An Experiment With Financial Incentives For A Small Software Development Team

Michael J. Glasgow
M. Susan Murphy

IBM Federal Systems Company
9231 Corporate Blvd.
Rockville, Maryland 20850

Introduction

By any standard, the Advanced Automation System (AAS) program is a large undertaking. Over a period of some 15 years, the nation's air traffic control system will be completely overhauled: more than 10,000 networked computers and approximately 2.5 million lines of code (mostly Ada) will be installed in over 300 Federal Aviation Administration (FAA) facilities. This overhaul is necessary to increase the capacity and efficiency of the system to support projected air traffic volume well into the 21st century. The current system is built largely on aging hardware from the 60's and early 70's, further emphasizing the need to modernize.

There are several challenging aspects of the AAS. The application is complex. Fault tolerance is imperative: a critical subset of the system cannot be down (which includes missing response time requirements) more than 3 seconds per year. Performance requirements are stringent: radar data received at the system boundary must be displayed for controllers in less than 1 second.

Perhaps the most challenging aspect of the AAS is its size. An organization of almost 1500 people supports the design, implementation, and testing of the AAS, including over 400 software developers.

It is in the context of this large organization and complex application that an experiment was performed to solve a difficult technical problem on an aggressive schedule using a small development team and financial incentives.

The Problem: The problem centered on a software application referred to as the Types Dictionary Services (TDS). TDS provides the ability to create, assign values to, and interrogate Ada objects without visibility to the corresponding Ada type. It is used extensively on AAS in both real-time and off-line support applications.

Although the original TDS implementation was exceptional in terms of its functional capabilities, it suffered from three significant deficiencies. First, its execution performance was three orders of magnitude too slow for use in certain AAS real-time applications. Second, it introduced a significant size problem in applications that used it (several megabytes of object code for the dictionaries). Third, the off-line process for generating the dictionaries was very time-consuming and error prone, and frequently resulted in long periods of programmer down-time.

TDS had to be redesigned. To do it right would require involvement of several different parts of the organization. And because TDS was severely distorting the real performance of the system, to such an extent that other potential performance problems could not be effectively analyzed, it had to be done quickly.

The Experiment: At the time the need for a TDS replacement became apparent, there was strong interest on the part of AAS management in experimenting with innovative forms of motivation and approaches for accomplishing work.

An approach that had been considered by management was modelled after the award or incentive fee type of contract. The opportunity to reap additional profit for achieving well-defined milestones has proven effective in motivating contractors. Perhaps a similar approach would be successful when asking individual employees to undertake a particularly difficult task.

Usually award recognition is after the fact and employees have no expectation of the additional compensation. The new concept would recognize the challenge of the task ahead of time and commit to a specific monetary award if the task was successfully accomplished.

Six developers were asked if they would like to participate in an experiment based on this approach. All six accepted and the team was formed. Collectively, the team represented experience in all facets of TDS. Members included:

- The current owner/maintainer of the service
- A person knowledgeable of the Rational Environment® (where the bulk of the dictionary generation process is implemented),
- A representative of the Builds and Controls department responsible for generating dictionaries on an ongoing basis
- A representative of a real-time client
- Two representatives of an off-line client

Management's goal was that all members of the team be dedicated full-time to the task. However on a practical basis, the skills needed for the TDS team conflicted with other commitments and two of the members could participate only half-time.

A challenging set of success criteria was established including an end date that allowed only 2 months for completion. All team members, including those working half-time, were to receive the *same* amount of money if all success criteria were met. The intent was to foster teamwork. If any of the criteria were not met, none of the award fee would be paid (no graduated scale). It was generally accepted that there was only a 50% or less chance of success.

A separate group of reviewers was appointed to determine if the success criteria were met. The reviewers included one manager and two senior-level technical staff.

Although no formal announcement was made, it was widely widely known among other software developers that the group would receive additional compensation upon successful completion of the task.

The team was given essentially no guidance as to how the TDS replacement effort was to proceed. No lead was appointed by management nor were any a priori assignments given to the team members. The choice of development environment was left to the team. Work hours could be set as desired. Management solicited and responded promptly to requests from the group for necessary items such as development hardware and lab space.

Results: The team met the success criteria and demonstrated an unusually high level of productivity in the process. Well over 10,000 lines of new Ada code were developed and tested in the 2 month period. The new TDS yielded dramatic performance and size improvements. The off-line process for generating dictionaries was much improved.

Also of particular interest were the *human aspects* of the experiment: the influence of working in the empowered small team environment, the interpersonal dynamics within the team, the interactions between the team and management, and the influence of financial incentives both on the team members and other software development personnel.

After a brief overview of the TDS replacement, the balance of this paper addresses some of the factors contributing to the high productivity of the team and relates experiences, both positive and negative, relative to the financial incentives. A summary of the lessons learned and what would be done differently in the future is provided in the conclusion.

TDS Overview

An overview of the TDS process is provided in Figure 1. The primary intent is to illustrate the scope and complexity of the TDS process.

The process is summarized as follows: (1) Type packages containing types that need to be registered with the TDS are developed by programmers throughout the software organization in accordance with certain restrictions imposed by the TDS. (2) All the type packages are transferred to a Rational processor and compiled. (3) A tool uses the DIANA representation on the Rational to produce portable Ada code referred to as *descriptor code* (4). When compiled and executed on a particular target processor, the descriptor code produces *descriptor data* (5) which describes the layout of the associated types (e.g., field names, bounds, bit offsets, etc.). This data is written to a set of files commonly referred to as *dictionaries* The dictionaries are read by the actual TDS (6) which is then ready to service requests from various client applications (7).

As indicated in the figure, the process involves and/or impacts essentially everyone in the software development organization. Consequently, inputs from the

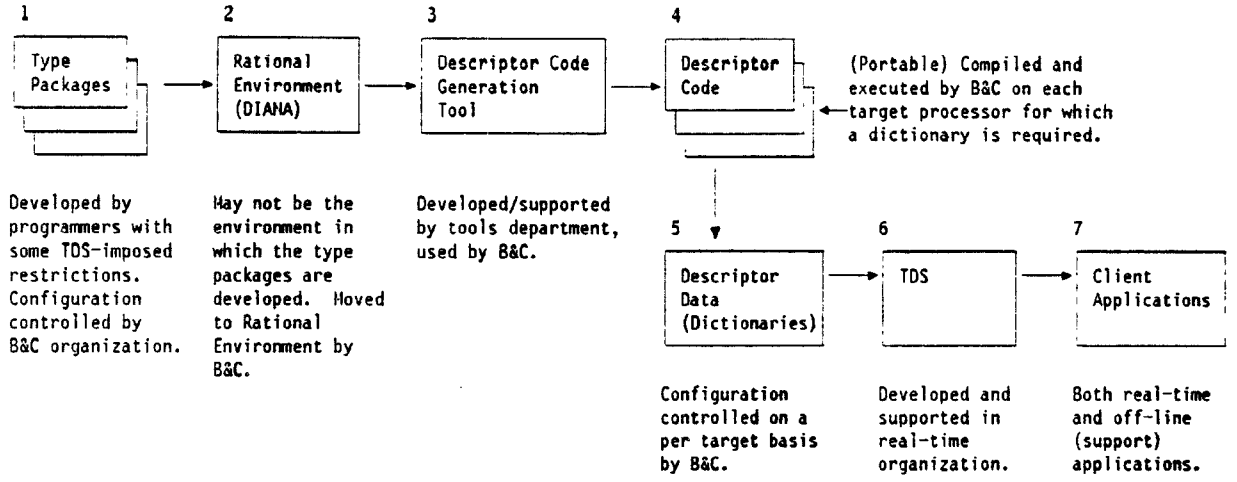

Figure 1. TDS Overview

entire organization had to be factored into the implementation.

A complete description of the TDS including examples of TDS client applications as well as the details of descriptor code and the descriptor data structures is provided in [1].

Human Aspects of the Experiment

There are three areas of interest concerning the financial incentives: 1) reactions by the team and non-team members (primarily those in the software development organization), 2) their effect on interactions involving team members, management, and other software developers, and 3) their efficacy as a motivating influence on the team members. Positive and negative results were seen in all three.

Reactions: Reactions to the financial incentives were mixed, but generally more negative than positive. They included:

1. The incentive introduced a feeling of greed among some team members that made them uncomfortable. It is more satisfying to feel motivated purely by the challenge of the technical work.
2. Team members felt awkward asking others outside the group for help. They sensed a reaction of "why should I help you when you're going to get extra money for this?". What would have been normal everyday cooperation became strained.
3. Some team members felt like they *shouldn't* ask other non-team members for help (even if help would willingly have been given) *because* they were receiving additional compensation.
4. Some non-team members reacted negatively because they weren't chosen for the effort.
5. Many felt that it could not have been determined up front that the effort required to solve the TDS problem merited additional compensation. It would have been better to have the team fix TDS and then determine if awards were appropriate.
6. Both team and non-team members concluded that it was unrealistic to expect the relative contributions of each team member to be equal, so a determination at the outset that all would receive equal compensation was inappropriate.
7. A few non-team members observed that employees should not be given additional compensation for, in essence, doing their jobs.

It is inevitable that some objections will be raised whenever any group or individual is given a particularly desirable assignment and/or an award. (Indeed, the appropriateness of any form of award is debated throughout industry.) It is also a hard truth that not everyone who would like to participate in a small team effort is needed or qualified. A management team must be prepared to deal with these concerns if the incentivized team approach is to be used.

It is true that management could not have known *exactly* how difficult a task the team faced, but there was ample evidence to suggest that a substantial challenge had been given: 2 months time to develop, test, and deliver several thousand lines of code and to construct a process that affected and had to be accepted by most of the software development community. From the standpoint that TDS was costing the project significant sums of money in lost development time, the amount set aside for the incentivized team would be well-spent if the team were successful. However, this is a concern that should be taken very seriously by management. Success criteria must be sufficiently stringent to justify additional compensation.

But, even with strong evidence that a task will merit some amount of additional compensation, it is difficult to assess how much, and the relative contributions of each participant will vary (particularly if some participants are not full time). This is an area that would be handled differently if the experiment were to be repeated. Instead of a fixed amount for each participant, a *range* of possible awards could be presented up front (e.g., $500-$2500). Management would retain the prerogative to determine the final amount, and it could vary by participant. (Team input into who should receive how much could be factored into the management decision.)

Interactions: The financial incentives had an overall negative effect on the interactions of all parties concerned. Some of the specific problems that were encountered include:

1. As alluded to above, interactions between the team and non-team members became strained due to awkwardness felt on the part of the team members or resentment on the part of non-team members.

This problem suggests that unless incentives of this type are being offered on a frequent basis and not to the same set of people repeatedly, they can have a detrimental effect on working relationships within the organization. This might not occur if everyone believes they will eventually get a turn.

2. Some team members felt that the incentives resulted in less than typical encouragement and support from management.

This was certainly unintentional, but there may have been a tendency to think that the incentives alone were sufficient encouragement. Other forms of encouragement, varying by the needs of the individuals, are still very important.

3. There was a significant amount of tension within the group because it was felt that the relative contributions of each team member did not justify equal incentives. This was due primarily to the part-time participation of two team members.

It is crucial to understand that the problems that arose from the involvement of part-time participants were not the fault of those participants, but rather inherent from the inequity of part-time participants receiving equal compensation to full-time participants. This was perhaps the hardest lesson learned from the experiment and is another area where future experiments should be conducted differently. Either part-time participants should be avoided altogether (most desirable) or compensation should reflect the percentage of time spent on the team task. To do otherwise is unfair to both the team and the individual, and is bound to result in unnecessary friction.

4. The incentives restrict the flexibility of management to alter the make-up of the team. Once an individual has been told they will receive a specific amount of money, a reassignment is not just a reassignment, it's the loss of the money.

This problem also became apparent as a result of the part-time participation. When it became obvious that the part-time participation was causing undue tension within the group, a change probably would have been made had the incentives not already been offered.

Motivation: Without question, the financial incentive played a significant role in motivating the team. This was particularly apparent as the team approached the completion dead-line. The amount of over-time worked during the final week was enormous, and there is little doubt that this was due, at least in part, to a desire not to lose the financial incentive.

The decision to take an all or nothing approach with the incentive was also a contributing factor: individuals did not want to be responsible for causing the team to lose the incentive.

But to the extent that the financial incentive generated tension within the team or between the team members and non-team members, it was more of a hindrance and a distraction than a motivating factor.

It is worth noting that *team membership* was cited as an equal or even more significant motivator by many of the participants. Most, if not all, of the team members indicated that they would have wanted to participate even if financial incentives had not been involved because of the opportunity to work in the

small, focused team environment to solve a challenging problem crucial to the success of the project.

Team Considerations

Team Leadership: The decision by management not to designate a team leader was consistent with the objective of empowerment. If the team felt a designated leader was necessary, *the team* could make that decision. The team decided not to designate a leader, and the resulting dynamics proved to be difficult, particularly when technical debates needed to be arbitrated or when there was disagreement over who should take on certain assignments.

An effective team leader will greatly enhance a team's likelihood of success. Of course a good leader will adapt his/her style of leadership according to the personalities, maturity levels, and technical abilities of the team members - leadership of five college new-hires will be markedly different from leadership of five highly competent 15-year veterans. But somebody within the group must have final decision authority on all technical issues and work assignments.

Team Construction: It has often been observed that if you want to get a job done, get the right people. There is certainly some validity in this exhortation, but, in practice, it's not that simple.

Team membership was largely determined by management. Managers from various areas of software were asked to identify the right person, and each dutifully selected competent individuals in whom they had confidence to complete the task successfully. But a collection of the right people from various areas do not necessarily come together to form an effective team. While all the team members knew each other prior to the experiment, only a few pairings represented well-established working relationships. Not surprisingly, this pot-luck approach to assembling the team resulted in some difficulties due to personality conflicts and significantly differing technical views. It is common for a team to require some period of time to gel and form effective working relationships. However, the schedule for the task did not include this time.

An alternative approach to assembling the team that might work better consists of two parts. First, rather than asking managers from various affected areas to identify who they think should participate, identify a team leader in whom complete confidence rests to solve the problem. Then solicit his/her inputs concerning the necessary number of participants and who they should be. If three are needed, require ten people to be identified. Then management can select from the ten. The benefit of this approach is two-fold: the team will be kept to only the size needed to do the job and the team members will be much more likely to work well together. (By selecting different leaders, participation in this kind of activity will naturally be spread across an organization.)

Second, realize that some adjustments in team membership may be required and insure that these adjustments can be made. As was stated earlier, the commitment of monetary incentives makes this extremely difficult.

Desirable characteristics of team members are mostly obvious: technical competence, ability to work with others, commitment to the task, and dependability to name a few. A characteristic that is perhaps not so obvious, but crucial in at least the case of the team leader, is the ability to *see the big picture.*

Not surprisingly, most team members tended to migrate toward and work on the aspect of TDS that most affected the area of the project they represented. This fostered a tendency to evaluate technical issues from a "what's best for my area" perspective rather than a "what's best for the project/system" perspective. While a certain amount of this form of parochialism is healthy, it can lead to the best interests of the project not being served. An effective team leader who possesses the ability to *see the big picture* will prevent this from being a problem.

Productivity Boosts

Several factors contributed to the high productivity of the team, including a commitment on the part of management to prevent distractions, reduced formality in the inspection process, and freedom to work odd hours.

Preventing Distractions: The team would not have been successful had management not been committed to preventing distractions. This was accomplished in two forms: unrelated assignments were avoided during the experiment (at least for the full-time participants) and dedicated office/lab space and equipment were provided.

The importance of avoiding unrelated assignments cannot be overstated. Most readers have probably observed situations where a 'tiger-team' was formed to address some critical problem, but failed because the participants had too many other responsibilities.

Since the participants in a small team are likely to be important contributors to their normal work area, management has to be prepared to accept some difficulties in those areas while the team is in operation.

Having dedicated office/lab space and equipment was viewed by some of the team members as the most important productivity factor. It provided a place to "hide" and avoid the normal interruptions of phone calls and visitors typical in one's office. Further it provided an on-demand conference location big enough for all team members. Dedicated hardware insured that time wasn't lost waiting in queues or competing for CPU resources with other developers.

(It is interesting to note that despite enthusiasm for the dedicated office/lab space as a hiding place, most of the team members said in interviews after the experiment that much of their creative work was done at home - to avoid the many interruptions at work.)

Reduced Inspection Process Formality: The time available to the team necessitated some modification to normal procedures. The inspection process in particular comes with a relatively high overhead. Inspections must be scheduled such that several individuals from different organizations can participate. Inspection packages consisting of source listings, test plans, and other materials must be prepared and distributed a few days before the inspection. The inspection results (number and types of errors) must be recorded.

There were two major pieces of software the team had to develop: the actual types dictionary service used by client applications and a tool that runs on the Rational to produce descriptor code (see Figure 1). The service was developed primarily by two members of the team, the tool by one. (The main reason why the tool was developed by only one person was the lack of Rational Environment knowledge by other team members.)

In the case of the service, no *formal* inspections were held. The code was inspected informally by the two developers as they progressed, each developer reviewing the other's code. The service, consisting of over 8,000 lines of code, has experienced a very low latent error rate.

In the case of the tool, the code was formally inspected, but the inspectors consisted primarily of other members of the tools department who were not team members. Consequently, they were at somewhat of a disadvantage from not having worked side by side with the principal author as the implementation progressed - they had to absorb a large amount of background information (the rationale behind the design of the entire TDS process) and review a large amount of code in a short period of time. The tool, consisting of over 5,000 lines of code, has experienced a higher latent error rate.

To be fair, the implementation of the service was an easier problem. The interface to the clients is finite and well-defined, as are the data structures that describe types. The tool, on the other hand, has to deal with an infinite variety of inputs (the types defined by developers) and there are many subtleties.

Nonetheless, the experience of the team suggests that the critical elements of the inspection process can be implemented informally without the high overhead of the project practices. It further suggests that programming partners and more frequent, smaller inspections contribute to increased productivity and uncompromised (perhaps increased) quality.

Hours: Though it may seem like a minor thing, freedom to adjust work hours as needed was quite helpful. It provided another way to avoid interruptions from phone calls and visitors, and, when use of non-dedicated hardware was required (e.g., testing in the S/370® main-frame environment), allowed the team to avoid peak load times.

Although the productivity benefit is debatable and difficult to quantify, team members also were appreciative of a relaxation of the unspoken dress code. A dress-down mode was adopted by most of the team for most of the experiment.

Conclusions

The TDS experiment demonstrated that the small team approach is very valuable. Given the right people, difficult problems can be solved with very high productivity. The small team environment also serves as a good learning experience for the participants - they are exposed to areas of the project they normally wouldn't see, and they go through the whole process of developing software in a relatively short period of time.

The experiment also demonstrated that monetary incentives can be a distraction instead of a motivator. Ironically, it can be argued that the TDS team succeeded *in spite of* the monetary incentives involved.

Based on the experiences of the team, the following recommendations were made:

1. Increase the use of small teams to tackle significant project challenges. Participation alone is a strong motivator.
2. Establish a team leader and consider his/her input in selecting other team members.
3. Keep the team as small as possible.
4. Require full-time participation for all team members.
5. Take steps necessary to eliminate team distractions.
6. Assign a management focal point and remember that normal forms of encouragement are still very important.
7. When the challenge is suitable for monetary recognition, consider identifying in advance a range of awards, but retain management judgement/prerogative in determining the final amount for each contributor.

References

[1] Glasgow, M.J., Hepner, D.L., and Schmidt, R.B., "Implementing a Table-Driven Types Dictionary Service in Ada", *Proceedings of the Second EUROSPACE Ada in AEROSPACE Symposium*, January, 1992, pp 676-690.

Trademarks