Lessons Learned in Deploying Software Estimation Technology and Tools

Nikki Panlilio-Yap and Danny Ho
IBM Canada Ltd.

Abstract

Developing a software product involves estimating various project parameters. This is typically done in the planning stages of the project when there is much uncertainty and very little information. Coming up with accurate estimates of effort, cost, schedule, and reliability is a critical problem faced by all software project managers. The use of estimation models and commercially available tools in conjunction with the best bottom-up estimates of software-development experts enhances the ability of a product development group to derive reasonable estimates of important project parameters.

This paper describes the experience of the IBM* Software Solutions (SWS) Toronto Laboratory in selecting software estimation models and tools and deploying their use to the laboratory's product development groups. It introduces the SLIM* and COSTAR* products, the software estimation tools selected for deployment to the product areas, and discusses the rationale for their selection. The paper also describes the mechanisms used for technology injection and tool deployment, and concludes with a discussion of important lessons learned in the technology and tool insertion process.

1.0 Introduction

Developing a software product involves estimating project parameters such as effort, cost, duration, and reliability. Estimates are crucial to developing the project schedule and allocating the necessary staff and resources. Estimating is typically done in the planning stages of the project when there is much uncertainty and very little information. Nonetheless, estimation is very important to software development since it forms the basis for project planning and management. It is a cross life-cycle discipline that applies to all phases of the development life cycle. During the course of running the project, constant re-estimation is vital to assess the risks at various stages of the project. In some situ-

---

*

- ***IBM***, ***AS/400***, ***OS/2***, ***AIX*** and ***BookManager*** are registered trademarks of International Business Machines Corporation.
- ***SLIM*** is a registered trademark of Quantitative Software Management, Inc.
- ***COSTAR*** is a trademark of Softstar Systems.

Nikki Panlilio-Yap is on a leave of absence from IBM Canada Ltd. and can be reached at Loral Federal Systems, 6600 Rockledge Drive, Bethesda, Maryland 20817, U.S.A. Her e-mail address is nikki@lfs.loral.com on Internet.
Danny Ho can be reached at IBM Canada Ltd., 844 Don Mills Road, North York, Ontario M3C 1V7, Canada. His e-mail address is danho@torolab2.vnet.ibm.com on Internet.

ations, the estimates have to be revised and the project has to be rescheduled.

This paper captures the experience of the IBM SWS Toronto Laboratory in deploying software estimation technology and tools, and summarizes the key lessons learned.

2.0 Estimation Technology and Tools Deployment

The deployment of software estimation technology and tools in the IBM SWS Toronto Laboratory [10] consisted of three major stages as illustrated in Figure 1. Activities associated with each stage are shown; each stage is described in the following subsections.

2.1 Understanding - The Early Stage

The Software Engineering Institute (SEI) self-assessment conducted by the IBM SWS Toronto Laboratory in 1991 revealed a critical need for software estimation techniques and tools. Probably the best tools for estimation are those that use models based on historical data from one's own organization or environment [1, 4]. In the absence of an internally developed tool based on historical data from the IBM SWS Toronto Laboratory or from similar IBM laboratories that develop multiple software products across multiple hardware platforms, it is logical and practical to use one or more commercially available estimation tools. Some of these tools have underlying models based on thousands of software development projects from industry. These tools typically use input on the size of the product to be developed, project constraints, characteristics of the development team, complexity of the product, and characteristics of the development environment.

The Tool Evaluation and Introduction Process described in Ho [7] was adopted in conducting pilots and early experiments. Once several promising tools and vendors had been selected, the vendors were requested to send detailed information or demonstration diskettes of the tools for evaluation. Pilot experiments with some software-development projects were also conducted by obtaining trial licenses or borrowing tools available at other IBM Canada Ltd. sites.

2.1.1 Criteria Used in Tool Selection

Several criteria were used to evaluate software estimation tools. Required basic features include the ability to:

- Give accurate estimates
- Perform automatic recalculation whenever some parameters are altered
- Break down the estimates into different phases of the development life-cycle
- Support different software sizing methods.

Some desirable and advanced features are the ability to:

- Track project actual data
- Conduct re-estimation if needed
- Perform what-if analysis to experiment with different parameters
- Be extensible to include user-specific parameters
- Be adaptable to user-specific development environments.

2.2 Installation - Making the Selected Models and Tools Available

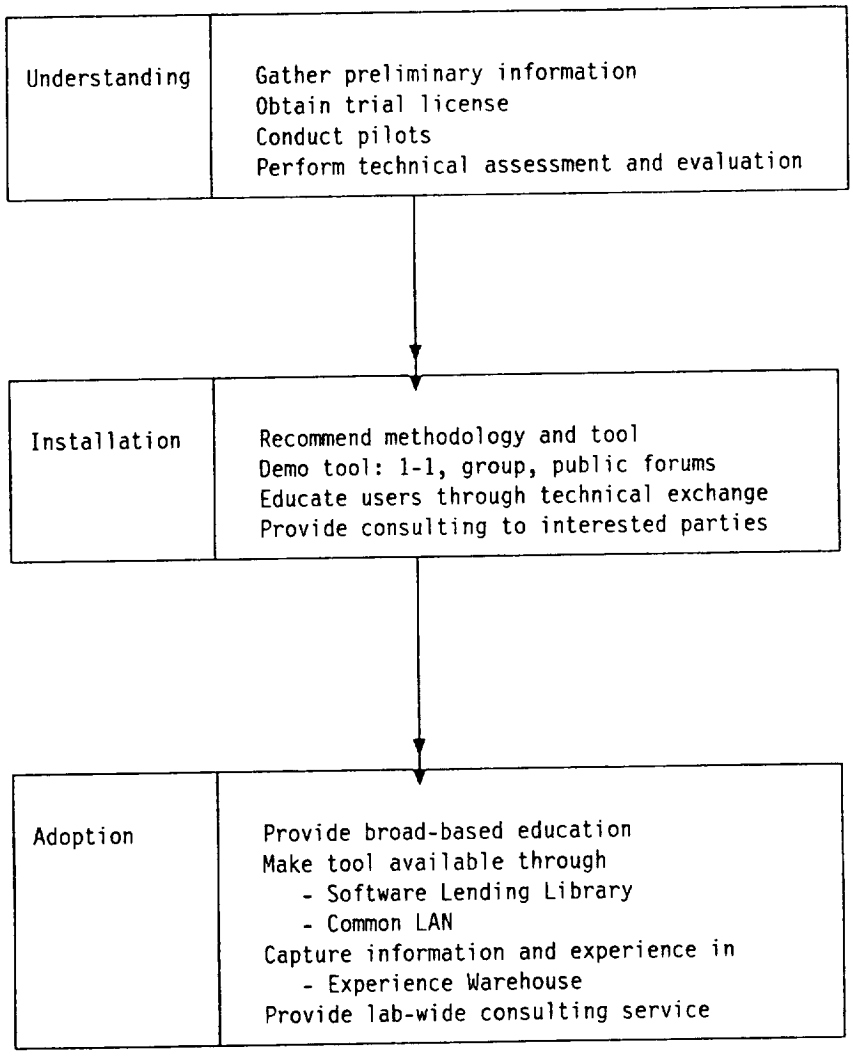

Figure 1. Stages of Software Estimation Technology and Tools Deployment

2.2.1 The Selected Models and Tools

The final decision in the choice of software estimation techniques and tools depended on the results of the pilot experiments. Both the SLIM and COSTAR products satisfied the basic requirements and possessed some desirable features for good software estimation tools. Both tools produced good pilot results.

The amount and complexity of input required for these tools is not nearly as cumbersome as that required for some other commercially available tools. In addition, the underlying theory of the SLIM and

COCOMO models is well-known and published in the public domain. Two models were adopted because neither one gave 100% accurate estimates. The use of more than one model may make up for some of the shortcomings of each one.

2.2.1.1 The COCOMO Model and the COSTAR Tool: The COnstructive COst MOdel [5] is a mathematical model that estimates the duration, staffing level, and cost of software projects. The model makes use of the effort equation as its fundamental calculation, using lines-of-code (LOC) as its fundamental input.

$$Effort = K_1 \times KDSI^{K_2}$$

where

Effort is in staff-months.
K_1 and K_2 are constants whose values are dependent upon the mode of development.
KDSI is kilo-delivered source instructions.

The effort equation is refined by multipliers from product, computer, personnel, and project parameters. The calculated effort also forms the basis for estimating the project duration and staffing.

The COSTAR tool, a DOS-based estimation product by Softstar Systems [8, 15], implements COCOMO. COSTAR 3.0 is currently deployed in the laboratory. Estimates are provided for the intermediate and detailed models, and estimation can be performed in a structured manner using subcomponents. The output consists of a development summary and a variety of reports.

2.2.1.2 The SLIM Model and Tool: The Software LIfe-cycle Model (SLIM) is a metrics-based estimation model developed by Putnam [11, 12], using validated data from over 3000 projects from industry. The projects are stratified into nine application categories ranging from microcode to business systems. The category into which most of the IBM SWS Toronto Laboratory products fall is system software.

The following gives the key equation for the SLIM model.

$$ESLOC = PP \times (\frac{PY}{b})^{\frac{1}{3}} \times Y^{\frac{4}{3}}$$

where

ESLOC is executable source lines-of-code.
PY is effort in person-years.
Y is duration in years.
b is a special skills factor that is a function of system size.
PP is a productivity parameter that translates into the productivity index (PI).

The formula is used to establish a cost-and-time schedule for development of a system of certain size. The productivity parameter can be baselined through historical project data and mapped through a translation table to the productivity index (PI). PI is a macro measure of the total development environment. It possesses different averages and deviations for different application categories.

The SLIM tool is a software product that embodies the SLIM model. It was developed by Quantitative Software Management, Inc. (QSM). The tool can be customized to a specific organization through calibration using historical data. It automates the calculation of the optimum solution based on project assumptions and constraints. It also

has a rich set of what-if capabilities for the assessment of time, effort, and cost risks. A more detailed description of the tool and its capabilities can be found in [6], [9], [13], and [14].

2.2.2 Demonstrations and Technical Exchange Sessions

During the course of injecting software estimation techniques and tools, the SLIM and COSTAR products were demonstrated on different occasions:

- To individuals in one-on-one sessions
- To development teams in group sessions
- In public forums such as conferences and tools expositions.

2.2.3 Limited Consulting

In addition to the demonstrations and technical exchange sessions, in-depth consulting was offered to a number of projects whose personnel showed commitment to learning and using the selected software estimation techniques and tools. We sat down with project managers, planners, and other key project personnel, and walked them through the software estimation process with the aid of the selected tools. We also provided analysis and interpretation of the estimates and tips on their use.

2.3 Adoption - Expanding the User Base

2.3.1 Broad-Based Education

To increase the penetration of software estimation techniques and tools within the laboratory, we developed a two-day course. Its objectives were to:

- Teach the underlying theories of the SLIM and COCOMO models
- Provide in-depth training on the SLIM and COSTAR tools
- Provide hands-on experimentation with the tools.

2.3.2 Tool Availability

One of the most important tasks in deploying promising tools is to make them available throughout the laboratory. The target users for the SLIM and COSTAR tools are primarily planners, project managers, and team leaders.

Since the majority of the laboratory community is LAN-connected, the Toronto Lab Common LAN [7] is used to make the tools generally available. The Common LAN is basically a collection of OS/2* file servers, AIX* file servers, and end-user OS/2 and AIX workstations, connected by multiple token rings. A license control mechanism limits the number of users concurrently accessing the tools to the maximum license count. The mechanism also provides a means to electronically invoke the tools in a more automated manner, as opposed to traditional manual software distribution.

The Software Lending Library is a central location used to distribute the tools to non-LAN users. A user who signs out a software package is given two weeks to experiment with the software. When the software is returned to the library, an online survey is sent to the user to gather feedback on the tool.

2.3.3 Information Availability

Availability of tools must be accompanied by availability of tool information and ease of access to the information. Tool information is accessible from:

- The Laboratory Experience Warehouse (EW) — the Toronto Laboratory's version of an Experience Base used to

store some forms of packaged experience as described in the Experience Factory concept proposed by Basili and his colleagues [2, 3]. It is a central repository for a wealth of information useful to the software development community. Its tool section consists of four matrices collecting information on tools under evaluation, under pilot (unsupported), supported (by the Tools Support Group), and rejected (not promoted). The tools within each matrix are grouped by development life-cycle, and the tool documents can be accessed through BookManager* hypertext links. The information includes some general description of the tool, formal evaluation report, and user feedback.

- The Window on the World (WOW) utility is an online utility to retrieve information for quick reference. Information for supported tools is kept on WOW. This includes general description, operation, licensing constraints, installation, environment constraints, invocation mechanism, support, and license agreement.
- The Software Lending Library was described in the previous section. Available information includes tool description, user feedback, mechanism for requesting the tool center of competence to contact the user and provide consulting, and manuals of the tools accessible through the Common LAN.

2.3.4 Lab-Wide Consulting

As more and more project groups demonstrated a need, we made software estimation consulting services available to the laboratory's development community. Because of resource constraints at the laboratory level, most consulting was provided to the project groups through project personnel who had been trained on the use of the software estimation techniques and tools. This allowed the project groups to develop their own local experts. It also allowed us to provide service to more development project groups.

2.4 Level of Deployment

Five sessions of the Software Estimation and Tools course have been offered to the laboratory. Over 70 laboratory personnel covering all major sub-business areas of the laboratory have been educated on the use of the software estimation tools. Several projects from each sub-business area have experimented with or used the SLIM and COSTAR tools. Client contacts have been established within and outside the laboratory. Five of the seven products submitted by the laboratory to the Market-Driven Quality (MDQ) Assessment in 1993 stated that they used estimation models and tools as their initiatives to improve their overall estimation process and the accuracy of their project estimates.

3.0 Key Lessons Learned

The experience we have described is based on over three years of solid work. The process we have followed can be applied in general to the deployment of other techniques and tools. Following are the key lessons learned from this experience.

3.1 Technology Injection Takes Time

Deploying state-of-the-art technology and tools takes time. Table 1 shows the elapsed time for each stage of deployment and the effort required on the part of the technology champions for each stage shown in Figure 1.

Table 1. Time and Effort for Technology Injection

Deployment Stage	Time (months)	Effort (PMs)
Understanding	7	7
Installation	12	14
Adoption	18 (On-going)	21 (0.2 PM/mo)

It took 37 calendar months and 42 person-months (PMs) of effort on the part of the champions to inject the technology and tools to the point where only 0.2 person-months per month is now required to maintain the level of deployment.

Users have to overcome many barriers to become knowledgeable in the field. In addition to learning the methodologies and tools, they have to learn about accessing the tools through the LAN or installing the tools (if not LAN connected). In some situations, users may have to configure, install, or upgrade certain components of the operating system and learn about it prior to using the tools. These are overhead tasks the users must face before any true benefit in adopting the methodologies and tools can be realized.

3.2 Management Commitment Is Essential

Long-term management commitment is essential to the successful deployment of technology and tools hitherto foreign to an organization. Management support is critical for both the consultants and the client organizations in terms of time and resource allocation to tackle the overhead tasks, education, cost of software and hardware, etc.

3.3 Champions Must Be Pro-active and Proficient

The technology champions must be in a position to give advice, provide consultation, and offer assistance. They must be able to conduct thorough analyses of project estimates, and point out both the strengths and weaknesses of the methodologies and tools to their clients.

3.4 Easy Access to Tools and Information Facilitates Deployment

The Toronto Lab Common LAN facilitates license sharing and tool invocation. There is a cost-saving benefit since acquisition of individual copies of software for each end user is avoided. Furthermore, end users are relieved from the burden of tools upgrade and maintenance.

It is important to document tool information, formal tool evaluation results, pilot results and user feedback, and to keep these documents up-to-date. The use of online surveys captures valuable tools experience that will benefit the other users within the laboratory and will help in defining the strategy for software estimation techniques and tools in the future.

3.5 Increasing the Laboratory Community's Awareness Promotes Buy-In

Demonstrations and technical exchange sessions are useful for introducing new technology and tools to the laboratory. These occasions have given some people an increased awareness in the area of software

estimation and allowed others to gain in-depth technical knowledge.

3.6 Broad-Based Customized Education Is Effective

Broad-based customized education is highly effective and rewarding. We strongly encourage the same infrastructure in deploying technology and tools in other areas. It saves the organization money. Course participants typically get more value from a course taught by local experts using real development data collected within the laboratory on more than one model and tool, compared to one taught by a tool vendor. Vendor courses tend to teach limited theory and are confined to their product offering.

3.7 Historical Data Collection Is Crucial

The collection of historical data is critical to process improvement. There is a crucial need to continuously capture historical data on in-process project parameters. The estimated and actual values of the schedule, resource allocation, defects, etc. should be collected to improve the quality of subsequent estimation. Having this data is critical for calibrating commercially available estimation tools and tuning them to the development environment.

3.8 Understanding How Data Will Be Used Is Essential

Many software developers resist capturing estimates and the actual values of project parameters They are afraid of how the numbers or measures will be used or misused by management or other groups. It is important to make them understand that the collected data will help managers identify strong points and bottlenecks, and help them set realistic goals for future software development projects.

4.0 Future Directions

Although the SLIM and COSTAR tools have been successfully deployed, much work still remains. In addition to the technology injection techniques discussed in the earlier sections (for example, demonstrations, lectures), users group meetings should be conducted periodically to update the users on the latest developments or breakthroughs. The group meetings will also provide opportunities for the users to exchange ideas and experience.

Another area that requires immediate attention is the technical assessment, evaluation, and recommendation of size estimation techniques and tools. Size estimates are critical inputs to software estimation models and tools. Other related activities that complement estimation are tracking and project management. The feasibility of integrating software estimation tools with project management tools should also be investigated.

As product development groups switch from the traditional approaches to object-oriented development, the models for software estimation are expected to change accordingly. It is unclear at this moment how well the existing software estimation models apply to object-oriented software development.

5.0 Acknowledgements

The authors thank Ann Gawman who provided valuable editorial improvements.

References

[1] John W. Bailey and Victor R. Basili, "A Meta-Model for Software Development Resource Expenditures," *Proceedings of the IEEE Fifth International Conference on Software Engineering*, 1981.

[2] V. R. Basili, "Software Development: A Paradigm for the Future," *Proceedings of the IEEE 13th International Computer Software and Applications Conference*, 1989.

[3] V. R. Basili, G. Caldiera and F. McGarry, et al, "The Software Engineering Laboratory - An Operational Software Experience Factory," *Proceedings of the IEEE 14th International Conference on Software Engineering*, 1992.

[4] V. R. Basili and N. M. Panlilio-Yap, "Finding Relationships Between Effort and Other Variables in the SEL," *Proceedings of the IEEE Ninth International Computer Software and Applications Conference*, 1985.

[5] Barry W. Boehm, *Software Engineering Economics*, Englewood Cliffs: Prentice-Hall, Inc., 1981.

[6] Kelvin B. Fowler, A Software Cost Estimating Tool for IBM., IBM ASD-Bethesda, Technical Report 86.0021, 1991.

[7] Danny Ho, Deploying Software Development Tools on the Toronto Lab Common LAN, IBM PRGS Toronto Laboratory, Technical Report 74.112, 1993.

[8] Danny Ho, Software Estimation Using the COCOMO Model and the COSTAR Tool, IBM SWS Toronto Laboratory, Technical Report 74.135, 1994.

[9] Nikki Panlilio-Yap, Software Estimation Using the SLIM Tool, IBM Canada Ltd. Laboratory, Technical Report 74.102, 1992.

[10] Nikki Panlilio-Yap and Danny Ho, "Deploying Software Estimation Technology and Tools: The IBM Software Solutions Toronto Lab Experience," *Proceedings of the Ninth International Forum on COCOMO and Software Cost Modeling*, 1994.

[11] Lawrence H. Putnam, *Software Cost Estimating and Life-Cycle Control: Getting the Software Numbers*, New York: The Institute of Electrical and Electronics Engineers, Inc., 1980.

[12] Lawrence H. Putnam, *MEASURES FOR EXCELLENCE Reliable Software on Time, within Budget*, Englewood Cliffs: Yourdon Press., 1992.

[13] Quantitative Software Management, Inc., SLIM Version 2.2 User Manual, McLean, 1991.

[14] Quantitative Software Management, Inc., SLIM Version 3.1 User Manual, McLean, 1993.

[15] Softstar Systems, COSTAR Version 3.0 User Manual, Amherst, 1990.

About the Authors

Nikki Panlilio-Yap works in the Software Engineering Department, Group Technical Staff, at Loral Federal Systems (formerly IBM Federal Systems Company) in Bethesda, Maryland. She is presently on a leave of absence from the IBM SWS Toronto Laboratory where she had been part of the Software Engineering Process Group (SEPG) since its inception in August 1990. She joined IBM Canada Ltd. in July 1989 and worked in Utilities Product Evaluation for the AS/400* system. Before joining IBM, she had several years work experience in the government, academic, and commercial sectors of the computing industry. She obtained her Bachelor of Science in Chemical Engineering from the University of the Philippines, Master of Arts in Computer Science from Duke University in Durham, North Carolina, and Master of Science in Computer Science from the University of Maryland at College Park. She was a Fulbright Scholar and a World Fellowship recipient of the Delta Kappa Gamma Society International. She is a member of the Honor Society of Phi Kappa Phi and the IEEE Computer Society.

Danny Ho works in the IBM Microelectronics Toronto Laboratory as a team leader in infrared wireless development. He joined the IBM SWS Toronto Laboratory in 1989 and worked in the Tools and Technology Group for four years. His areas of special interest are software estimation, object-oriented software development, complexity analysis, tools delivery mechanisms, and tools platforms. Prior to joining IBM, Danny worked as a Software Engineer in the Communications Division of Motorola Canada Limited and was responsible for the analysis, design, and implementation of wireline and radio frequency communication systems and protocols. Danny received his Honours Bachelor of Science in Computer Science with Electrical Engineering and Master of Science in Computer Science from the University of Western Ontario. He is currently a member of the Association of Professional Engineers of Ontario.

Lessons Learned in Deploying Software Estimation Technology and Tools

November 30, 1994

Nikki Panlilio-Yap and Danny Ho
IBM Canada Limited
Software Solutions Division
Toronto Laboratory
844 Don Mills Road
North York, Ontario M3C 1V7
Canada

Internet: nikki@lfs.loral.com &
danho@torolab2.vnet.ibm.com

AGENDA

- Introduction
- Stages of Deployment
- Level of Deployment
- Key Lessons Learned
- Future Directions

DESCRIPTION OF ORGANIZATION

- Sub-businesses:
  - Application Development Technology Center
  - Database Technology
- Number of projects/products: close to 50
- Number of people: approx 1300 (approx 1000 developers)
- Skill Mix: OS/2, AIX, OS/400, VM
- SEI assessment in 1991 revealed a critical need for software estimation techniques and tools
- Joint effort by Software Engineering Process Group, and Tools and Technology Group to assess, evaluate, recommend and deploy

SOFTWARE ESTIMATION

Estimate duration, effort, cost and reliability of software development, based on product size

Why is software estimation important?

- Crucial to project schedule and staff/resource allocation
- Uncertainty of project parameters in planning stages
- Cross life-cycle discipline which applies to all phases of software development
- Vital to assess parameters at various stages of the project and re-estimate if necessary

SOFTWARE ESTIMATION

Why use estimation models?

- Form basis for disciplined planning
- Calibrate to experience
- Allow sensitivity and what-if analysis
- Provide insights in productivity and quality improvement
- Validate bottom-up estimates

Models are not perfect, so use more than one

STAGES OF DEPLOYMENT

Understanding - The Early Stage

Installation - Making the Selected Tools Available

Adoption - Expanding the User Base

UNDERSTANDING - THE EARLY STAGE

- Gather preliminary information
  - Literature search
  - Detailed information or demonstration diskette from vendors
- Obtain trial license
- Conduct pilots
- Perform technical assessment and evaluation

INSTALLATION - MAKING THE SELECTED TOOLS AVAILABLE

- Recommend the selected models and tools (SLIM and COSTAR) based on results of pilot experiments
  - Level of input required
  - Comparison with project actual data
  - User satisfaction
- Demonstrations and technical exchange sessions
  - One-on-one
  - Group
  - Public forum
- Direct project involvement - provide consultation and advise on:
  - Model and tool usage
  - Tool calibration

ADOPTION - EXPANDING THE USER BASE

- Broad-based education: two-day course
  - Teach underlying theories
  - Provide in-depth training on the selected tools
  - Provide hands-on experimentation with the tools
- Lab-wide consulting
- Tool and information availability
  - Experience Warehouse
  - Software Lending Library
  - Common LAN

LEVEL OF DEPLOYMENT

- Offered 5 Software Estimation and Tools courses
- Trained over 70 laboratory personnel
- 5 of 7 products submitted for Market-Driven Quality Assessment in 1993 have used estimation models/tools
- Established client contacts within and outside the laboratory

KEY LESSONS LEARNED

Deploying state-of-the-art technology and tools takes time

Table 1. Time and Effort for Technology Injection

Deployment Stage	Time (months)	Effort (PMs)
Understanding	7	7
Installation	12	14
Adoption	18 (On-going)	21 (0.2 PM/mo)

KEY LESSONS LEARNED

Management commitment is essential

- Need long-term management commitment of time and resources

Champions must be pro-active and proficient

- Must be in a position to give advice, provide consultation, and offer assistance

KEY LESSONS LEARNED

Easy access to tools and information facilitates deployment

- LAN facilitates license sharing, tool invocation, tool upgrade and maintenance
- Online surveys capture valuable tools experience
- Access to tool information, formal tool evaluation results, pilot results, and user feedback help others in defining strategy

Increasing the laboratory community's awareness promotes buy-in

- Demonstrations and technical exchange sessions are useful for introducing new technology and tools

KEY LESSONS LEARNED

Broad-based customized education is highly effective

- Create local focal points in the product areas
- Deploy more than one theory and tool
- Tailor course to suit local development environment
- Reduce cost

Collection of historical data is crucial to process improvement

- Improve the quality of subsequent estimation
- Calibrate commercially available tools and tune them to the development environment

KEY LESSONS LEARNED

Understanding how collected data will be used is essential

- Reduce developers' resistance to capturing estimates and actual values of project parameters
- Help managers identify strong points and bottlenecks
- Help set realistic goals for future projects

FUTURE DIRECTIONS

- User group meetings
  - Update users on latest developments
  - Provide opportunities for exchange of ideas and experience
- Size estimation and project tracking – new areas to investigate
- Software sizing, estimation, project tracking and management tools should be integrated
- Tools that truly exploit LAN
  - Client-server computing model
  - Using servers as repository for both data and software
  - Utilize remote LAN data services
- Object-oriented software development – how well do these models fit?

Session 5: Reliability and Safety

Using Formal Methods for Requirements Analysis of Critical Spacecraft Software
Robyn Lutz, Jet Propulsion Laboratory

Experimental Control in Software Reliability Certification
Carmen Trammell, University of Tennessee

Generalized Implementation of Software Safety Policies
John Knight, University of Virginia