding; they baseline the documentation that currently exists in the environment. Packagers then apply step 2 of the paradigm when they interview experienced developers, maintainers, testers, and managers to assess how closely the actual software engineering practice maps to the documented process. They determine where the process has changed and how it has been tailored for different situations, and they validate this information by analyzing empirical data. Packagers also review SEL study reports and experiment results to identify new techniques that have been recommended for inclusion in the standard process. Finally, all of the input is synthesized to define the new process. Facilitated workshops are then organized to clarify issues and to develop consensus on the process content.

Package Development

Once the content is decided, the focus shifts to designing the right package to communicate the information. Users are interviewed to understand their work habits and approaches to learning. They are asked to cite the strengths and weaknesses of existing courses and guidebooks. This helps packagers choose the most effective communication style based on the users' preferences and to choose the most appropriate course format for their audience and subject matter. Although guidebooks and training courses have the same goal of describing a process and the rationale for applying it, they are fundamentally different communication media. Whereas guidebooks provide standalone text references, successful training courses leverage the combination of written (visual) material and a human instructor in an interactive setting. These two types of packages require different production processes.

Guidebooks

For guidebooks, the packagers begin by developing prototypes of key sections to get early feedback from the users on the "look and feel" of the document. This is an extremely effective way to validate and further refine the packagers' understanding of the right level of detail and to get feedback on different communication styles. The text and layout of the guidebook are then developed iteratively, allowing the users to review both content and format. As the document evolves, key developers serve as expert reviewers. The packagers also solicit comments from a broader group of reviewers in the final stages of development, before deploying the final product. The SEL has found that a typical guidebook may go through 3–4 iterations before it is ready for final review. Throughout the review and feedback process, SEL packagers carefully scrutinize and synthesize all review comments to avoid inserting into the guidebooks "special interest" language (e.g., individual preferences or compliance "loopholes") and to guard against writing in the obtuse style that often results from mass review.

The SEL has discovered that this iterative process results in user-friendly guidebooks that are actually used. SEL guidebooks use graphics to illustrate concepts and lead users to the information they need. For example, the *Manager's Handbook* uses an innovative graphic layout to communicate measurement models, and the *Recommended Approach* includes keywords, notecards, icons, "chapter highlights," and a detailed subject index. Both are designed as references, rather than for one-time reading.

Training Courses

When developing training, the first step is setting goals for the course and identifying the primary audience. This is typically done in a brainstorming session with personnel from the development organization, where packagers gather information about user needs. Packagers then meet with the selected instructor(s) to define the course outline and to choose the best organization and apportioning of the information into individual class sessions. Packagers work with the instructor to determine the appropriate level of interaction for the various classes based on the material to be covered. We have found workshops in which participants can practice new techniques to be essential when teaching new skills. Classroom brainstorming and role playing exercises help students discover new ways of thinking about familiar subjects, and lectures are most useful for conveying new information.

Course developers create a preliminary set of training materials, including lecture slides, project examples and handouts, and workshop exercises. The course is then reviewed for content and continuity. When the content is stable, the slides are livened up with graphics and polished to become a cohesive package that will hold the participants' interest. At this point, a dry-run of the course is held as a final review. This allows the instructor to get comfortable with the material and provides a forum for soliciting final review comments before deployment.

Package Deployment

Deployment is a critical step in the packaging process. If guidebooks are simply dropped on people's desks or training courses are simply announced, the packaging effort is likely to fail. Software developers must read the books and attend the courses for the information transfer to take place. The SEL has found that a publicity campaign is important. The people need to

know that a new guidebook or course is coming, that it is new and different, that it will be useful (we show examples), and that their colleagues (we name them) contributed to it. Those who were involved in the package review already have "bought in," so their marketing help is solicited. Managers are invited to attend dry runs of the training courses and to review guidebooks. This is an excellent way of getting their input and support; managers are in the best position to encourage their people to attend training and to use the guidebooks. Briefings at all-hands meetings and posters also work well to call attention to a new package.

The SEL has found that guidebooks that are closely followed by a related training course are particularly effective for infusing process change. The training course serves to get users "into" the guidebook, demonstrating for them how it can support their work. Training also provides a forum where revised elements of the process can be pointed out and clarified. Accompanying workshops provide a safe environment for getting hands-on experience with new techniques.

3.4 Investment in Packaging

In the SEL, we spend about 10% of the software budget on process improvement activities, of which a relatively small percentage is spent on the packaging process described here. Over the past 5 years, the SEL has spent 1.5% of the total software budget on packaging: 1% on guidebooks and 0.5% on developing training courses.

During that time, we have tracked costs and schedules for most of the packages that we have produced. Our data show that it costs about 24 staff-hours per page to develop guidebooks and 55 staff hours per hour of class time to develop training courses. The effort expended and the relative size of the guidebooks and training courses are shown in Tables 1 and 2, respectively. Please note that these numbers reflect the effort spent by the packagers only; none of the developers' time (which is relatively small) is included here.

Table 1. SEL Guidebook Cost and Schedule Data

Guidebook	Pages	Effort (staff-months)	Schedule (months)
Manager's Handbook	76	13.2	23
Recommended Approach	200	28.6	30
Ada Supplement	33	5.0	10
Software Measurement Guidebook	131	20.6	20
C Style Guide	89	3.7	4.5

Table 2. SEL Training Course Cost and Schedule Data

Course	Class Hours	Effort (staff-months)	Schedule (months)
Orientation	6	*	2
Task Leader/ATR Course	12	4.2	5
Recommended Approach	24	8.8	6
Principles of Flight Dynamics	30	*	7

* Effort data not available

The calendar time required to develop a package is harder to predict. For guidebooks, it appears that the schedule is driven by the scope of the material; the broader the subject, the longer it will take. For training, schedule depends more on the length of the course and the level of detail presented. Our experience indicates that time to develop generally depends on how new and different the material and/or the format is and how difficult it is to capture the experience. For example, capturing process information for the *Recommended Approach* and *Ada Supplement* took much longer than distilling information for the *C Style Guide*, which addressed a single product standard.

4. Lessons Learned

Over the past 12 years, the SEL has tried different approaches to packaging the software process. We have continually improved both our packaging process and our products based on user feedback and measured results. Some guidelines follow for producing successful guidebooks and training courses based on our lessons learned.

Standards Should Reflect Local Experience

Standards should be based on the best practices of *what is actually done* locally rather than *what an outside source says should be done*. Developers and managers tend to ignore standards that have little or no connection with their real world. It's best to introduce the most promising new methods and techniques from ongoing experiments so that the guidebooks will be current when released. Where possible, address the problem areas identified by the developers during the interviews and workshops.

It's important to clearly state what is expected to be done and provide guidance for decision-making and tailoring. Rarely will a methodology be applied exactly as it is specified; therefore tailoring guidance is extremely important. Don't overload the guidebooks with rationale. If the methodology is based on local experience, the rationale will be evident to most users. Training courses provide an opportunity to elaborate on the methodology in an interactive setting, and they are an excellent vehicle for demonstrating its proper application using local examples.

Design Packages for Ease of Use

Our interviews with developers indicate a typical usage pattern for SEL packages: developers initially read a guidebook cover-to-cover or attend a training course to get the whole story and then they primarily use the guidebooks and training materials as references during project execution. Therefore, the packages are designed with this type of use in mind. We have found several key attributes that help usability:

- Keep documents small. It's best if they can fit in a briefcase.
- Make information easy to locate. Use graphics to guide the eye.
- Use clear direct language and local terminology.
- Use graphs and pictures to clarify text.
- Provide a good, hierarchical index.

Treat Developers as Customers

Developers and managers are the primary users of guidebooks and training courses. The products are intended to help them do their jobs better. When soliciting their input to support a packaging effort, we need to treat them as customers:

- Listen to them.
- Solicit their requirements.
- Build useful and usable products for them.
- Keep them involved in package development.
- Don't ask them to do the work. (They are busy building software.)

Dedicate a Small, Highly Skilled Team to the Packaging Effort

The packagers' job is to gather, distill, and communicate information based on a thorough understanding of the environment. This requires additional skills that are not commonly found among software developers. For example, people skills are extremely important. Packagers need to be good listeners and interviewers to elicit information from people. They must be able to analyze and synthesize information and

then organize and present the resulting process effectively. Desktop publishing and technical writing skills are also critical to the quality of the final product.

We have found that a team of three people tends to be optimum for developing packages. The team is composed of

- A lead writer or course developer who has experience in software engineering, but not necessarily in the local domain;
- A domain or subject matter expert—usually a manager or senior developer who has extensive experience in the local domain (for courses, this person may also be the instructor);
- A publication specialist, who has expert presentation, editing, and desktop publishing skills.

Don't Get Bogged Down in Detail; Update Packages Judiciously

Programmers get annoyed and confused when they are constantly receiving updates to the standard process. The amount of maintenance required to keep standards current depends on the level of detail in them. We found that it is best to avoid the detailed "how to" information; instead, build in space for the process to evolve, which is happening all the time in an improving organization. Interim changes, such as updated cost models and new techniques, can be integrated into training courses and tools. For example, the SEL updated the Recommended Approach course twice in 18 months to reflect change in the local process; whereas the guidebook on which the course is based has been updated only once in 6 years since its deployment in its revised form.

Don't include experimental processes in the standards until they have been tried and proven beneficial in the local environment. In addition, keep language-specific standards separate from process information, because they tend to change independently. Let the amount of change in the organization, process, or environment drive when guidebooks and training courses are updated, rather than a fixed time interval.

5. Summary

For process improvements to be effectively infused throughout an organization, they must be packaged in a useful form. Effective standards, guidebooks, and training courses cannot be produced by programmers in their "spare" time. It takes a focused effort from a team of dedicated people with the proper skills to capture, synthesize, and communicate the improved software process. The SEL has demonstrated its commitment to broad-based improvement by investing both time and money in experience packaging. As a result, we have produced the high-quality guidebooks and training courses described in this paper. The investment has paid off—as it will for other organizations committed to managing their software process. Today, software developers in the SEL are building software faster, better, and cheaper using many techniques and methods that were considered experimental only a few years ago.

Acknowledgment

The authors thank Maureen McSharry for her editorial help in producing this paper and the corresponding presentation given at the Nineteenth Annual Software Engineering Workshop.

References

1. McGarry, F., G. Page, V. Basili, et al., *An Overview of the Software Engineering Laboratory,* Software Engineering Laboratory, SEL-94-005, December 1994

2. Landis, L., F. McGarry, S. Waligora, et al., *Manager's Handbook for Software Development (Revision 1),* Software Engineering Laboratory, SEL-84-101, November 1990

3. Landis, L., S. Waligora, F. McGarry, et al., *Recommended Approach to Software Development (Revision 3),* Software Engineering Laboratory, SEL-81-305, June 1992

4. Condon, S., M. Regardie, M. Stark, and S. Waligora, *Cost and Schedule Estimation Study Report,* Software Engineering Laboratory, SEL-93-002, November 1993

5. Kester, R., and L. Landis, *Ada Developers' Supplement to the Recommended Approach,*

Software Engineering Laboratory, SEL-81-305SP1, November 1993

6. Doland, J., and J. Valett, *C Style Guide*, Software Engineering Laboratory, SEL-94-003, August 1994

7. Green, S., *Software Engineering Laboratory Cleanroom Process Model*, Software Engineering Laboratory, SEL-91-004, November 1991

8. Hendrick, R., D. Kistler, and J. Valett, *Software Management Environment (SME) Concepts and Architecture (Revision 1)*, Software Engineering Laboratory, SEL-89-103, September 1992

9. Doland, J., R. Pajerski, and S. Waligora, *Software Engineering Laboratory Training Plan*, Software Engineering Laboratory, SEL-93-TP1, September 1993

Closing the Loop on Improvement: Packaging Experience

Sharon Waligora
Linda Landis **Jerry Doland**

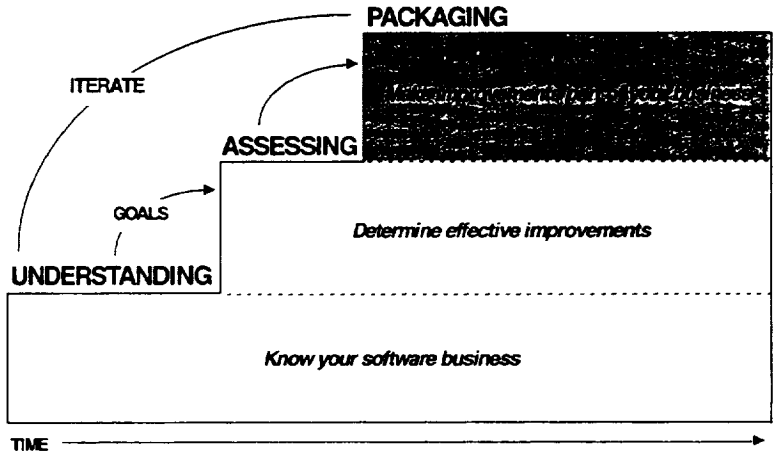
Computer Sciences Corporation

Topics

- What is Experience Packaging?
- The SEL Packaging Process
- Cost and Schedule
- Package Development Guidelines

Where Does Packaging Fit In?

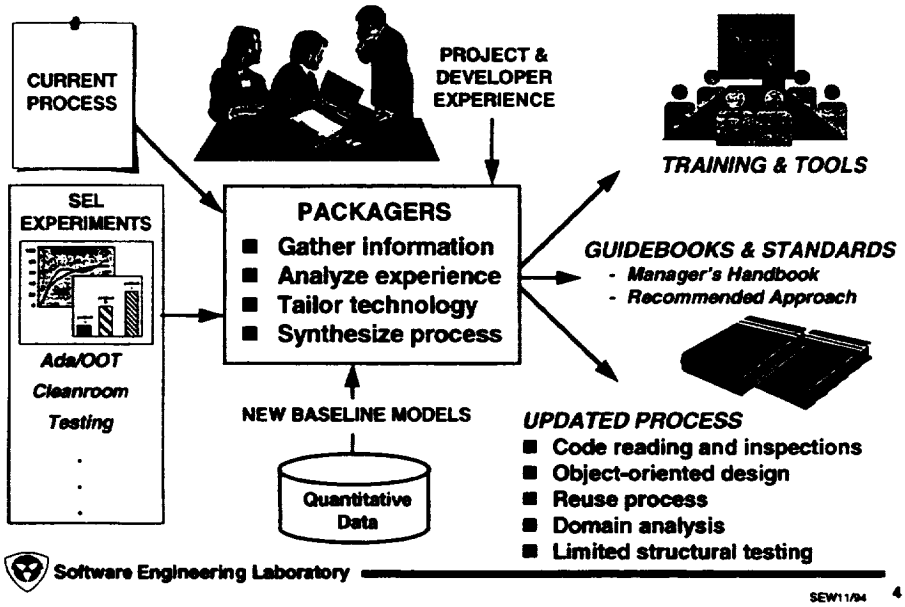
SEL Process Improvement Paradigm



Software Engineering Laboratory

SEW11/94 3

What is Experience Packaging?



Software Engineering Laboratory

SEW11/94 4

SEL Guidebooks, Tools, and Training

Guidebooks

Baseline Packages:
- Manager's Handbook
- Recommended Approach to Software Development
- Guide to Cost Estimation

Tailored Packages:
- Ada Supplement to Recommended Approach
- C Style Guide
- Cleanroom Process Handbook

Training

- SEL Training Plan

Courses:
- Orientation
- Principles of Flight Dynamics
- Recommended Approach
- Task Leader/ATR

Tools

- Software Management Environment
- Automated data collection

Are SEL Packages Used?

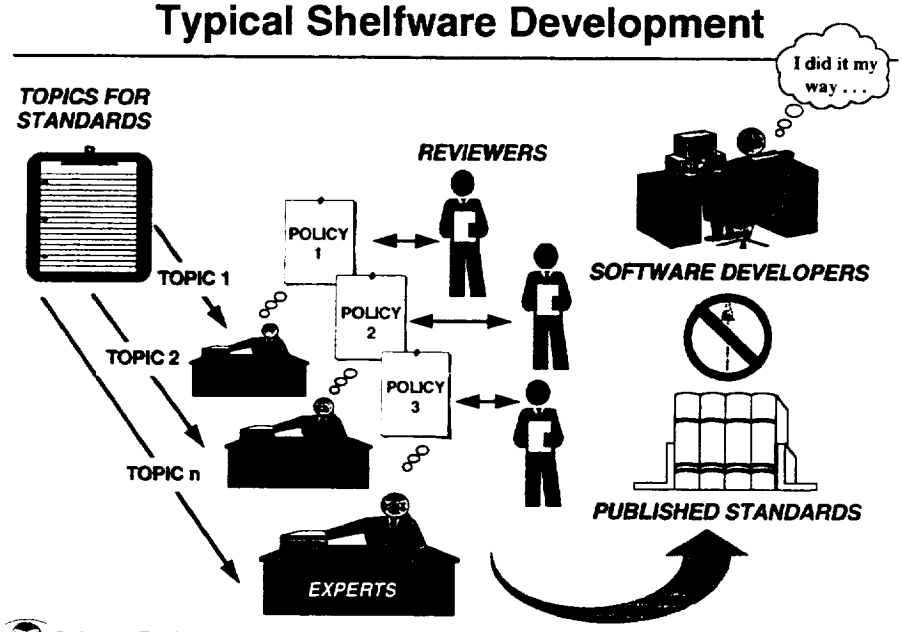
Guidebooks

- **Survey of 110 developers, maintainers, and managers**
  - **89% have used guidebooks**
  - **76% of project leaders and managers use guidebooks regularly**
  - **95% of developers found the guidebooks fairly easy to use and understand**

Training

- **Course evaluations show**
  - **95% of participants found course content directly applicable to their jobs**
  - **Most participants felt training was valuable and worth their time**

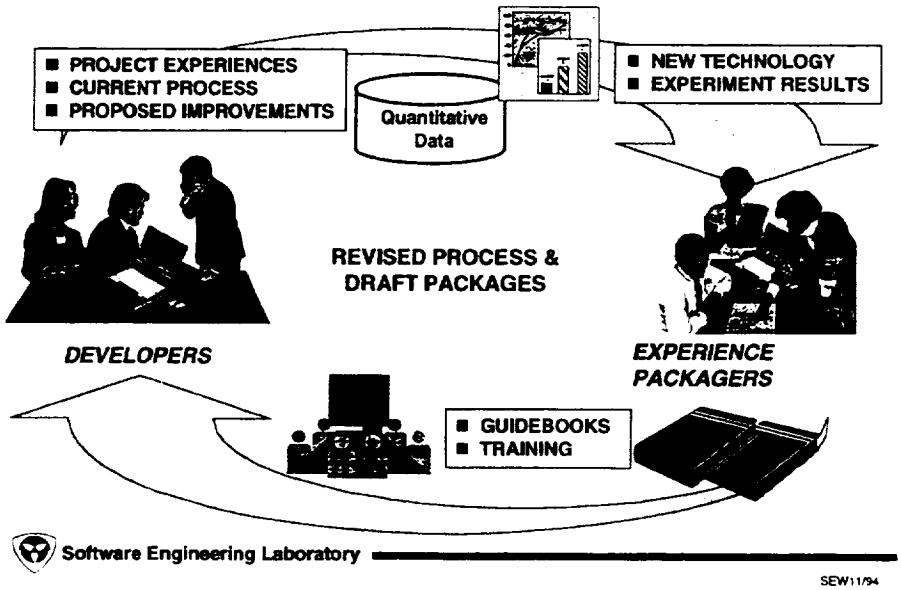
Typical Shelfware Development



Software Engineering Laboratory

SEW11/94 7

SEL Experience Packaging Process



Software Engineering Laboratory

SEW11/94 8

How Much Does a Package Cost?

- Cost models
  - Guidebooks = 24 staff-hours per page
  - Training = 55 staff-hours per class hour
- 1.5% of software budget is spent on packaging
  - Approximately 1% on guidebooks
  - Approximately 0.5 % on training courses

Guidebooks	staff-months	pages
Manager's Handbook	13.0	76
Recommended Approach to Software Development	28.6	200
Ada Supplement	5.0	33
C Style Guide	3.7	89

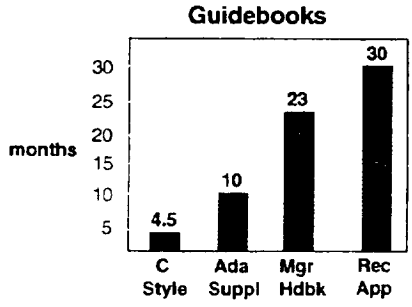
Training Courses	staff-months	class hours
Orientation Course	*	6
Principles of Flight Dynamics	*	30
Recommended Approach	8.8	24
Task Leader/ATR	4.2	12

* data unavailable

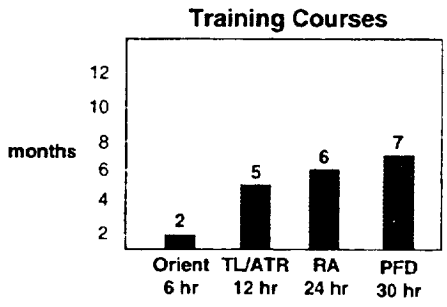
Note: Process improvement is 10% of total SEL software budget

How Long Does it Take to Develop a Package?

- Time to develop is dependent on
  - Novelty of material and format
  - Availability of packagers
  - Ease/difficulty of capturing experience data



Driven by scope of material



Driven by length of course and level of detail

Guidebooks Should Reflect Local Experience

- Document the best practices
  - From local domain experiences (what *is*)
  - Not from outside experts (not what *should be*)
- Introduce most promising new methods and technologies from successful experiments
- Address problem areas identified by developers
- Clearly state what process/method is expected
- Provide guidance for decision-making and tailoring
- Do not overload with rationale
- Use training courses to expand, show good examples, and to explain rationale

Design Packages for Ease of Use

- Keep documents small
- Make information easy to locate (index, icons, graphics)
- Present material clearly and directly
- Use local terminology
- Prototype to get early feedback on package "look and feel"

If you can't find information and understand it, you can't use it.

Treat Developers as Customers

- Developers are the users for guidebooks and training
- Experienced developers are the key source of information
- Treat developers as customers
  - Listen to them
  - Solicit requirements
  - Build useful products
  - Keep them involved in package development
  - Don't ask them to do the work

Dedicate a Small, Highly Skilled Team to Packaging Effort

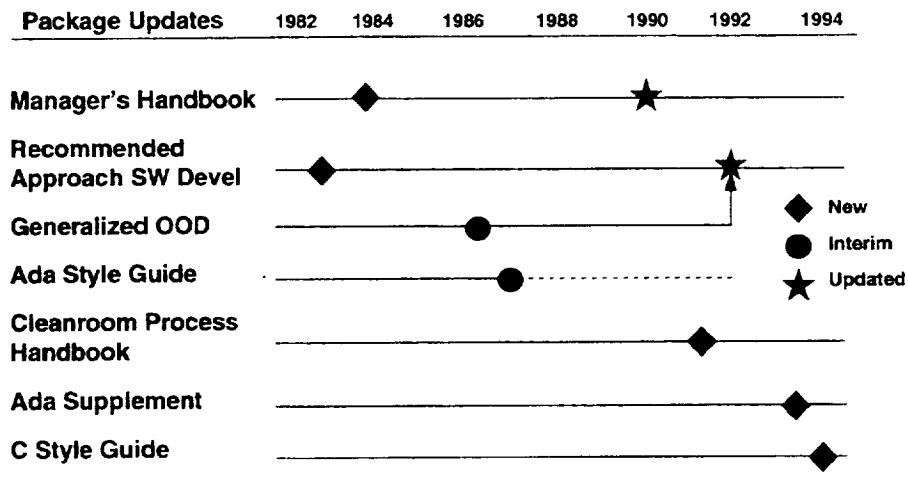
- Packagers gather, distill, and communicate information based on a thorough understanding of the environment
- Packaging team
  - Lead writer or course developer
  - Domain or subject expert
  - Publication specialist

- Packager expertise
  - Software development experience
  - Good interviewing and listening skills
  - Strong interpersonal skills
  - Able to analyze, synthesize, and organize information
  - Presentation skills
  - Technical writing
  - Desktop publishing

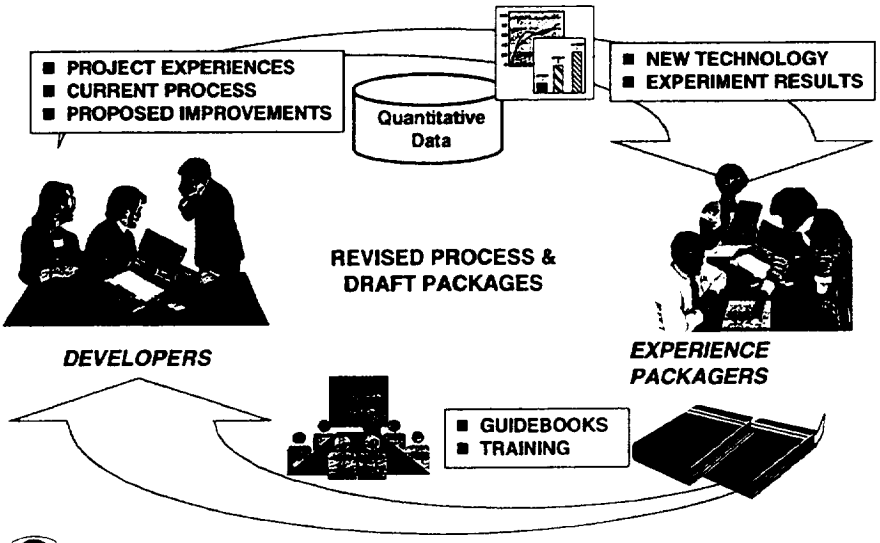
Update Packages Judiciously

- Guidebook maintenance depends on level of detail
  - Avoid detailed "how-to" information
  - Build in space for process to evolve
- Use training courses and tools to integrate interim changes
  - Cost model updates
  - New technique guidelines
- Include experimental processes when proven locally
- Separate language standards/style from process descriptions
- Update guidebooks when the organization, process, or environment changes significantly

SEL Package Development History



SEL Experience Packaging Closes the Loop



Software Engineering Laboratory

SEW11/94 17

Session 2: Process

Process Maturity Progress at Motorola Cellular Systems Division
Alan Willey, Motorola

The Personal Software Process: Downscaling the Factory?
Daniel Roy, Software Engineering Institute