On Management #4: Caveat Emptor

The previous article in this series peeked inside the organizational form. This is a massive subject in it's own right and one that determines what management roles exist and what is expected of them. Organizational form is in part a function of what the company is trying to achieve. Rather than discuss structure in detail I want to turn my attention to the roles we find in software development groups. However, before I do so I want use this article to offer some warnings.

Management isn't homogenous

How does one define *management*? Or rather, how does one know management work when we see it?

Perhaps in years gone by management work could be defined as that which was not manual labour. Rather than assembling things on a production line managers organized the production line; rather than dig minerals out of the ground managers concerned themselves with finding people to dig out minerals, selling the minerals and accounting for the profit and loss.

On this definition managers are those who work principally with their brains rather than their hands – some would distinguish between white collar workers and blue collar. But on this basis software developers are managers because their work is principally in the mind.

Alternatively we could say than managers concern themselves with organizing the work rather than doing it. Extending the analogy to software development, managers are those who don't code. But this leaves out software testers and others. So perhaps the definition is *those who do not directly touch the product during development.*

Lesson 1: What constitutes *Management* and what the dividing line is between *manager* and *worker* has never been well defined and is even more blurred today,

On this basis we might exclude Business Analysts from the management group. Yet (as we will see in a future article) the role filled by Business Analysts in some companies is filled by Product Managers in others. And since Product Managers have the word *Manager* in their title they must be managers, QED.

Unfortunately the word manager is somewhat abused. This is most obvious when writing program code. Naming an object "Manager" – for example SecurityManager or LogManager – is usually an indication that the object is poorly understood and ends up being a collection of functions with a vague connection.

Not only is the word *manager* is added to a title in an attempt to explain a vague role it is also added as a form of aggrandisement. Title inflation means that some *managers* manager little more than their own time. While those who truly do manage many people – or things – become *Directors*.

A corollary to this confusion is that managers manager different things in different ways. The work of a Product Manager is different to that of a

Project Manager which is different to that of a Line Manager which in turn is different to.... –get the picture?

It is a mistake to think that there is a theory of management and that it apply to all those titled Managers. While there are theories of management they are not universally applicable simply because there is no universal role of manager.

Lesson 2: Different management roles operate in different ways.

Given that there is confusion of who is, and who is not, a manager, and that different managers operate under different conditions with different objectives it is pointless to see a *them and us* divide.

Caveat emptor: The Power and Danger of theory

> "The ideas of economists and political philosophers, both when they are right and when they are wrong, are more powerful than is commonly understood. Indeed the world is ruled by little else. Practical men, who believe themselves to be quite exempt from any intellectual influence, are usually the slaves of some defunct economist. Madmen in authority, who hear voices in the air, are distilling their frenzy from some academic scribbler of a few years back." *John Maynard Keynes (Keynes 1936)*

When coding, in theory at least, there is a *right answer* – true it isn't always so straight forward in practice but most decisions are contingent. That is to say, given a set of conditions the next action can be predicted.

Even when there is a dispute over the *right answer* it should be possible to conduct an objective experiment and measure the result – which version executes fasters, which is easiest to understand, etc. Conducting the experiment will not change the context, conduct it again and the result will be the same. Judgement and intuition are usually used to sidestep the need for an experiment and speed things along.

Management isn't like that. Management is about dealing in ambiguous situations, there are many more variables, some variables are unknown and some defy logical - because they often concern people and emotions. Management isn't contingent; intuition and judgement are not short cuts but a way of life.

Lesson 3: Management occurs in ambiguous situations with missing data and incomplete understanding of the problem. Only sometimes is it possible to postpone a decision and collect this data. Other times judgement, intuition and clear thinking are required.

Even if there were a set of rules for management then managers would still need intuition and judgement. The passing of time is not a neutral event. Sometimes waiting a little while can resolve a problem but on other occasions delay can make things worse.

Instead managers seek objectivity in theory. And since management is fundamentally a social science, these theories are social theories that are difficult to replicate in an experiment. In fact, the application of these theories changes the environment.

Lesson 4: There are very few hard facts management - it is an art not a science.

There is a famous experiment (Goldman 1996) were pupils at one school were divided into two groups: poor performance and high performers.

One group of teachers were told that the pupils they were teaching were under performs and not were not expected to achieve much. The other group of teachers were told the opposite; that their pupils were high achievers and great things were expected of them. In fact the pupils had been divided randomly between the two groups. When tested several months later the supposedly "under performing" pupils did indeed under perform while the "high achievers" did exactly that.

Lesson 5: You get what you expect, expect the worst and it may well come to pass.

(This experiment was conducted in the 1950's, today's ethical standards would not permit the "under performers" to be treated in this way.)

Management theories exist to shape the way in which managers' react. Yet they are dangerous for exactly this reason. A manager who mentally holds a poor theory in their head will make decisions based on that theory – exactly the same as they would if the theory was good.

Yet the only way to test a management theory is to use it and in using it the context is changed. There is no such thing as statelessness or side-effect free management.

For these reasons, and others, many management theories are not only unproven but unprovable by way of experimentation. Learned management journals like the Harvard Business Review and MIT Sloan Review publish plenty of management theories and advice, many of these articles are based on case studies rather than experiments. Observation and reasoning is about as good as it gets.

Better people than me have pointed out the problems with management theory. If you want a real tour de force on the subject read *Bad Management Theories Are Destroying Good Management Practices* (Ghoshal 2005).

Lesson 6: Bad management theories can be very destructive.

For the purposed of this Overload series I will strive for objectivity, I will provide references and examples were I can, but it is impossible to be totally objective. What I write will always be coloured by the theories I believe in.

In managing software development work, and in advising others on how to improve their software development I find that it is the hundreds, thousands, of small decisions made every day that make the difference. While it is nice for managers to consider big ideas, like organizational structure and corporate strategy, this only forms a small part of management work. Most management work is in the small everyday decisions.

The thousand small decisions made everyday are usually made based on intuition and our own underlying beliefs. Each big difference is made by a thousand small decisions. Every decision is an opportunity to effect the direction, means and effectiveness of work.

Lesson 7: Making the most of every decision opportunity requires managers to have a clear goal and vision of how the goal can be achieved. A manager's personal philosophy of management plays a key role in ensuring consistent decision making.

Software Management

In Mythical Man Month Fred Brooks wrote:

> "In many ways, managing a large computer programming project is like managing any other large undertaking - in more ways than most programmers believe." (Brooks 1975)

Brooks was right, there is much in modern management literature that applies to the management of software projects. Those charged with managing software development can learn a lot by looking beyond the software community for ideas and practices.

Brooks continued to say:

> "But in many other ways it is different - in more ways than most professional managers expect." (Brooks 1975)

Perhaps surprisingly some in the management community look to the software profession for examples of good management practices (e.g. McFarland 2008; Sull 2007). Software development is both very forgiving of poor practice and very sensitive to it. Even poor management with poor development practices can produce software that generates revenue but to produce great software, and to continue creating great products requires excellence in both domains.

So while managing software work is a lot like managing anything else it is also more different. Which raises the question: does one need experience as a software developer to manage software development? There is no simple Yes or No answer to this question.

Rather than single out software development let us try some alternative questions:

- Does one need a background as an accountant in order to manage the financial operations of a company?
- Does one need a background in marketing in order to manage the marketing department?

There will always be individuals who are so able, so skilled, that they will be able to turn their hand to managing anything. Such individuals will have quick minds, be good people managers and be excellent at listening. And it is a truism to say such people are few and far between.

Lesson 8: In general, managing IT work is best done by those who have experience of IT work.

So, on the whole a software development background is necessary to manage a software development group. However such a background is not in itself sufficient to manage a group. Different skills are required, skills such as people management, listening, political-acumen, organization and others. Some will posses these skills already while others will need to learn them.

But, the skills that make someone a good software developer can also trap that person when they move to management. Understanding code is good, however, jumping in to change someone else's code is not good, it undermines trust and responsibility. A fascination with technology can drive a good developer but the same fascination can mislead a manager.

As a developer the first thought upon hearing a problems should be: its a problem with the code. As a manager it pays to look beyond the immediate problem, Jerry Weinberg has said: *Its always a people problem* (Weinberg 1985). Looking for a technical problem when the real issue is a people or process issue can be comforting but also wasteful.

In a way, technical problems are the easy bit. Compilers don't get upset when you find a bug in the code, computers don't sulk when your software causes them to crash again and switches don't have bad days and only route half the packets.

The hard bit is the people bit, its the soft stuff. The truly difficult stuff is the interlocking processes and systems that we find in organizations. The same systems and processes that allow the organization to exist in the first place are also the ones which cause the problems.

Maybe that's why, 25 years after Mythical Man Month, Fred Brooks wrote:

> "Some readers have found it curious that The Mythical Man Month devotes most of the essays to the managerial aspects of software engineering, rather than the many technical issues. This bias ... sprang from [my] conviction that the quality of the people on a project, and their organization and management, are much more important factors in the success than are the tools they use or the technical approaches they take." (Brooks 1995)

About the author

Allan is an IT management consultant helping companies achieve operational excellence in software development activities. He helps companies realise improved effectiveness and efficiency by assisting the adoption of Agile and Lean techniques applied using a combination of training, coaching, advice and interim management.

He is the author of "Changing Software Development: Learning to become Agile" (2008) and holds BSc and MBA degrees. More information at http://www.allankelly.net.

References

Brooks, F. 1975. The mythical man month: essays on software engineering: Addison-Wesley.

Brooks, F. 1995. The mythical man month: essays on software engineering. Anniversary edition Edition: Addison-Wesley.

Ghoshal, S. 2005. "Bad Management Theories Are Destroying Good Management Practices." Academy of Management Learning & Education 4(1).

Goldman, D. 1996. Emotional Intelligence: Bloomsbury.

Keynes, John Maynard. 1936. The general theory of employment, interest and money. [S.l.]: Macmillan.

McFarland, K.R. 2008. "Should you build strategy like you build software?" MIT Sloan Management Review 49(3):7.

Sull, D. 2007. "Closing the Gap Between Strategy and Execution." MIT Sloan Management Review 48(4):8.

Weinberg, G.M. 1985. The secrets of consulting. New York: Dorset House.