Best Practices in Software Technology Governance

Strategic reuse of software is a compelling avenue for engineering productivity

Ross Seider
On-Fire Associates

Jim Masciarelli
Power Skills Solutions

Abstract

This white paper describes the results of an industry search for best practices in software engineering operational governance and software reuse in particular. Changing a distributed organizations' reuse culture from opportunistic failure to a core managed strategy, requires new operational governance practices. The desired managerial model must be scalable, distributed, and proven in similar environments. Successful organizations' savings from duplicate development costs, fund numerous new ventures and customers benefit from easier interfaces and more consistent products.

Introduction

Strategic driven reuse of developed software is one of the most compelling avenues for rapidly increasing engineering productivity. The many gains in operating efficiencies can extend to support organizations and to end customers. In many cases, leveraging commonalities of technical assets, methods and tools is central to being competitive in the marketplace.

To achieve the benefits across business unit divisions, corporations may need to adjust certain operating behaviors and priorities. This is especially the case when the divisional culture and performance goals are strongly aligned along the business unit's boundaries (i.e. vertical alignments). This research sought to identify best in class methods and incentives to encourage cross business unit (horizontal) reuse programs hereby referred to as software reuse governance.

Large corporations often structure themselves into business unit organizations, each run as profit and loss centers. In many cases, the linkages between business units are intentionally limited, to allow the management teams maximum flexibility to achieve market success. The most important goals are associated with the vertical performance of the business unit and relatively few goals relate to horizontal, or corporate-level objectives. For business units with technology overlaps, duplication of development efforts is a common situation. When this occurs, the potential for cost savings can be significant.

Objectives

The objective of the study was to identify effective technology sharing governance methods that impose minimal disruption to the vertical focus of the business units. The study focused on software technology, in part because the sponsor's primary value was delivered in software and secondarily, software is one of the more complex of R&D deliverables to share among projects. The study specifically sought organizational structures, roles, metrics and incentives that promote technology sharing. It did not address the readily available solutions regarding technology challenges associated with sharing software.

The desired outcome was to get engineers and managers working in different business units, to selectively undertake efforts that would not directly benefit their vertical organization. Of greatest interest were the one-to-many opportunities, i.e. where multiple consumer teams utilize development from one producer team. Such efforts would be particularly valuable if the producer team agreed to support and maintain the code for the benefit of the consumer teams. Of lesser value, but still of interest, were joint development efforts (one-to-one opportunities) between two engineering teams. Such situations often arise independently of a strategic reuse effort, and are justified solely on the business self-interests of the two parties.

Discussion

A good definition of governance comes from Impact, Fall 2005, MIT Center for Technology and Industrial Policy:

"Governance is the formal and informal patterns of interaction, structures and systems that's serve to orient and connect independent stakeholders over time so as to advance their internal, separate interests and their collective system wide interests."

There are two operating strategies for software technology sharing; reuse and repurposing. Reuse is more valuable (and challenging) as it intends the developed software to be used, as is, without alteration. Repurposing implies a redesign effort for each new sharing application. The paper will discuss both, but focuses on reuse.

It is expected that governance practices that work well between business units should work at least as well within a business unit. By solving the former, an enterprise-wide solution can be created. The study identified four non-technical management practices that enable a multi-business unit reuse strategy.

- Discovery –the methods by which new technology sharing opportunities are uncovered and existing reusable technology is advertised
- Incentives / funding – the consideration that encourages the production of and consumption of sharable technology
- Controls – the practices that control the design, and obligations to support and maintain shared technology
- Sustainability – the means by which a shared technology creates a self-sustaining ecosystem, surviving business changes and the expiration of incentives.

Developing software for reuse imposes considerable short and long-term burdens on the producers. Beyond increases in the initial development complexity and validation efforts, over 50% of software life cycle costs are due to long-term support and maintenance.[i] The additional design burdens can be undertaken in parallel with the initial development or, afterwards as a post-development activity. The former yields a longer initial development cycle, and the latter requires two development cycles. The latter approach also yields two related code bases, and two support burdens. The additive effort can be performed by the designers of the application functionality, who need to apply the reuse design requirements to their code, or by re-use experts who will take the completed application and adapt it for reuse. In either case, special skills are required to prevent overly generalized, bloated and poorly performing software.

Management Reuse Models in Software Development

Numerous researchers have pointed out that an enterprise's ability to successfully create and consume reusable software is highly correlated with high maturity organizations. Systemic reuse is a strategy and isn't free. R&D's traditional "new feature by new feature" drumbeat has to be replaced by new operational considerations. Numerous technical issues (beyond of the scope of this paper) come into play as well. The bottom line is that design teams, if driven strictly by vertical BU incentives, will not undertake the burdens of reuse. Management must create offsetting considerations for strategic reuse to succeed. In fact, recognizing commonality and reuse of technology as a business and organizational strategy is key to the success of each of the following identified management reuse models:

Centralized Model

The funding of a centralized development and planning organization is the most frequently encountered way enterprises organize to produce common software.

Sharable software can be produced, delivered and maintained by a central development team, located external to the business unit. Modules are delivered to the business units for installation into products and services. The central producer group is a cost center, funded as overhead or via a "development tax" on the business units. Revenue from sales is not allocated to the producer group, but remains in the business unit. The development roadmaps for centralized teams are determined, in part, by the business unit community and executive management. For the software to be shared (rather than repurposed by the BUs), the producing group usually commits to support the code for a multi-year time period. This implies a long term funding commitment to the centralized group, its technology or platform.

Self-interest can drive the business units to utilize the common technology. The business units avoid duplicate expense and can apply their resources into other areas. Where self-interest fails to sway, management dictate usually forces the use of a common software technology.

One primary problem with the centralized model is that the producers are somewhat removed from marketplace consequences. Business unit

performance metrics and pressures, e.g. ones that associate revenue, growth and margins to what developers do and how they behave, do not directly ripple back to centralized producer groups. For better or for worse, the BUs provide a buffer. With management commitments to multi-year funding, there is less pressure on centralized groups to continuously improve software development operations.

Centralized teams have a propensity to grow bigger each year and get less responsive to their "customers". The business ratios that help guide expense budgets do not work on centralized teams funded through overhead or business units taxes.

Library Model

The library model establishes a corporate repository for reusable modules. Producing organizations are rewarded when reusable modules are donated. There are donation criteria in place to qualify that the software has reuse potential, and has met quality, performance and documentation criteria. A variant of the model has a library staff commissioning business units to produce certain modules.

The library's contents are actively promoted to consumption groups. The timely discovery of reusable technology is important because architectures are strongly influenced by the selection of major software components. Once an architecture is established, new reuse opportunities diminish.

The consuming group is responsible for validating the usefulness of the code, learning how it operates, integrating it into the project and long-term support. Each consumption event scales these costs. Producer and library costs also factor into the total economics, but are sunk costs by the time the project makes a reuse decision. A generally accepted rule of thumb; if more then 75% of the code can be reused without modification, it is more economic than designing new code. Less then that is either of marginal benefit or negative benefit.[ii]

Library based reuse models have been successful with stable, well understood, low level application areas such as statistical packages, hardware drivers etc.

A variant of the library model was in practice in Japan in the late 1980s. Companies like NEC and Toshiba used the software library like a component database, to log and characterize working software modules. In brief, they manage the software modules as if they were semiconductor parts. The distinction; the producer retained long-term responsibility for the maintenance and support.

OEM Model

There are countless companies supplying technology components in the form of software. Reuse is central to building and sustaining their business strategy. Long ago, these companies solved the challenges of building reusable components and making them available for worldwide engineering reuse. The OEM model applies their business methodology to governance.

An obvious difference between the library / central models and the OEM model is that the OEM producer has a long-term financial interest in the successful implementation by the consuming party. License fees, professional service support fees, and maintenance fees accrue from successful implementations and market success. Long-term support comes from a single producer team rather than each consumer team. The consumer groups can avoid building duplicate expertise and can put their funds to better use elsewhere. The OEM "contract" provides adequate assurances that the producer will support the consumer appropriately.

What is straightforward to implement between companies is not necessarily easy to implement between business units of a company. Creating a self-sustaining producer eco-system is challenging when the entire available "market" is limited to internal business units. Tracking license usage across product lines (and tracking cross-BU revenue recognition) adds complexity to business accounting systems. The OEM model suffers from sustainability if the reused software is unavailable to external markets.

Open Source Model

The Open Source community has a well-established model for software sharing governance.

Open Source are not-for-profit corporations, set up as virtual communities of volunteers with common interests. They are bonded by an allegiance to the mission they promote or the projects they undertake. Product roadmaps are driven by a collective altruism, community commitment, and technology interest.

Projects are run in a highly distributed manner, each governed by a separate program management committee, linked to the community by basic operational by-laws. What they produce is offered free-of-charge to licensees.

A self-sustaining volunteer eco-system is maintained by technical challenge, a meritocracy that recognizes long-term contributions and bragging rights. The producer teams provide best effort support to the end user community. New volunteers earn their initial community credentials by spending time on the maintenance team. If the software's end user base is broad enough, third parties may offer formal, for-fee support services and service level agreements.

One must be careful in drawing governance parallels between corporate development environments and Open Source communities. The commercial pressures facing enterprises and the management hierarchies they employ to drive timely decision-making are for the most part absent in Open Source communities. Decisions in Open Source organizations can take weeks as the team works towards consensus. Open Source programs rarely face the same delivery time pressures as enterprise programs, or deal with inter-project competition for scarce resources.

Open Source communities have demonstrated that distributed, self-selecting, volunteer development teams can produce complex and competitive products that parallel the best of enterprises. Open Source's collegial, consensus-driven development method does appear in rare corporate cultures and its team based code methodology is similar to X-treme programming techniques used in new software development paradigms.

Survey for best practices

The following section describes some software reuse practices at four major companies. These firms were selected because of their multi-business unit structure and their history with technology reuse. The data comes from executive interviews and public-domain documents. The narratives cover somewhat overlapping time periods, within the last 10 years, and represent practices at some of the business units within the enterprise.

Cisco Systems

Cisco Systems is organized into roughly twenty business units each made up of 150 – 750 people. They have 14,000 engineers, thousands of contractors plus outsourced programs. Their business units are centered on a technology or product set. Development and product management comprise the business unit, with other services being centralized at a division or corporate level. Due to Cisco's aggressive acquisition strategy, new products and staff are constantly being merged and integrated. Once integrated, most business units share corporate services for manufacturing, sales, finance, service and marketing.

Cisco runs three centralized development groups as cost centers. They are the network operating system platform group, and the network management platform group. These technology groups plus a tools and methods group report to the Chief Development Officer (CDO). The software developed in these groups is utilized within many Cisco products and development departments. Product sales revenues are attributed to the business units. Revenue is not back allocated to the central development group. Business units pay for maintenance service costs incurred by the centralized engineering groups. Recently, they also began paying for customization development (see sidebar story).

Usually, business units utilize the core platforms and methods. Acquisitions are rapidly migrated off legacy equivalent technology and onto the standard platforms. Business units vie for the mind-share of the central group, for work on their high priority features and capabilities. Business units can fund special developments in central engineering, or contribute developers for important features. A Cisco Technology Council settles resource contention disputes.

The Cisco technology council, comprised of senior technologists from each business unit, is chartered to promote technology reuse. The council meets once a month. Each council member is responsible for knowing the details of all projects within their business unit, and being generally aware of projects underway at all other business units. Technology sharing opportunities are uncovered and advertised during council sessions.

The council controls a modest expense budget that is used to encourage the behavior of business units. The council can commission work to be executed in one of the central engineering groups, in a business unit's development team, or at external third parties.

Motorola

In the late 1990s, Motorola was organized as five business units, each run as an semi-autonomous P&L. Certain corporate functions were provided either as an external service (e.g. Motorola University) or as a centralized service with dotted-line providers embedded within the BU (e.g. Finance or Human Resources). Each P&L was further sub-divided into BUs with multiple market and product development teams within each. Motorola overhead funded a centralized research group that had a dotted-line

management link to research groups within the business units.

The Motorola CEO's office required each of its top 35 business unit to host a long-range plan event every year. This event included a five-year forecast of technology efforts, a multi-year market outlook, as well as a long-range business plan. The business units present this information in a two day event, one day focused on technology / market futures, and the other on business plan outlook. The 35 events consumed a significant portion of the corporate calendar and required diligent preparation. It was a primary inter-BU discovery mechanism and the way the corporate officers stayed connected to the evolving businesses.

The content and formats for the outlook review were standardized. The presentation and backup materials were published one week before the meeting and attendees were expected to have read the content before arriving. The market roadmap review included analysis of competition, market share, market outlook, cost / price trends and disruptive threats and opportunities. The technology portion included technology roadmaps, experience curves, quality projections, and disruptive technology threats. The standardized materials and advanced publication allowed the sessions to be fast paced. Contentious issues, either within the BU or across BUs, were to be highlighted. The first day's review was "scored" for content, quality and presentation. A business unit's yearly performance review included this evaluation.

Depending on the business group, the first day's meeting could have very broad attendance from other BUs. Invitations to technology roadmap reviews were opportunities to strengthen professional linkages and the primary means to discover new opportunities for cooperation. The second day's meetings were for business planning executives.

Software technology sharing efforts were self-funded by the business units. There were no special financial incentives from the corporation. Motorola was one of the companies to employ an OEM license model (and/or transfer pricing mechanisms) to allocate the revenue benefits between cooperating business units. Reused software was sustained by the revenue and profits earned by successful integrations.

Apache Software Foundation and the Eclipse Foundation.

The Apache Software Foundation (ASF) was formed in 1999 as an outgrowth to work of the National Center for Supercomputing Applications (NCSA) at the University of Illinois. The NCSA had developed an HTTP (web) server platform that achieved considerable market success due to its generous licensing, functionality and reliability. For various reasons when NCSA discontinued their efforts in 1999, a user community took over support for this code-base. The Apache server has been widely accepted and is a dominant web server on the Internet. The ASF today has roughly 50 active projects.

IBM founded the Eclipse Foundation in 2003. It was established as an Open Source development effort that soon attracted a dozen sponsors from industry. Eclipse develops standardized tools and methods for object-oriented software development environments.

Both communities are set up as virtual, not-for profit (501-C3) corporations. There is a minimal corporate hierarchy and most members are volunteers. The corporate structure shields the members from third-party patent infringement claims.

Boards of Directors govern the foundations. Most Eclipse directors are nominated by the sponsoring firms. The Apache directors are meritorious volunteers, nominated by the contributing staff. The Boards are responsible for a modest level of fundraising, establishing and monitoring bylaws, running the licensing programs, protecting copyrights, and chartering new development programs.

Program management committees run each development program. The program management committees sub-divide into a 1) requirements committee (feature capture, evaluation, feature themes, priorities), 2) a planning committee (roadmaps and release themes) , and 3) architecture (design and development). The program teams are independent of each other and are chartered to promote the long-term objectives of their programs. They operate within the bylaws of the foundation. Communications is public and other programs can easily monitor the progress of any other program and uncover reuse opportunities.

The ASF directors run a new program incubator where new concepts can be temporarily parked while interest levels are assessed. The incubator filters prospective projects based on them growing into successful meritocratic communities. The ASF's concern is that a new project needs a critical mass of volunteers to sustain its execution. If the program cannot attract skilled volunteers, it is cancelled. Also, ASF will only start an incubator program if a code base exists. They have found that a collegial approach is difficult to maintain if there is no existing code base.

New program nominations for the Eclipse Board of Directors, come mainly from the Eclipse Requirements Council, a group comprised of members from the 25 sponsors.

Hewlett Packard

Of the for-profit organizations evaluated in this study, Hewlett Packard's technology reuse efforts in the HP-UX and Openview business units (now called HP Software) bears considerable resemblance to the Open Source model.

Founded in 1933 in a Palo Alto garage, HP served the corporate market with highly engineered, high quality instruments and systems The corporate culture, known as "The HP Way" was famous for high consensus and collaborative style. Organized into autonomous, vertically driven business units, development teams focused on what mattered to corporate buyers. This culture began to change in the mid-1980s with the explosive growth of PCs and ink-jet printers.

Up to that time, HP had not been known for agility or responsiveness. Moreover, the attributes valued by retail customers and distribution channels were different than the values desired by corporations. For example, while HP was able to serve the corporate market with just 5 models of printers, it needed 50 models to support the retail market.[iii] Retail distribution priorities were availability, price and functionality, in that order. This was the opposite of the priorities of corporate customers.

A radical cultural adjustment was needed for the retail products to succeed. The solution had HP move the printer division operation to a green-fields location in Boise Idaho. There they built a new culture more tuned to the new business realities. In this regard, HP's experience with inkjet printers mirrored IBM's experience with the early PCs. To establish an agile and responsive retail cultural, IBM also was compelled to move their emerging PC operations to a green-fields location near Dallas Texas. Today, the "HP Way" includes a culture that resembles an Open Source program.

The HP-UX operating system and the HP Openview platform businesses have been providing value to customers for over a dozen years. HP-UX is a variant of the UNIX operating system and is used on most of the larger HP computers. Openview is a systems and network management platform for data center management. The platforms earn direct revenue from end item sales as well as integrate into other HP products and services.

A re-use eco-system provides economic support to both programs. A worldwide third party development community produces and consumes platform extensions and variants. Over 1,000 partners create and sell Openview applications. The platforms are sold and supported like any other HP product. The profit margins sustain and extend the eco-system. The platforms' corporate footprint encourages enhancement efforts by third parties. Functionality from acquisitions like Peregrine, Mercury Interactive and Opsware accelerated the breath of Openview's attractiveness to third parties.

Among the processes that promote discovery of new opportunities, long-range roadmap reviews are held with developers and partners. Yearly events, such as the Openview Forum, and HP-UX user groups, showcase the roadmap and provide third party feedback. Regional updates take place at trade shows and other forums.

Alternative practices

Discovery - To foster inter-group discovery, Microsoft's CEOs hosts a twice a year summit with the top 100 business managers. Meetings take place around a large conference table with one seat occupied by the business unit leader and seats behind the leader (away from the table) occupied by the business unit's CTO and one or two other key staff. The meetings include a presentation by each business unit exposing a 3-year forward, and one-year back operational view, and the challenges they expect to face in meeting future objectives. The CEOs present common thrusts and issues. The meeting exposes participants to efforts underway across the organization, opportunities for shared efforts and potential duplications of effort.

Reuse at Oracle has been stymied by aggressive acquisitions and marketing commitments to maintain and extend overlapping product sets. Reuse at Oracle is more an event than a managed strategy. Top down directives from the CEO / CTO are frequently the result of a business situation. Incentives are sometimes used to drive the initiative. The study found similar approaches at numerous other technology companies.

Conclusions

Corporate wide governance practices and policies are mandated by regulations like Sarbanes Oxley with oversight by organizations like the SEC and compliance is a matter of law. The biggest challenge

facing organizations is in how to define operational controls implied by such legislation. Guidelines for Corporate governance at the board level is well understood but operational governance and risk management practices vary on many dimensions in each working environment.

Product development operational governance revolves around project lifecycle management, quality management, and portfolio management. Commonly encountered life-cycle models are Stage Gate processes, waterfall development practices, total quality management, six-sigma, ISO-9000 etc. The governance can be loosely coupled across the enterprise, each division exercising its freedom to operate as it chooses.

Effective technology sharing governance practices tend to be tightly coupled across the enterprise. This is especially the case with the incentives, funding and discovery processes. However, opportunistic, ad hoc cooperation between business units frequently yields successful results as well.[iv] Among enterprises, there is wide range of styles. The practices reflect the culture, history, leadership, individualism, power structure, etc. of the company. Some companies view their distinctive practices as a competitive advantage.

There are major challenges to software reuse. Few lie with the front line developers. Developers already reuse or repurpose code (most often their own). They do not have a "not invented here" bias except when the consequences of failure (imposed by management) overwhelms their reticence to recreate something that already exists. The barriers to software reuse are not technical, though there are many technical challenges to work through. Software technology is advancing at unprecedented rates, and many of these advances make software more modular and reusable.

The primary business challenge is solving the discovery and incentive problem. Developers cannot reuse software they do not know about. A project's reuse opportunity window is narrow. Software reuse components must be considered very early in the development cycle, or the opportunity window will have passed.

Proper incentives can overcome skeptical concerns that software reuse is either risky or another management "program de jour". The most effective incentive systems set aside funds that spanned multiple fiscal years. Depending on the model, incentives may be needed for both producers and consumers.

Management impatience / intolerance to failure is another challenge. Creating a self-sustaining reuse process takes multiple cycles of learning, adjustments and time. Not every project attempt will be successful. There are numerous complexities in selecting the best reuse opportunities and correct incentives. This is especially the case when beginning.

Leadership matters--someone must be able to drive the vision for leveraging commonality and technology sharing and be accountable for delivering the results. Identifying a "technology reuse czar", or chartering and funding a reuse committee or "commonality board" are typical approaches. They are responsible for allocating the incentive funds, tracking actual reuse activity and calculating the delivered value.

Governance Case Study

Cisco Systems – MPLS/RSVP Development

Cisco is well known for its aggressive acquisition strategy. When the expertise for major, new, shared functionality arrived through acquisition, the project's size and demands overwhelmed the small number of experts. The situation forced changes to the standing governance rules.

In the late 1990s, Cisco Systems Inc. commissioned its central engineering groups to develop a new class of routing protocols broadly known as Multi-Protocol Label Switching (MPLS) and the accompanying Reservation Protocol (RSVP). MPLS/RSVP was an emerging inter-networking standard for reserving network bandwidth and providing better end-to-end performance across IP networks. The development was to serve all Cisco business units (BUs) and would become part of Cisco's core platforms. Many Cisco BUs were demanding this capability.

The technical expertise on MPLS/RSVP primarily originated at a company that Cisco recently acquired. The engineers were assigned to the central engineering team to work on the project. As the development proceeded, it ran into governance difficulties as it approached the point of integration into the four Cisco's platforms for which the central engineering team had responsibility.

- MPLS was an unusually large and complex development, and skilled engineers were in short supply
- It had to integrate, in parallel, into four different platforms (IOS, IOX, DCOS and Linux).
- The standard was still evolving. There were follow-on MPLS standards that had to be developed
- There were customer specific special adaptations that would be required before the functionality was deployable.
- Teaching the BU's engineers how to connect their software to the new MPLS platform functionality required considerable effort.

BU-specific engineering development is done within the BU's operating funds. Functionalities developed by corporate-funded central engineering teams are imported as common components and libraries. A Cisco Technology Council establishes the priorities for central developments. The Technology Council is a corporate consortium of technologists, marketers and business leaders who also control a pool of funds applied to opportunistic shared technology development. Cisco's Technology Council had contributed funds to hire additional engineers for the platform MPLS / RSVP development effort. Now the question was whether they would fund any of the costs needed to make MPLS deployable?

It was expected that there would be inter-BU disputes regarding feature sequencing and availability across the four platforms. What was not anticipated was the level of customer-specific customization. Historically, this was the BU's responsibility. A related problem was how to validate numerous requests for customization coming from the BUs. Without skin in the game, the BUs were not motivated to filter the incoming customization requests

New governance policy had to be developed to account for the desired feature development velocity and the funding model for customizations. The new rules required the BUs to compensate the central group for requested customizations. All changes had to be provided by the central development teams. The chief development officer (CDO) arbitrated disputes between the parties (or the Technology Council) over what constituted a customer special versus a needed standard feature. Disputes over the expenses (dollars or time) to execute a customization were also settled by the CDO.

The primary benefits of this model were 1) it forces the BUs to be financially prudent regarding which customizations to undertake and 2) it got the Technology Council back to using its incentive budgets for opportunities, not core requirements.

Summary Comparison Chart

	Cisco	Motorola	Microsoft	Apache Software Foundation	HP Software Division
Reuse Sharing Model	Centralized	OEM	Library	Open Source	Open Source-like
Reuse Discovery	Monthly Technology Council Meetings	Yearly roadmap reviews	6 Month Summits	Public communications	Eco-system roadmap reviews
Reuse Incentives / Funding	"Tax" to fund central group Sustaining and customizations fees	BU self-funded + Revenue allocation	Self sustaining eco-system	Altruism of volunteers Self-interest of program team	Self-sustaining eco-system
Reuse Controls	Disputes arbitrated by CDO or Tech Council	Inter-BU agreements	Common practices from Technology Councils	Incubator and program centric decisions	Common practices from technology councils
Reuse Sustenance	Yearly cost budget	BUs fund their interests	External sales and P&L	Critical mass community in place	External sales and eco-system sponsorship

.

[i] Griss, M, L; Software Reuse From Library to Factory; IBM Systems Journal; December 1993

[ii] Cusumano, Michael, A., Systematic Versus Accidental Reuse in Japanese Software Factories; Sloan School of Management, MIT Cambridge, Massachusetts; WP#3328-BPS-91

[iii] Fisher, Lawrence M; How HP Runs its Printer Division; Strategybusiness.com/press/16635507/12217?tid=230&pg=all

[iv] Fichman, Robert G., Kemerer, Chris F., Incentive Compatibility and Systematic Software Reuse; Journal of Systems and Software, July 2000.