Motivation, Teamwork, and Agile Development

Giovanni Asproni

aspro@acm.org

http://www.giovanniasproni.com

Introduction

Motivation as defined by the Merriam-Webster dictionary 11th edition is "1 a : the act or process of motivating b : the condition of being motivated 2 : a motivating force, stimulus, or influence : INCENTIVE, DRIVE".

The fact that motivation is the most important factor for productivity and quality is not a new discovery. It has been pointed out for the first time by the studies conducted Elton Mayo around 1930. Since then there have been several studies that confirmed the same results in several industries including the software development one [2], [4], [10], [14].

Nevertheless, until recently the main focus has been on *process-centric* methodologies, the ones that Jim Highsmith calls Rigorous Software Methodologies (RSM) [5].

The basic assumption behind RSMs is the same behind scientific management—that is, to improve productivity and quality, it is necessary and sufficient to improve and formalize the activities and tasks of the development process. In this kind of methodologies people have to adapt to the process.

The advent of the Extreme Programming first and the Agile Movement later, has put people back at the center of the development activities. In these kind of methodologies "people trump process" [3], the processes have to be adapted to the needs of the people involved.

According to Jim Highsmith agile development methods appeal to developers because they reflect how software really gets developed [5]. In this article I claim that agile methods also appeal to developers because they reflect how they really *like* to develop software.

Since nowadays, most software is developed by teams, I have taken the approach of showing the strong connections between motivation and effective teamwork, and then showing how agile development methods are related to the latter.

Motivation

The classic experiments that demonstrated the influence of motivation on productivity were conducted by Elton Mayo from 1924 to 1932 at the Hawthorne Works of the Western Electric Company in Chicago [8]. The results are known under the name of "Hawthorne Effect."

The original purpose of the experiments was to find the effects of illumination on productivity.

The results were quite surprising:

- When illumination was increased, productivity went up
- When illumination was decreased, productivity went up
- When illumination was held constant, productivity went up

After these results, Mayo and his associates began to wander what kind of changes in the work environment could influence productivity.

They set up an experimental group taking six women from the relay assembly line. The group was isolated from the rest of the factory, and put under a supervisor who had management style akin to a leadership-collaboration style [5].

The experiment consisted in introducing some variations to the work conditions—for example, reducing the number of working hours, increasing the number and length of pauses during the workday, etc. The experimenters introduced the changes always keeping the experimental team informed, asking for advice or information, and listening to complaints. Productivity always went up. Eventually, all the improvements were removed. Productivity was the highest ever recorded.

The final conclusion was that the six women formed a team that cooperated spontaneously and wholeheartedly to the experiment. The team had considerable freedom of movement, was not pushed by anyone, and was involved in every decision that could affect their work. Under these conditions the workers developed a higher sense of responsibility that induced them to do a better job, and at the same time, feel happier and more satisfied.

These experiments evidenced for the first time that workplaces are *social environments*, where people are motivated by many factors other than economic interest. Mayo concluded that recognition, security, and a sense of belonging were more important to productivity and morale than the physical environment. Finally, a friendly relationship with the supervisor was very important in securing the loyalty and cooperation of the team.

These studies were a breakthrough. In fact, at the times when the studies where conducted, the prevailing theory was Taylor's scientific management [12], that was based on the assumption that the main motivational factor for workers was high wages. Mayo’s findings clearly did not match very well with Taylor's theory.

Motivation Theories

After the Hawthorne experiments, several theories have been developed to try to characterize motivation. Each of them has strengths and weaknesses. None of them is general enough to be applied in every situation.

The factors that influence motivation can be identified in two main categories: *intrinsic factors* and *extrinsic factors*.

The intrinsic factors are the ones that come from the work itself and the goals and aspirations of the individual, i.e., achievement, possibility for growth, social relationships, etc.

The extrinsic factors are the ones that depend on the surrounding environment, or the basic human needs, i.e., salary, office space, responsibility, etc.

Some prominent motivation theories—that can help in explaining what motivates software developers—are Abraham Maslow's *hierarchy of human needs*, Frederick Herzberg's theory on *motivators and hygiene factors* [4], and David McClelland's *achievement motivation* theory [9].

Maslow’s hierarchy of human needs classifies the human needs in five levels. According to this theory, higher level needs become motivators only when the lower level ones are satisfied. The hierarchy, ordered from from the lowest to the highest level, is

- Physiological, i.e., salary, office space, appropriate facilities, lightning
- Safety, i.e., job security, pension scheme, medical insurance, sick leave
- Social, i.e., interactions with colleagues and customers, teamwork

- Self-esteem, i.e., reputation, recognition and appreciation from colleagues, subordinates, and supervisors
- Self-actualization, i.e., the realization of the full potential of the individual, “what a man can be, he must be” [7].

Herzberg’s motivators and hygiene factors theory relies on different assumptions from Maslow's one. According to it, there are factors that have a positive impact on the increase of motivation—the motivators, that Herzberg identifies with the intrinsic factors; and other factors that have to be present in order to avoid de-motivation, but by themselves cannot increase motivation—the hygiene factors, that Herzberg identifies with the extrinsic factors [4].

According to this theory, motivators derive from “that unique human characteristic, the ability to achieve and, through achievement, to experience psychological growth” [4]. They are, in order of importance, achievement, recognition, work itself, responsibility, advancement, and possibility for growth.

Instead, hygiene factors are a consequence of humankind's animal nature and relate to the basic biological needs—for example, the need for food makes money a necessity. Some hygiene factors are company policy, office space, supervision, personal life, and salary.

McClelland’s achievement motivation theory characterizes the motivation of a particular class of people—the ones who have a strong desire to achieve. According to this theory, achievement-motivated people have the following characteristics:

- They like difficult, but potentially achievable, goals
- They like to take calculated risks
- They are more concerned with personal achievement than with rewards for success
- They have a strong need for concrete job-relevant feedback. They want to know how well they are doing

These three theories are related with each other. Herzberg's extrinsic factors correspond to the lower levels of Maslow classification, intrinsic factors correspond to the higher ones. Achievement-motivated people tend to be more motivated by Herzberg's intrinsic factors. Achievement itself is an intrinsic factor.

In general, in the workplace, intrinsic factors tend to be much more effective than extrinsic ones in motivating people [13].

Software Developers' Motivation

The first ten motivational factors for software developers—in decreasing order of importance—reported by Boehm [2] are

- Achievement
- Possibility for growth
- Work itself
- Recognition
- Advancement
- Technical supervision
- Responsibility

- Relations with peers
- Relations with subordinates
- Salary

The data on which the list is based is more than 25 years old, but I think it is still valid, since it matches pretty well with my experience.

Achievement is the strongest motivator for software developers, furthermore, most of the other ones are intrinsic factors as well. So McClelland's and Herzberg's theories are suited to explain what motivates them.

It is interesting to notice that the factor that managers tend to use most to motivate employees—salary—is listed in the last position. Actually, I have yet to know a good developer who is really motivated by salary. Certainly I know some very good ones who recently refused highly paid job offers, because the work was not “interesting enough”. Of course money is important, but it becomes a strong motivator or de-motivator only when the availability is respectively very high or very low.

Teamwork

Nowadays, most endeavors are so complex that they can be accomplished only by a team, so it makes sense to know what make teams effective.

Larson and LaFasto [6] undertook a three year study to understand the characteristics of successful teams. The teams studied ranged from football teams to the one that built the Boeing 747 airplane. There weren't any software development teams. They found that all the highly effective teams always had these characteristics

- A clear, elevating goal
- A results-driven structure
- Competent team members
- Unified commitment
- A collaborative climate
- Standards of excellence
- External support and recognition
- Principled leadership

From this list is evident that effective teamwork has a strong relationship with motivation.

A clear, elevating goal is absolutely necessary for achievement. Clarity means that it is possible, concretely and unequivocally, to verify that the goal has been achieved. An example of clear goal can be “the executable must not use more than half gigabyte of RAM at any given time”. On the contrary, a goal stated as “the executable must not use too much RAM” is certainly not clear. A goal is elevating if, from the point of view of the team, it is important or worthwhile. For example, it can be a technical challenge that stretches the skills of the team to the limits; or it can instill a sense of urgency. People want to be involved in something that gives them an opportunity to make a difference, so if the goal is clear but not elevating, achieving it could be difficult, since it could be perceived as uninteresting or even worthless.

A team has a results driven structure when it is organized according to the goal that it has to attain. Team structure comprehends the process, the communication channels, the roles, and the skills of

the team members. It is an hygiene factor. In fact, its presence makes achievement possible, but doesn't motivate people, and its absence is certainly demotivating since it can make achievement, at best, difficult, and at worst, impossible.

Competence has an important influence on achievement motivation. Achievement motivated people like challenging, but potentially attainable goals. Lack of competence can make the goal impossible to reach. There are two types of competencies, both equally important: technical competencies and personal competencies. Technical competencies refer to the knowledge and skills necessary to achieve the team's goal. They are clearly necessary. Personal competencies refer to the personal skills of the individual plus the ability to work effectively in a team—they can make the real difference in team performance. A team of star developers who cannot work well with each other, is generally outperformed by a team of average developers who work well together.

Unified commitment is not easy to define. It is “team spirit”, when individuals feel a strong identification with the team. It is when all team members are willing to devote time and energies for the achievement of their common goal pulling together in the same direction. It is when the team has its own identity. Unified commitment can be fostered, first of all, by establishing a clear, elevating goal. Then involving the team in all the phases of the project. Involvement enhances commitment. If unified commitment is lacking, even if there is a clear, elevating goal, the possibilities of success are severely reduced.

A collaborative climate is described by the phrase “working well together”. It is important to foster unified commitment, a sense of belonging, and to give team members a possibility for growth. In order to have a collaborative climate is necessary for team members to trust each other. In this way, they can focus on the attainment of the goal. Furthermore, communication and coordination are more efficient, and the quality of the outcome is greatly improved.

A standard defines an expected level of performance. It defines expectations on the skill levels of team members, on the initiative and effort they are able to demonstrate, on how the results are to be achieved, etc. A standard of excellence defines a standard in which the expected level of performance is very high. A consequence of setting high standards, is that the expectations on the team become high as well. This positive enforcement can bring the members to exert pressure on each other in order to keep up to the expectations creating a whole that is more than the sum of its parts. Consequently, the self-esteem of team members receives a big boost and so do motivation and product quality. Standards are hard work and require a great discipline, so the best way to make them easier to follow is to make them concrete. They should not be stated as general principles like “the code must be of excellent quality”, but they should be defined in terms of what can be done concretely in order to follow them—for example, they can mandate the usage of unit tests, refactoring and pair programming as techniques to keep the quality of the code high.

External support is about giving the team the resources it needs to get the job done. In motivational terms, it is an hygiene factor. Without sufficient external support it is very difficult to achieve any goal. Furthermore, it gives the team the message that their work is not very important (making the goal less elevating), with consequent drops of motivation and morale.

Recognition are the rewards linked to achievement. The rewards must be tied to performance and viewed as appropriate by team members. Recognition is a strong motivator for software developers.

Leadership is one of the most critical factors for effective teamwork. A very effective leadership style is what Larson and Lafasto call a principled leadership [6], and Highsmith [5] calls leadership-collaboration. Principled leaders don't give orders, they inspire and influence people, they trust their followers to get things done, and use power only sparingly. Effective leaders, according to Larson and Lafasto [6] “(1) establish a vision; (2) create change; and (3) unleash talent.” In a team with this kind of leader there is a great opportunity for responsibility, technical

supervision, and advancement.

In conclusion, effective teamwork are strongly tied together. Most of the characteristics of effective teams are motivators or hygiene factors, and the remaining ones have a direct effect on it.

Agile Development and Teamwork

In this section I'll show how agile development methods have the basic characteristics that make effective teamwork possible.

A Clear, Elevating Goal

Iterative and incremental development along with user collaboration plays a central role in keeping the goal visible and clear.

The usage of incremental development allows the developers and the customer to define and work on smaller but clear goals. An important part of the increment is the definition of—often automated—acceptance tests. Their definition is what really makes the goal clear to the team.

The usage of, preferably short, iterations allow the team to have the feedback necessary to understand if what they have done is what the customer expected. In fact, acceptance tests are very helpful, but the experience of using the software gives the customer a better understanding of her needs. This, often, leads to a refinement of the goal without a loss in clarity.

Making the goal elevating is not an easy thing. There are no sure recipes for that. Certainly, having a customer who closely works with the team can be of great help. In fact, she can continuously remind the (great) importance the software has for her or her company, or she can instill a sense of urgency making it clear why the product must be absolutely ready for a certain date.

Some often successful techniques are to create a challenge by setting up tight, but achievable, deadlines or to give the team the possibility to learn new skills. In a project I'm currently involved, my teammates and I used both of these techniques to make it interesting. Up to now it has been a great success—we and the customer are both satisfied.

A Results-Driven Structure

Agile teams are structured in order to deliver valuable software on time and on budget in a context of frequent changes in requirements.

An effective team structure has four necessary features [6].

First, there are clear roles and accountabilities. For agile development some of them are defined by means of the rights that the customer and the development team have.

The customer has the following rights

- The right to an overall plan that defines what can be accomplished, when, and at what cost
- The right to receive the maximum value for each iteration
- The right to change or substitute priorities without incurring in exorbitant costs
- The right to be timely informed about schedule changes, so that scope can be changed, or to cancel the project and still have an useable system

The developers have the following rights

- The right to clear requirements with clear priorities

- The right to always produce quality work
- The right to request and receive help from their peers, their managers and the customer
- The right to create their own estimates and to update them when the problem becomes more well defined
- The right to accept their responsibilities rather than having them assigned to them

The roles and accountabilities inside the development team depend on the specific methodology used.

Second, there is an effective communication system. Agile development puts an emphasis on face-to-face communication—arguably the most effective communication channel between human beings; the team members tend to be located close to each other, possibly in the same room—so the speed of communication is optimized; the customer is encouraged to interact closely with the developers—so the feedback loop is shortened and the goal remains visible and clear.

Third there is a way for monitoring individual performance and providing feedback. In agile development this is a consequence of the high level of interaction between the parties involved. If someone is not doing his best, this becomes very clear very early to everybody.

Fourth, all the judgments are fact-based. Agile development methods submit all activities and products—including software—to an usefulness test: they must contribute in some way to the achievement of the goal, otherwise they are dropped. The process is streamlined by executing only the activities that simplify the work of the team. Documentation is written only if there are people willing to read it. Software is kept as simple as possible, so it is easier to change. Future extensions will be examined when the need will arise. Gold plating is loathed and avoided. Code quality is kept as high as possible. Finally, technology is used only when necessary—very often using a whiteboard or CRC cards during a design session is more effective than using the latest CASE tool.

All these techniques allow the development team to travel light and focus only on what matters for the achievement of the goal.

Competent Team Members

Agile development methods need people with both technical and personal skills. In my opinion, the latter are the ones that are more important, since at the end make the real difference. This goes against the common belief that agile methods require a higher proportion of expert developers in the team than RSMs do. In fact, in my experience, the ability *of learning new skills, to adapt to changing situations,* and to *apply acquired skills in new ways*, for agile software developers are much more important than having strong technical skills. Furthermore, the high level of interaction required by agile methods, require people that can collaborate effectively with others—practices like collective design sessions, pair programming, and collective code ownership could be impossible to implement otherwise.

Unified Commitment

Agile methods tend to involve the entire team in all phases of development. The customer is in close contact with the team, so everybody can better understand the requirements, and have the goal clear. Design sessions make extensive usage of techniques—such as CRC cards and whiteboards—that bring the whole team together to discuss design and programming issues, in which everybody can give a contribution. All these things help greatly in fostering unified commitment.

A Collaborative Climate

Agile methods put a strong emphasis on collaboration. The first value of the Agile Manifesto clearly states that individuals and their interactions are more valuable than processes and tools, and customer collaboration is preferred to contract negotiation. The setting of a collaborative climate is one of the main objectives of several agile practices. The emphasis on face-to-face communication, shared or, at least, very close office spaces, common design sessions, collective code ownership, and customer collaboration, incremental and iterative development, are all factors that help in creating a climate of trust and collaboration.

Standards of Excellence

The level of performance to which agile methods aim are quite high: satisfy customers with continuous, on time, and on budget delivery of valuable software; and write technically excellent software— that is, few minor bugs, self-documenting, fully tested, simple, and modifiable.

To make standards easier to follow, agile methods make use of some concrete practices such as iterative and incremental development, extensive usage of testing, code refactoring, and coding standards, and keeping the code simple and self-documenting.

Some methods—for instance Extreme Programming—also use pair programming and promote the concept of collective code ownership. These two techniques are a powerful incentive to keep up to the set standards, since every line of code can be, potentially, read and modified by any member of the team.

External Support and Recognition

Agile methods recognize explicitly the importance of external support. In fact one of the principles of the Agile manifesto states “build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.” They recognize that, without appropriate resources, software development is simply not possible.

The allocation of office space, putting developers in close with each other, the emphasis on face-to-face communication, the availability of appropriate development tools, and close customer collaboration require a great deal of external support to be implemented.

As far as recognition is concerned, when it comes from an happy customer it has a powerful effect. The simple act of showing appreciation for job well done is a very powerful motivator. It helps in increasing the self-esteem of the developers and the level of trust between them and the customer. This, in turn, leads to better communications and better software.

The usage of iterative and incremental development can be instrumental in increasing the external support and recognition. A successful team that delivers early and continuous results is more likely to receive the support it needs from all the stakeholders.

Principled Leadership

The leadership theme is not dealt with directly in the Agile Manifesto and its principles. Generally, each method has its own way to deal with it.

There is a common theme that is clear from the literature: leadership is important, and it must be a principled one [5] [11] [10].

A principled leadership is a natural fit for agile development, in fact is the most suitable for environments where change is the norm [5], and is the only style compatible with the agile values

and principles. A command-and-control style wouldn't be a good fit for an environment in which "individuals and communications" are valued more than "processes and tools." [1].

Conclusion

I have shown why I believe agile methods can be more productive and appealing for developers. They leverage the most important factor for productivity and morale: motivation.

The role of processes and tools in this context is still very important, since they are used to streamline all the repetitive tasks, letting the developers focus on what really matters: to satisfy the customer by producing valuable software.

Acknowledgments

I thank Mike Cohn for sharing his ideas with me, and for reviewing the article. I also thank my colleagues Alexander Fedotov, Rodrigo Fernandez, Federico Garcia-Diez, and Renato Mancuso for their help and support.

About the Author

Giovanni Asproni is an Italian Software Craftsman currently working as Senior Software Engineer for the European Bioinformatics institute, near Cambridge, UK. He is a member of the AgileAlliance, the ACM, and the IEEE Computer Society. He can be reached by e-mail at aspro@acm.org, or, through his website, at http://www.giovanniasproni.com.

References

[1] Beck, K., et al., *The Agile Manifesto* , http://www.agilemanifesto.org
[2] Boehm B. W., *Software Engineering Economics*, Prentice Hall, 1981
[3] Cockburn, A., *Agile Software Development*, Addison Wesley, 2002
[4] Herzberg, F., *One More Time: How do you Motivate Employees?*, Harvard Business Review, 1968
[5] Highsmith, J., *Agile Software Development Ecosystems*, Addison Wesley, 2002
[6] Larson, C., E., LaFasto, F., M., *Teamwork: what must go right / what can go wrong*, Sage Publications, 1989
[7] Maslow, A., *Motivation and Personality*, Addison Wesley, 1987
[8] Mayo, E., *The Human Problems of an Industrial Civilization*, Macmillan, 1933
[9] McClelland, D., *The Achieving Society*, The Free Press, 1967
[10] Peters, J., Waterman, R., H., *In Search of Excellence: Lessons from America's Best Run Companies*, Harper, 1982
[11] Poppendiek, M., Poppendiek, T., *Lean Software Development*, Addison Wesley, 2003
[12] Taylor, F., W., *The Principles of Scientific Management*, Dover Publications, 1998
[13] Thomas, K., W., *Intrinsic Motivation at Work: Building Energy & Commitment*, Berrett-Koehler, 2003
[14] Weinberg, G., M., *The Psychology of Computer Programming: Silver Anniversary Edition*, Dorset House Publishing, 1998