Incentives versus targets - a practical experience

Michael Wasmund
IBM Deutschland GmbH
Schoenaicher Str. 220
W–71032 Boeblingen
Germany
Tel: (011)–49–7031–16–2273
Fax: (011)–49–7031–16–3545.
E–mail: wasmund@boevm4.vnet.ibm.com

Abstract

Papers recently published about Tech transfer issues in the area of reuse agree upon the following: The major obstacles in realizing the expected benefits from reuse are rather related to human behavior than to technical barriers. A very frequently discussed means to stimulate desired behavior are incentives: Major companies established incentive programs to stimulate reuse and consider this helpful particular in the beginning stage. However, massive progress solely through incentive programs is not reported. An alternative mode of stimulation is the establishment of quantitative targets for reuse. Opinions differ on the usefulness of mandatory versus voluntarily stimulations.

At IBM's system software development site in Boeblingen, Germany, we had the opportunity to test both approaches. This position paper depicts experiences with either approach. The result is that mandating reuse targets significantly raised the level of practiced reuse in general, but also generated some unwanted minor side effects.

Keywords: Reuse, Incentives, Experience Report, Software Reuse, IBM.

Workshop Goals: Share experience about process modification and implementation. Learn how object-oriented techniques and reuse merge.

Working Groups: Tech Transfer issues, Reuse & OOT.

1 Background

Papers recently published about Tech transfer issues [1, 2] in the area of reuse agree upon the following: The major obstacles in realizing the expected benefits from reuse are rather related to human behavior than to technical barriers. A very frequently discussed means to stimulate desired behavior are incentives [3, 4]. Major companies established incentive programs [5, 6] and consider this helpful particular in the beginning stage of tech transfer. However, no one reports that incentives have caused an avalanche effect, i.e., massive and broad change in behavior of addressed people.

Other reuse experts [7] recommend a more mandatory enforcement, i.e., establishment and monitoring of quantitative targets for reuse by site management. Opinions differ on the usefulness of imposing a desired behavior through targets in contrast to stimulation of voluntarily moves.

At IBM's system software development site in Boeblingen, Germany, we had the opportunity to test both approaches. This position paper depicts experiences with either approach. The result is that mandating reuse targets significantly raised the level of practiced reuse in general, but also some unwanted minor side effects.

2 Position

Prior to application of any methods to change behavior, we found it essential to recognize inhibitors: The following technical inhibitors are most often mentioned by tech transfer agents:

- Lack of reusable components
- Lack of component compatibility
- Lack of appropriate development environment
- Lack of appropriate database retrieval mechanisms.

Motivating people to practice methods which are not technically supported would render all tech transfer efforts meaningless. Therefore, technical inhibitors must be removed or at least smoothened before addressing the following often cited non-technical inhibitors:

- Lack of management commitment
- Lack of long-term product strategy
- Not-invented-here syndrome (NIH)
- Lack of education in software engineering principles

Practitioners and tech transfer agents agree that 'lack of management commitment' and 'Not-invented-here' are the major hurdles.

2.1 Incentives

At IBM, all major development sites applied incentive programs. These programs are cheap compared to otherwise redundant software development and reward desired behavior, particular removing the NIH inhibitor.

There are basically two classes of incentive programs: Point-based and not point-based. The point-based programs are set up similar to a Frequent Flyer Program: The more 'reuse' an individual practices, the more points are credited. A certain amount of accumulated points qualifies for a (mostly monetary) award.

'Practicing Reuse' means either using available components for product development or it means production of reusable components itself. The latter is in most programs linked to the condition that the produced reusable component is actually reused at least once in order to justify the effort.

The other class of incentive programs (non-point based) grants monetary or other awards to selected individuals or teams in order to show and reward successful examples of the desired mode of work. The criteria for this class of incentives are rather qualitative (e.g., innovative approach) than quantitative (cost savings, quality gains).

IBM Lab Boeblingen chose a non-point based incentive program. The rationale for this decision was:

1. To avoid the overhead of calculating and managing accounts of individual's points,
2. To avoid frictions between comparable individuals in assessing the correctne ss of accumulated points,
3. To ease the opportunity to grant team awards as alternative to team awards.

2.2 Impact of the chosen incentive program

An instance of the non-point based class of incentive program has been established in spring 1992 at our lab. The program was announced by a personal letter of the area manager to all employees. In the course of 1992, three applications for reuse awards were submitted, one thereof was actually granted.

Analysis of the slim acceptance of the program showed:

1. The extremely tight schedule of product developments did not encourage additional investments in making software reusable. Even if someone did invest his private time in doing that, he would be eligible for a reuse award only if a second user of his software could be found.
2. On the other hand, integrating reusable components into new products should help in meeting tight deadlines by prevention of redundant coding, particular in the area of general-purpose routines.

However, there are some risks involved in integrating reusable components from other sources:

- Time must be spent for searching the matching component, without knowing whether a component will be found,

- If a component is found, integration of the obtained component can yield unsatisfactorily results not visible from the description accompanying the component: Unknown bugs, interface binding problems, runtime problems, performance.
- Dissemination of object-oriented methods facilitating reuse was just at the beginning. Very few people could enjoy an object- oriented environment including class libraries.

Only few developers were willing to trade in these risks for the chance of saving considerable coding - and maintenance - effort. Most individuals would follow known and always practiced methods.

In summary, the impact of the incentive program was minor. It did not change adherence to traditional development methods to a great extent.

2.3 Reuse Targets

Realizing the slow progress, management decided in January 1993 to mandate reuse by establishment of a formal reuse target of 20the organization. The targets were propagated down the hierarchy and even included in some individual's performance plans.

The effect was an immediate jump in requests for reusable parts. Fortunately, most requests could be satisfied, particularly by our own unique parts center [8] providing general purpose components for abstract data types and class libraries. The fact that now there was no choice for the professional but to leave traditional paths in order to support the reuse target, and the support of the management team, which also had the objective to utilize new software development methods, literally changed the world at Boeblingen Lab. The ratio of reuse related activities 1992 to 1993 is close to 1:4.

2.4 Side Effects

The establishment of targets had generally a very positive effect. However, we obeyed the following side effects:

- Some organizational units overdid the setting of targets by prescribing each individuals quantitative contribution to the reuse figures. The opportunity to reuse available components greatly depends on the technical contents of the work an individual has to do, so mismatches between realistic reuse opportunities and set targets occurred. In addition, the partition of the technical project content in pieces can lead to very heterogeneous reuse results within a group. If each group member has personal reuse targets, frictions can occur. In summary, I do not recommend to define reuse targets more granular than on 'group' or 'project' level.
- Secondly, bad surprises caused by unprepared application of complex reusable components came up at some places. Some groups intensively embarked on practicing reuse immediately after announcement of the targets without having had appropriate training or other preparation. This lead to severe problems during the coding phase, because the design originally did not reflect that large amount of reused software. It turned out, that the first application of reuse took considerably more time than expected; heavy support by the originator of reusable components was needed.

  Learning from this, setting of reuse targets must reflect education effort in order to effectively apply new methods rather than to 'jump into'.

- Another side-effect was caused by flat undifferentiated application of the same reuse target across all departments regardless of the department's actual task. There was a sound definition established for reuse of *software* [9]. However, other operations such as Information Development and Test, had problems to apply reuse targets on their business, without support of established measurements. Subsequently, there was great confusion, what the target means and how achievements of different operations contribute to the reuse results.

- Last but not least, some individuals tried to enhance their personal reuse result by integration of extra large reusable components, whose whole function set was not really needed.

Despite these non-neglectible side-effects, the comparison between 1992 and 1993 shows a crystal clear preference for introduction of new technologies through establishment of sound quantitative targets.

3 Comparision

It is the objective of this position paper to convey the lessons learned. The definite preference statement made above applies to our specific organization. Other organizations may react different to the same set of motivators. The history, composition, and mission of an organization heavily determines its reaction to different tech transfer approaches. Our organization is characterized by a major amount of legacy code maintenance and enhancements, a major amount of assembler code, and some progressive projects using object-oriented technologies. The average age of our professionals is 40+ years, which also makes a difference. I think, these are important attributes to know in order to assess the applicability of the position made in this paper.

References

[1] R. Joos, "So Much for Motherhood, Apple Pie and Reuse," in *5th Int'l Workshop on SW Reuse*, (Palo Alto, California), 26-29 October 1992.

[2] M. Wasmund, "Critical Success Factors of Reuse at IBM Boeblingen, Germany," in *5th Int'l Workshop on SW Reuse*, (Palo Alto, California), 26-29 October 1992.

[3] R. Fairley, S. L. Pfleeger, T. Bollinger, A. Davis, A. J. Incorvaia, and B. Springsteen, "Final Report: Incentives for Reuse of ADA Components," Tech. Rep. Vols. 1 through 5, George Mason University, Fairfax, Va., 1989.

[4] H. B. Carstensen, "A Real Example of Reusing ADA Software," in *Conference on Software Reusability and Maintainability*, (The National Institute for Software Quality and Productivity, Inc., Tysons Corner, Va.), 1987.

[5] M. J. Cavaliere, "Reusable Code at the Hartford Insurance Group," in *Workshop on Reusability in Programming*, (Newport, R.I.), 1983.

[6] J.W.Hooper, *Software Reuse Guidelines and Methods*. Plenum Press, New York, London, 1991.

[7] J. R. Tirso, "Establishing a Software Reuse Support Structure," in *Proceedings of the IEEE International Conference on Communications*, June 1991.

[8] Lenz, Manfred, H. A. Schmid, and P. F. Wolf, "Software Reuse Through Building Blocks," in *IEEE Software*, July 1987.

[9] J. Poulin, "Issues in the Development and Application of Reuse Metrics in a Corporate Environment," in *Fifth International Conference on Software Engineering and Knowledge Engineering*, (San Francisco, CA), pp. 258–262, 16-18 June 1993.

4 Biography

Michael Wasmund *IBM Germany,Dept. 3238, Schoenaicher Str. 220, W–71032 Boeblingen,* is Site Reuse Champion at IBM's system software laboratory in Boeblingen., focusing on implementation of a reuse methodology throughout product development. Previously, he was research staff member at IBM's European Networking Center, Heidelberg, Germany, where he published several papers about networking in heterogeneous environments. At the beginning of his IBM career, he developed IEEE 802.3 adapters for S/370 series. He obtained his B.S.E.E. and M.S.E.E. degrees in Ulm and Bremen (Germany), respectively.