Software Reuse Across Continents

Roland Martin
Hewlett-Packard GmbH Waldbronn Analytical Division
Hewlett-Packard Str.
D7517 Waldbronn, Germany
email: rmartin@hpwadrd.wad.hp.com

Gary Jackoway
Hewlett-Packard Co. Avondale Division
P.O. Box 900
Avondale PA. 19311
email: gary@avo.hp.com

Charu Ranganathan
Hewlett-Packard Co. Scientific Instrument Division
1601 California Avenue
Palo Alto, CA. 94304
email: cr@sid.hp.com

Abstract

The Analytical Products Group of Hewlett-Packard Co. is practicing software reuse at its three sites, located in the US and Europe. This paper describes the organization, goals, processes and results of a reuse program. Data from two example projects illustrate the approach. Lessons learned from these and other examples, as well as key factors for economic success, are presented.

Keywords: software reuse, reuse without modification, component library, economics

1 Introduction

Hewlett-Packard (HP) Analytical Product Group (APG) supplies instrumentation, software and systems for a wide range of analytical techniques including Gas Chromatography, Liquid Chromatography, Mass Spectrometry, Spectrophotometry and Automated Sample Preparation.

Single-user workstations and multi-user data systems with software for instrument control, data analysis, automation and information management are essential elements of APG's business.

APG has divisions at Palo Alto, CA., Avondale, PA. and Waldbronn, Germany. Each site has worldwide business responsibilty for a subset of above techniques, and does research and development (R+D) for instrumentation and software for their business segments.

Sharing of software workproducts across APG organizations began as early as 1982 with voluntary design for reuse and cooperative efforts among design teams. APG's motivation for practicing reuse has two major elements:

1. Enhance value to customers through consistency and compatibility across products
2. Improve productivity by reducing redundant development efforts

In late 1988, the first author was assigned to further stimulate and lead these reuse activities with small project teams at all three sites funded by and reporting to the APG R+D manager.

Activities include improving the infrastructure for multi-site reuse, identification of workproducts for reuse, re-engineering existing workproducts for reuse, new designs for reuse, consultation to projects designing with reuse, and maintenance of reusable and reused workproducts.

Activities are organized in a multi-site program, i.e. a set of related and coordinated, but independently managed projects.

2 Reuse Program Outline

The mission of this multi-site reuse program is to create a set of software components for reuse without modification in APG products. A subset of these components is designed for multiple platforms (i.e. hardware and operating system), others are specifically designed for one platform. APG software products are developed for HP-UX/X-Window, MS-DOS/MS-Windows, and firmware platforms.

Reusable components are expected to meet the following criteria:

* Functionality is generic within our domain: meet the need of at least three products.
* Platform dependencies are isolated.
* Internal elements, in particular data structures, are hidden through encapsulation.

Workproducts and services other than just code are essential to facilitate reuse across platforms, products and organizations. The following items are provided together with reusable components:

* Documentation: Specifications, Application Programmer's Interface, Installation Guide.
* Test suites at component level for quality control independent of an application.
* An example application.
* Consultation to consumers.
* Commitment for maintainance over several years of component lifetime.

The content of a reusable component is very similar to that of a software product. The same is true for its development process, i.e. the component life cycle. We are using a scaled down software product life cycle to create and maintain software components. A panel of R+D section managers from all three sites chaired by the program manager reviews the projects and approves phase transitions. This panel holds telephone or video conferences every two week for this purpose. A council consisting of all APG R+D Managers meets three times per year and provides strategic guidance and funding for the reuse program.

3 Results

The program started in 1989 and produced an asset of about 50 000 lines of code plus the additional workproducts for reuse without modification in multiple software products. This asset covers more than 10 percent of the lines of code released with APG software products. With the currently active producer and consumer projects we expect to increase the asset to near 150 000 lines covering more than 30 percent of the code shipped with APG software products. On average each component is reused in 4 products.

4 Examples

4.1 Signal Processing Algorithms

A set of signal processing algorithms to calculate quantitative information from chromatograms was created in the early 1980's in Pascal. It was carefully designed for portability. Several software products made use of this library. The creator was reassigned to other projects after completion. The library was subsequently ported to new operating systems by product development teams and adapted to other analytical techniques. This resulted in six different versions with subtle differences. Once a defect was found in one of the versions, it was a challenge to verify and fix all others.

Another effect was observed: the code size grew with each port or adaption. The biggest growth in code size came during a port from Pascal to C using a translator program. The original 3 kNCSS grew to 9 kNCSS, essentially without any additional functionality.

The first project within the components program was set up to re-engineer this library for strict black box reuse in all software products.

The result: size is down to 5 kNCSS, speed improved by a factor of two.

The project took about one year for two engineers (one of them the original creator). This effort included a detailed assessment of the requirements of all potential consumer projects and the creation of an extensive test suite with test cases from all analytical techniques which need this type of algorithm.

After release of this component, four products on two platforms replaced their individual versions with this one common stream of code. More products are under development using the same source without modification.

Meanwhile, the component development team added new capabilies to the library and is responsible for all maintenance activities for this component.

4.2 Command Processor

Like the first example, the command processor (CP) concept was created in early 1980's. It started as a mechanism to coordinate a collection of small programs and evolved into a language interpreter. During this evolution process the CP concept was adopted by the majority of APG R+D teams and became a key element for the architecture of APG's software products and systems. The development teams adapted code from other projects to fit the requirements of their product.

Different from the algorithm example, the perceived needs for CP capabilities are significantly different from project to project. Over time, projects teamed up and shared the same code for a few product releases but afterwards felt the pressure to implement those extensions that were most important for their product's customers. The extent of adaption ranged from small enhancements to almost complete re-writes. In the late 1980's we found ourselves with around a dozen sets of CP code. All versions do support the same core capabilities but with small, sometimes annoying differences.

It was impractical to create one single set of code for Unix and DOS. For each environment one version was identified for re-engineering. The core capabilities were identified and described in a rigid syntax guide. Product-specific elements were eliminated from the core. Mechanisms were designed and implemented that allow addition of product-specific capabilities without modifying the core source code.

Both versions of the CP library are currently in the test phase and each have delivered pre-release versions to two consumer projects. More project teams plan to use the same code without modification. An essential shared deliverable is one common validation suite to check compliance with the syntax guide mentioned above.

Size of both libraries is about 20 kNCSS. We spent about two person years for each of the two libraries. This included the creation of the syntax guide and a validation suite shared between the two.

The CP teams will continue to enhance and maintain the two components for a growing number of consumer products.

4.3 Other Reuse Activities

Ten other software component libraries of similar nature and size covering other funtions needed by three or more APG projects are in progress and could have served as examples as well. Some of them are new designs, others do re-engineering of existing functions. Many of them are sponsored and funded by APG divisions and complement those funded by Group R+D.

5 Some Lessons We Learned

From the above examples our organization learned a few lessons, including:

1. White-box reuse (i.e. reuse with modification) has marginal economic benefits.

2. Black-box reuse (i.e. reuse without modification) help us to achieve our goals, i.e. (a) to achieve consistency across products and (b) to reduce redundant development and maintenance efforts. However, it requires sustained attention over the full life cycle of components and products using these components.

3. Unsustained black-box reuse degenerates within 2 years.

4. Voluntary and cooperative reuse efforts do not last in a tough business environment. Management support and some organizational structure is necessary to sustain reuse.

6 Key Success Factors

6.1 Products

APG products cover a diverse range of analytical techniques, but share many functions, not only in software, but also in hardware. Identifiction of these commonalities is rather obvious. End users strongly demand consistency across products for these common functions.

6.2 Culture

During years of steady growth, APG built a strong base of experienced people. Best practices are exchanged among teams worldwide through formal and informal channels. This exchange of best practices results for example in:

* One product development life cycle is practiced by all teams
* The same operating systems are used by all new products
* All new code is developed in C.
* Reasonably consistent software development tools exist at all sites.

6.3 Management Support

Upper management supports and expects reuse and allows for the necessary investment within reasonable constraints. Significant effort has been spent, and will be necessary in the future to provide the environment for reuse across projects, organizations and continents, including:

* Communication tools (telephone and video conferencing, fax)
* Common development tools
* Global controlled access to reused workproducts
* Defect tracking for components
* Keeping track of who is using what (configuration management)
* Common business goals for the participants of a reuse program
* Consensus on technical concepts (architecture, user interface style)
* Compatible ways of doing things (processes)
* Validation of compatibility

6.4 People

Definitely the most important factor for the success of a reuse program are good working relationships among qualified, dedicated and experienced people in all roles and levels.

No technology can compensate for the lack of this key factor.

Unfortunately we cannot give a simple recipe for achieving this. Regular face-to-face meetings between the key people are definitely important to build and maintain such relationships. We have experienced some difficulties in our program whenever more than two months passed without personal contacts between two project teams.

7 Conclusions

Software reuse is more than just creating and/or obtaining workproducts for/from a public domain. In order to get economic benefits, it requires upfront investment and sustained effort. More than any particular technology, the consensus between producers and consumers of reusable workproducts is essential for success.

Hewlett-Packard Analytical Products Group (APG) has made a two year investment in software reuse and is beginning to see economic benefit from this program. We are prepared to sustain that effort.

8 Biography

Roland Martin, born 1946 is managing the APG reuse program. He holds a Ing.(grad) degree in electrical engineering from Fachhochschule fuer Technik, Esslingen, Germany. He joined HP's Boeblingen division in 1969 and relocated to the newly created Waldbronn Analytical Division in 1975. His past assignments included production engineer, starting up a PC board test facility, design of hardware, firmware and software for liquid chromatography, project- and section manager. He is single, lives 5 minutes from work in a small village at the edge of the black forest. Between his frequent trips to the US, he tries to keep his skills as a glider pilot up to date.

Gary Jackoway, is an R+D software engineer. He holds a Master's degree in Computer Science from Duke University, and a Bachelor's degree in Mathematical Sciences from Stanford University. Before joining Avondale Division, Gary worked on automatic routing and placing algorithms for printed circuit board design. Gary is an avid volleyball player, and a contract bridge Life Master.

Charu Ranganathan has been a software engineer at Scientific Instuments Division of Hewlett-Packard (Palo Alto) for the past 2 1/2 years. She is currently working on the Command Processor and will continue to be part of the reuse program at HP. She holds a BS in Computer Science from Bangalore University, Bangalore, India and an MS in Computer Science from University of South Western Louisiana, Lafayette, LA. She is an Indian classical musician (vocalist) and is interested in painting/sketching.