Experiences in Introducing and Measuring Software Reuse at IBM Endicott Programming Laboratory

Patricia Stump

IBM Endicott Programming Laboratory
Dept. G95/16-4D10
17C & Glendale Drive
Endicott, NY 13760
Tel: (607) 752-6237
Email: stump@gdlvm7.vnet.ibm.com

Jim Gesacion

IBM Endicott Programming Laboratory
Dept. G85/17-3R16
17C & Glendale Drive
Endicott, NY 13760
Tel: (607) 752-6254

Abstract

This position paper describes approaches to introducing software reuse as an ingrained part of software development for the Virtual Machine / Enterprise Systems Architecture (VM/ESA) operating system and related products at the Endicott Programming Laboratory (EPL). We discuss the major inhibitors and how to overcome them. Our approach is to motivate software developers to participate without making large investments in time and effort, and to measure reuse in a simple, useful, and comprehensive way.

Keywords: Reuse experiences, reuse program introduction, reuse measurements

Workshop Goals: Learn and exchange information on reuse methods and experiences.

Working Groups: Reuse process models, Reuse and OO methods, Reuse and formal methods, Useful and collectible metrics, Reusable component certification

1 Background

The authors have been active in the Endicott Programming Laboratory's Reuse Advisory Board formed in 1990. Patty Stump has been the IBM Endicott site Reuse Champion since 1991, leading the Reuse Board in several approaches to introduce Software Reuse. She's been involved in reuse education, tool evaluation and selection, measurements, local success stories, combining reuse improvement with quality improvement goals and measurements, and exchanging experiences with other IBM Reuse site Champions. Jim Gesacion has been involved as a Reuse Board management representative since 1992 and has lead work groups to define criteria for reusable component candidates, to upgrade the EPL's Reuse Incentives Program, and Reuse advertising, as well as advocating software reuse among the EPL management team.

2 Position

Getting a new tool or process, such as software reuse, successfully introduced, supported, and ingrained in an existing team of technical people and its processes is difficult[1]. Since new software is where reuse investments are usually justified, an environment of legacy code development presents an even greater challenge for the introduction of software reuse[2]. The bulk of the EPL's VM/ESA product consists of very procedure-oriented, tightly-coupled subsystems and modules of low cohesion (EVB87), often written in Assembler programming language. When new function is created for the VM/ESA product, it must blend with old product technology, making the exploitation of new development technologies very limited. At the same time, to remain competitive, the EPL must invest in new technologies that promise productivity and quality improvements. The EPL has learned that it can demonstrate significant quality improvement by removing errors from a product in the field. However, in order to deliver new function to that product and maintain equivalent low error rates, new development technologies must be employed. Software reuse is a technology which allows the creation of more and more new function with less and less error introduction. The EPL combines scavenging of existing product code with the examination of new product function to identify all reuse opportunities.

2.1 What Seems to be Working; Identifying Informal Reuse

Spending time and effort to closely analyze the full collection of existing reusable code in the legacy operating system, for example macros, module entry points and subroutines in any language, and to then document and measure them provides many benefits. It addresses the common problem of not having a robust repository of reusable parts for developers to use, as software reuse is introduced. If many parts are not available, willing participants in the introduction of software reuse have only one choice, to create new reusable parts. This often implies an investment larger than just writing new code[3, 4], an investment which is very difficult to justify. This "seeding" of existing parts creates a library of parts of unknown certification level [5], i.e. the desired or required level of documentation, quality, testing, legal information required of reusable parts by the local Reuse Board may or may not be met. These parts may be narrow in scope, and may not justify claims in productivity improvements, since reusers may know and use them anyway. But with respect to quality, existing parts have real customer-world history and data that can be used to identify and record known quality information of parts for reusers of those parts. This level of quality is difficult to achieve through extensive internal testing and quality verification of new code. What

this repository of existing parts also provides to software developers is access to the new tools and new processes of software reuse, through the familiarity of software parts they have worked with for years. By making change as effortless as possible, people can adopt new habits of software reuse, i.e. looking for parts to use on a new project, "owning" a reusable part, and understanding documentation, quality, and test requirements of a reusable part. These habits will be required and will reap larger benefits as more formal and broader domain reuse is employed. As people understand through personal experience, the value, the tools, the requirements of good reusability, they will more naturally be lead to create software that is more reusable and less domain-specific when possible.

Scavenging of reusable components from existing software can be accomplished in many ways[6, 7], and is likely more cost-effective than building new ones in an environment where the amount of existing legacy code that is maintained and enhanced far outweighs the amount of new code produced for the same software product. The EPL uses a set of reuse characteristics when manually scanning through existing code and when working with developers to identify reusable parts. This initial repository is further refined by examining encapsulation, domain breadth, frequency of current reuse, and known quality. Though only a subset of parts may be selected to make improvements based on these criteria and resource availability factors, all items remain on the list to be searched for by reusers. Once the repository of existing reusable parts is created, it becomes a repository of opportunity for quality improvement, reusability improvement, and certification of the reusable part. Domain experts and reuse experts can decide which parts need which work. For example, some parts may easily meet certification criteria once the quality history is checked. In other cases the documentation may need to be improved, and often, testing information needs to be found and recorded. With a tool to track which criteria are met and which aren't, it becomes easier to share the "costs" of certification, both among people and over time. A module owner, or one of the reusers, or a Reuse Board representative, or anyone could help with what they can afford to do in meeting certification criteria for a reusable part. Instead of requiring an author to do all these "extras" for some new reusable component and requiring them to fit it in with their development schedule now, it can be added to the repository, and mature to certification as it gets reused, or as time permits. We see this approach as an evolution of parts to the certified level, and an evolution of the organization to the broad employment of software reuse technology.

2.2 What we Measure

Measurements are most effective when they measure the behavior that is being encouraged. The behavior the EPL Reuse Board wishes to encourage is the practice of reuse to attain quality and productivity improvements. Therefore, many types of reusable parts are measured, from as-is parts, to thoroughly proven "certified" parts. The simplest measurement is the number of parts in the repository, and how many certification criteria need to be met by the uncertified parts. This measurement is presented to teams, organizations of teams, and management.

The EPL measures quality in our process and products quite extensively. Reuse measurements need to reflect reuse's contribution to a product's quality improvement. The EPL uses IBM corporate reuse measurements[8]. The Reused Source Instructions (RSI) measurement and its derivatives, the Reuse Percentage, Reuse Cost Avoidance, and Reuse Value Added are oriented toward measuring the combined value of quality and productivity improvements. Productivity gains cannot be claimed for all uses of a high-quality reusable part because people will naturally use it over and over and teams will naturally share it. However, each instance of a certified reusable part in the product, from a quality-only perspective, represents a unique contribution to the quality of the product. If a

developer chooses to use a "certified" routine over an as-is routine, there is a different quality result in the product. There are potential services costs avoided each and every time the certified part is used. The "used instructions" (UI) (PH93) is used with an RSI which is based on total number of instances. This reuse measurement complements the existing collection of defect-oriented product quality measurements used in the EPL by showing the total amount of certified product code at its lower error rate, and its effect on the product's overall error rates and quality. These RSI-based measurements are presented to product owners, project leaders, and management on a quarterly basis, or when product development cycles produce new results.

To encourage teams to create certified reusable parts when possible, s and to improve the quality of as-is reusable parts, the Reuse Percentage measurement is used to set and track team goals.

To be able to explain reuse in terms familiar to product developers, additonal categories of reuse are measured, including code ported from one platform to another and code imported from external sources. So that everyone understands the "best" kind of reuse they can do, these reuse categories are further refined into the least desirable and most desirable based on two criteria, productivity improvement and quality improvement. This matrix helps encourage reusers to focus on the underlying requirements of "better" categories of reuse. "Best", in this case, would be a reuser modifying the reusable part as little as possible (ideally none), a reuser relying on the original owner to maintain a single copy of the reusable part, and the reuser knowing and being able to prove the quality of the reusable part. This "best" reuse, when practiced, will result in the highest quality improvement and simultaneously the highest productivity improvement.

2.3 Comparison

There are many reports on reuse experiences available. Most describe experiences in an environment of application development or systems programmed with languages that better lend themselves to reuse, such as Ada, C, and C++. We've found none that describe a reuse program that is introduced for low-cost and that continues to evolve to more mature forms of reuse.

Although our measurement methods are based on existing measurement methods, ours have been refined to focus on quality improvement, and have been expanded upon in an effort to focus on the education, and on the progress of getting reuse momentum going in a development team at the same time that the measurements are used to explain the financial costs and benefits.

2.4 References

References

[1] B. Bouldin, *Agents of Change: Managing the Introduction of Automated Tools*. Yourdon Press, 1991.

[2] J. Doll and P. Stump, "VM Quality Improvement Through Software Reuse at the Endicott Programming Lab," Tech. Rep. TR 01.C173, International Business Machines, November 1991.

[3] B. Barnes, T. Durek, G. Gaffney, and A. Pyster, "Cost Models for Software Reuse," in *Proceedings of the Tenth Minnowbrook Workshop*, July 1987.

[4] T. Bollinger and S. Pfleeger, "The Economics of Reuse: Issues and Alternatives," in *Proceedings of the Eighth Annual National Conference on Ada Technology*, March 1990.

[5] R. T. S. Center, "IBM Reuse Methodology: Qualification Standards," Tech. Rep. Z325-0683, International Business Machines, 1992.

[6] G. Caldiera and V. Basili, "Identifying and Qualifying Reusable Software Components," *IEEE Computer*, February 1991.

[7] G. Mayobre, "Using Code Reusability Analysis to Identify Reusable Components from the Software Related to an Application Domain," in *Proceedings of the Fourth Annual Workshop on Software Reuse*, November 1991.

[8] J. Poulin and W. Hayes, "IBM Reuse Methodology: Measurement Standards," tech. rep., International Business Machines, 1992, July.

3 Biography

Patty Stump is a staff programmer at IBM's Endicott Programming Laboratory, Endicott, New York. After numerous software development projects as a developer, her current primary responsibility is to coordinate the Endicott Reuse Advisory Board's efforts to introduce formal software reuse into the software product development process for EPL-produced products. Her interests include software development process models, quality improvement, and systems science. She is a member of the IBM Corporate Reuse Council, the 100X Quality Improvement team in Endicott, and IEEE Computer Society. She received her Bachelor's degree from East Stroudsburg University, Pennsylvania, and her Master's degree from State University of New York and Binghamton, New York.

James Gesacion is a first line development manager at IBM's Endicott Programming Laboratory, Endicott, New York. He is currently responsible for development of Client/Server products. His software reuse responsibilities include serving on the Endicott Reuse Advisory Board, and providing management support and focus. His interests include achieving Six Sigma Quality in products and processes. He received his Bachelor's Degree from Youngstown State University, Ohio.