On Management 3: Understanding who creates software

When I was at school, studying for my Computer Studies GCE 'O' level I was issued with a little green book of computing terms published by the British Computer Society. Amongst other things this book described the titles and roles found in a computing department.

The book is long gone and so too are some of the titles – I've never met a *Head of Data Processing* and don't expect to. This is shame because I find job titles more and more confusing. What one company calls a "Development Manager" another calls a "Project Manager", what one calls a "Product Manager" another calls a "Business Analyst" and so on.

Being a manager is different to being a developer. As a developer I could buy Scott Meyers *Effective C++* and stick to his 50 rules. Yes C++, Java and C# hard to use and there is a lot to learn but there are important differences between developing code and managing the activity.

Managers work in a far more ambiguous environment than developers. Not only are the parameters within which they work unclear and changing but the actual practice of management is ambiguous.

Neither is it clear what managers actually do. Newly promoted managers often tell me they go home at night wondering what they have done. As a developer it's possible to measure the lines of code written today, or the bugs fixed, or even the UML diagrams drawn. A bad day is one spent in meetings and discussions without any code cut. Measuring management work is more difficult.

Lesson 1: Management software development is different to developing software and requires different skills. It is a mistake to manage people and processes in the same way as files and systems.

Few management books are of help. Books for managers often focus on a specific idea – "re-engineering", "knowledge management" or "outsourcing". General management books discuss the things managers *should* be doing – thinking big thoughts, setting strategy, objectives and measuring value. In fact most days are spent in a constant round of *fire fighting* and crisis control.

Discussing "managers" in general is not very insightful. Management roles are not equal. In this article and future articles I would like to discuss what managers do, and what they "should" be doing. Thus, in this and future articles, I would like to look at the role of management in software development projects by looking at some of the roles managers undertake.

In the same way that management roles differ so too do organisations. Therefore, before looking at management roles it is necessary to consider organizations. Superficially similar roles, with the same title, can be very different in different types of organisation. In order to understand any given role one has to understand the organization, in order to understand the organization one has to know what types of organization there are.

Lesson 2: One size does not fit all; different types of organizations require different structures and different roles. And each individual organization will have its own unique differences. Don't try to force one company into the mould of another.

Thus this article will look at a few of the most common organizations that develop software. As such it will set the stage for future discussion. It is not meant to represent any kind of *best practice* but it is meant to describe a reference model.

Title inflation

Senior managers are now more likely to be called *Directors* whether they sit on the company board or not. Other managers may manage people, things, or simply organise their own time.

As in program code the term "manager" is often used as a general catchall name. The term itself implies a degree of seniority and authority. Rather than actually managing something many "managers" would be better thought of as "specialists". I once worked with several "Product Managers" at a telecoms firm; these managers would have been better described as "Mobile telephone radio specialists."

Small companies with big companies as customers tend to suffer more than most from title inflation. There is a need to appear big, to send people of equivalent "rank" to meetings so titles are often aggrandised for marketing reasons. And all companies are prone to offering title enhancements in place of financial rewards.

Such practices are harmless as long as the individuals and their co-workers don't attach too much significance to the title. One developer of my acquaintance acquired the title "Chief Software Architect". This would have been harmless enough if the individual concerned had carried on as before but with only about 10 developers the company hardly needed a Bill Gates type figure to direct the architecture. What it did need was a software engineer who understood how things hung together; helped more junior engineers design and got their hands dirty with code.

Types of organization

Broadly speaking there are three types of organization which develop software:

- Independent Software Vendor (ISV) or Software product company: A company that produces and sells the same software products to multiple customers. For example: Microsoft, Oracle, Symbian and SalesForce.com. If these companies did not produce software they would not have a business.

- Corporate IT / In house: A company that develops software to support its activities. For example: Barclays Capital, Unilever/Lever Brothers and British Airways. These companies sell a product or service that is not software but develop software to make the product or service. To keep things simple the term Corporate IT is used to include both central corporate IT functions and distributed IT activities, e.g. a bank may well have developers working for the equities trading desk.

- External Service Providers (ESP): A company that develops software for customers. For example Accenture, EDS, Infosys and Thoughtworks. Such companies may also provider additional IT services such as operations control and data centres. For some customers they may provide such services but not develop software. Most customers are corporate who need some IT services.

ISVs do on occasion contract ESPs but since their business depends on their ability to create software this is uncommon. ISVs also have internal IT needs and may contract an ESP to run aspects of the computing system, e.g. Microsoft have outsourced some of their internal support operations to an ESP.

The three categories outlines will serve for reference. This list is by no means exhaustive and new business models are constantly arising which obscures the boundaries. For example, Software as a Service (SaaS) pioneer SalesForce is included here as a ISV but their business model rests on providing a service in a similar way an ESP might. The difference is that the SaaS model offers the same software to all customers.

Other companies produce products that would not be possible without a software element but would not think of themselves as an ISV. For example the makers of digital radio sets are dependent on software but are clearly not an ISV. As more and more products contain complex software – cars, televisions, alarm clocks – more and more companies will be dependent on software for their key products.

The rest of this article will focus on the ISV and corporate IT models. This is not because ESPs are any less worthy but because ESPs usually operate either as an extension of a corporate IT (think outsourcing) or are used to provide a specific product (similar to an ISV).

One of the biggest mistakes made by young ISVs is operating as a Corporate IT department and not an ISV. Software engineering skills are largely the same inside corporate IT departments and ISVs. If you can code Java in a bank you can code Java for a company that sells software. But the same is not true at management level. Managing the creation and delivery inside a corporate requires different skills and judgements to those required to successfully manage the delivery of software products.

Lesson 3: Know what type of software producer you are, and which you are not. Understand what role your software plays in delivering the final product to the customer and generating revenue.

Internal development groups have different objectives, roles and processes to those which produce software for sale. Particularly in the UK where most IT is based insides corporates this is a common mistake. In the US were there is a longer tradition of software product companies there are more role models available to ISVs.

The reverse mistake is not so often made and is actually less dangerous. ISVs live and die by their ability to delivery software, therefore their practices need to deliver and they have a software centric culture. The same is not true in corporate where the culture will come from the main business. If a software development is late or fails, the business will usually carry on as before, in other words the company can afford a few IT failures.

I would like to make the very broad generalisation that ISVs tend to have better practices than corporate IT groups. Since ISVs depend on selling software in order to survive one might expect that their development practices are better than corporate IT departments. After all, if an ISV cannot create good software the business cannot continue.

However, experience shows that even poor ISVs can survive for a surprising amount of time. If they have an existing customer base, or a product that is genuinely innovative they can often scrape together enough money to continue for some time. In the extreme these companies can even trade on their poor quality by selling customer maintenance contracts and upgrades to fix faults.

For those looking to improve the performance of their software development activities high performing ISVs are a good place to look for practices and techniques. These techniques can then be transplanted to the corporate IT world provided account is taken of the differences.

Traditionally corporate IT departments only dealt with internal customers. If they produced a program which was difficult to use the users had little choice but to use it. The IT department could always offer training courses, tell people to read the manual or simply refuse to support other systems. However this is no longer true and corporate IT department need to take lessons from ISVs.

For example, until 10 years ago the IT department of a package holiday company only had internal users, and perhaps the a few travel agents. Today they may be asked to build a website to be used directly by customers to book holidays. If the customers find the website hard to use, confusing or slow they may go elsewhere. Ten years ago, the few customers had no choice – they had to wait.

So corporate IT departments face changing times. Together with the traditional internal only systems they are used to developing and supporting they are also being asked to develop customer facing systems. These systems need a different approach and require different skills to build.

Corporate IT roles

Inside a corporation the IT department is just one more function alongside Marketing, Finance and so on. Such a group will be lead by a Director of IT or Chief Information Officer (CIO) – or, if I remember my BCS booklet the Head of Data Processing. The structure of the group may look something like Figure 1. The size of an organization will have an obvious effect on this chart. Larger organizations may have more levels and smaller companies may combine roles and groups.

Software development is only one part of the CIO's responsibilities. Quite likely there will be an operations group and maybe a business analysis group. Software Architects may report directly to the CIO or they may report to another senior manager, it all depends on the type and role of the architect.

Where an organisation uses external suppliers – ESPs, ISVs or rents data centre space – there may be a group to co-ordinate this work too.

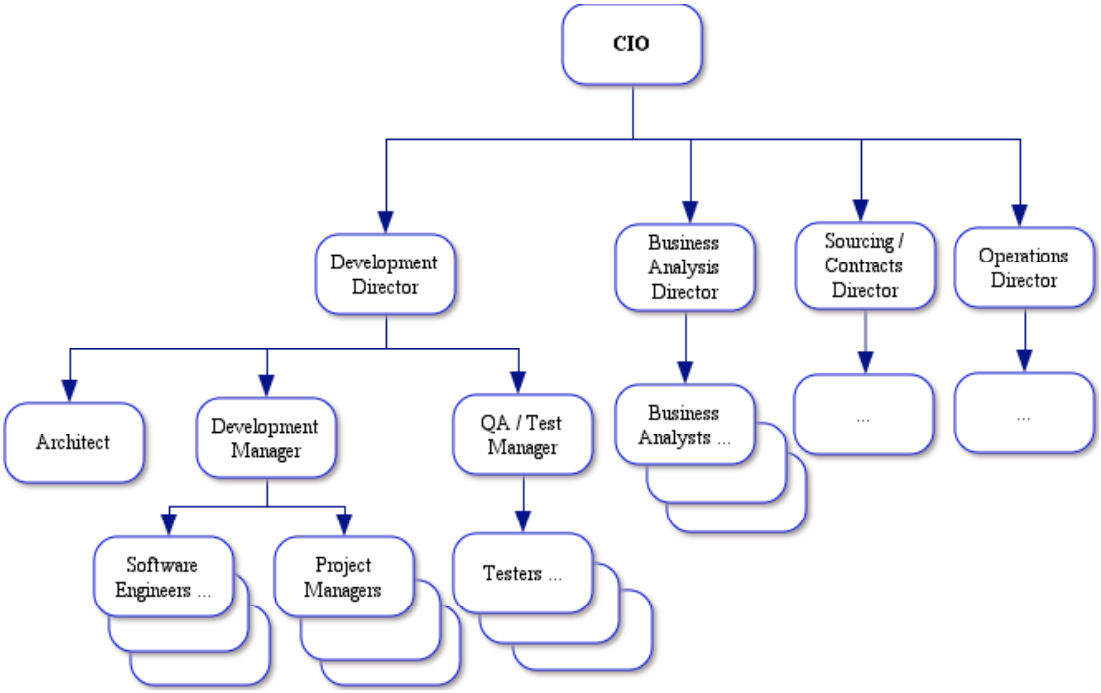

Figure 1 - Corporate IT structure

The development of new systems is just one part of the CIOs responsibilities and would normally be headed by a senior manager such as a Development Director. Reporting to this director would be one or more development managers and one (or even more) QA/Test Managers.

On this diagram project Managers, Testers and Business Analyst have been shown at different levels. The important point is that each of these groups exists as a group in their own right, the level at which the head of the group reports varies.

This description is static and shows reporting lines. Actually creating and maintaining software requires that a team is brought together from different groups. Corporate IT groups may have several software development teams – as in Figure 2. Project teams may be short lived, lasting a matter of months, or they may last for years building and supporting the application.

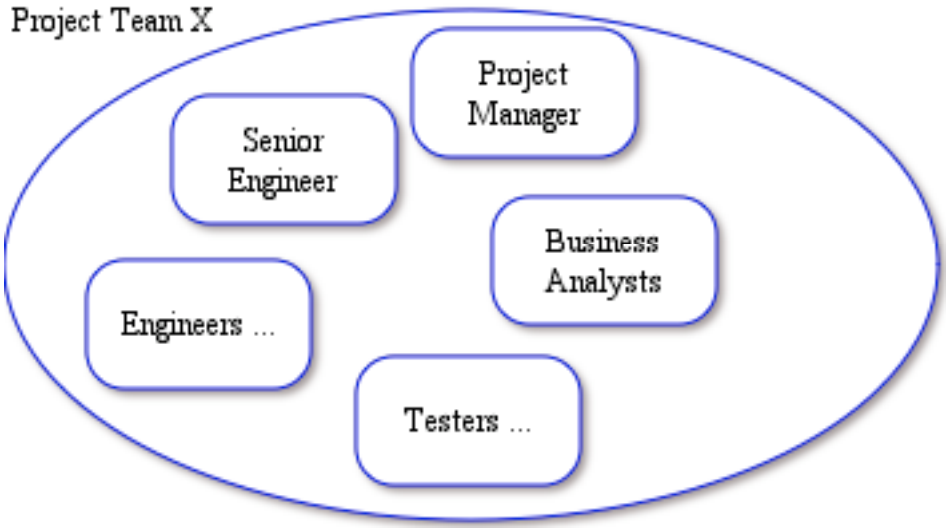

Figure 2 - A project team

As a result of this structure the actual workers – engineers, business analysts, testers, etc. – are considered a pool of resources and matrix management is common. So for example, a Software Tester would report to the Test Manager for line (or personnel) matters and professional test issues but to a Project Manager for the specific project.

The three key features of this reference model are:

- CIO is head of the organizations
- Project teams are drawn from resource pools
- Matrix management

Lesson 4: Corporate IT departments exist to support a business. Software development is not the business; it is only a means to an end.

Independent Software Vendor

Technology companies may well have a CIO role as described above. Like all other companies – especially when they get large – there are corporate information needs, and the need for corporate IT services. However, when a company's life blood is technology itself - and specifically software vendors – there is a need for another role, the Chief Technology Officer or CTO.

While the CIO role is internally focused on processes and systems to support the business, the CTO role and the organization they lead is focused on the application of technology to create products. This creates a different organization with different roles.

Things get a little confusing when the company does not sell technology but sells a product or service that is inherently dependent on technology. For example an online retailer like Amazon or a travel company like Expedia.

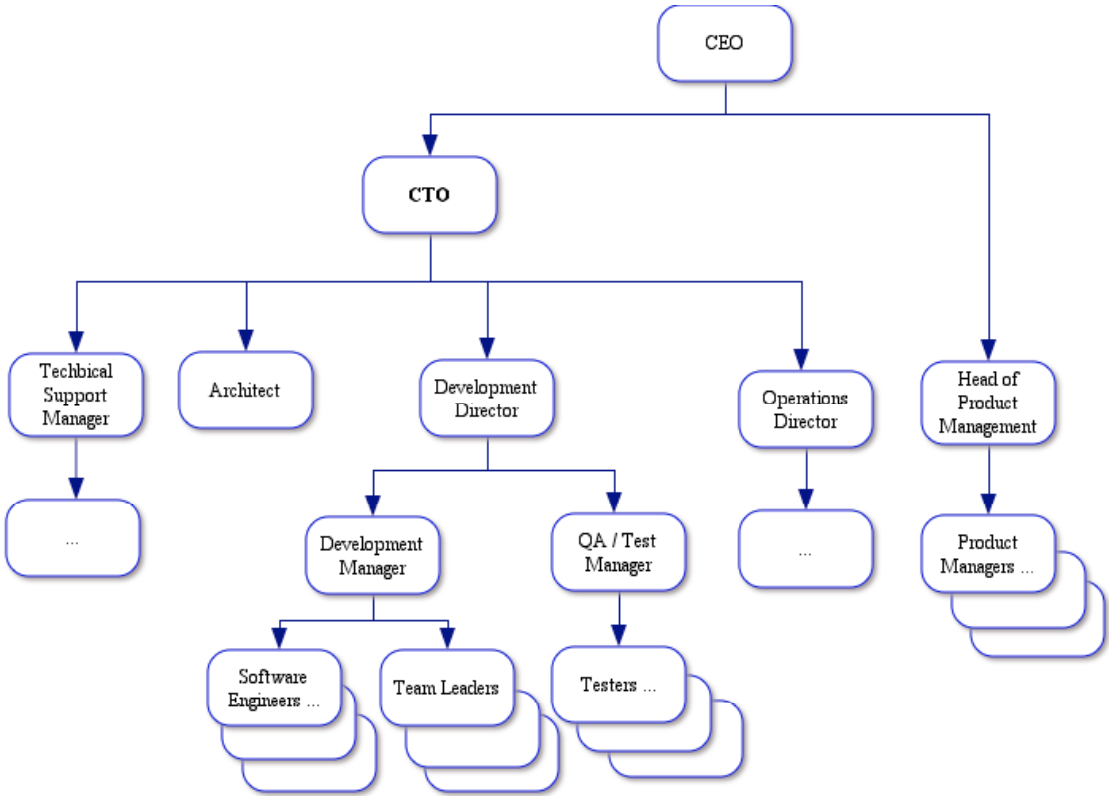

Figure 3 - Technology company structure

A technology company has roles that don't exist – or don't exist in the same way – as a non-technology corporate and this is reflected in the corporate structure shown in Figure 3.

Some CTO's choose to take a hands-on role in managing the development department. Other CTO's define their role as architects and involve themselves directly with the development of products – even coding – while leaving it to a Development Director, or Vice-President of Engineering to organize the department. Another type of CTO concentrates their efforts in the board room and may spend most of their time making strategy or evaluating merges and acquisitions.

Lesson 5: The CTO's role is what the CTO and other senior managers choose to make it.

Companies with multiple products may have product heads who run their own organizations within organizations. Development teams and other resources working on one product may have little involvement with those working on a different product.

Traditionally software companies that delivered software on a disc had no need of an operations department. Now when software is delivered online (as a service) there is an operations element thus there is usually a Technical Operations group ("TechOps") also reporting to the CTO.

Whether delivering on a disc or online there is support to users so there are often support desk operations. These sometimes report to the CTO but more often report elsewhere, say to sales or client services.

Perhaps the biggest difference is the replacement of Business Analysts reporting to the CIO with Product Managers reporting to the CEO. In a technology company knowing what technology products to develop is very important thus they report to the CEO – of course some companies will have them reporting to the CTO.

The Business Analyst and Product Manager roles will be discussed in future articles but for the moment it is enough to say that while the BA is inward focused, looking at systems and processes within the organization the Product Manager is outward focused looking at what customers want. Both roles feed the development teams with requirements and requests but they discover their requirements in a different fashion.

The inward looking nature of business analysis makes it well suited to the corporate IT world where systems are developed for internal users and to change company processes, while the customer facing nature of Product Managers makes them more suitable to ISVs.

Lesson 6: Both Product Managers and Business Analysts can create requirements for development teams. In an ISV it is typically outward looking Product Managers who supply requirements while in corporate IT departments it is typically inward looking Business Analysts.

In recent years Scrum (Highsmith 2002) has popularised the term "Product Owner". This role is decides what the development team should be working on but it does not prescribe how the decision is arrived at. Making these decisions required the skills of a Product Manager or Business Analyst.

Whatever the role is called it is important that someone is concerned with what the software will do, that someone is asking what the customer or user needs and that that person is directing the team on what should be developed and when.

Lesson 7: Software Developers should not be deciding what to develop and when to develop it. This is a separate role and should be filled by someone with the appropriate skills.

When this role is not filled explicitly there is a void. Nature abhors a vacuum and this is no exception. Sooner or later someone steps in to fill this void even when they are not explicitly tasked to do so. With luck this person is motivated to do the job, is knowledgeable about the field and has the right skills. Unfortunately it is also possible the person who steps in is not knowledgeable, has the wrong skills and has their own agenda.

Conclusion

The structures given here are examples to help discuss the role and responsibilities of management. They are also intended to highlight the difference in organizations that develop software.

There are countless variations on these models - not least those caused by culture and national differences, and business trends and fashions. The arrival of CEO – and other "C" level - officers in British companies is

relatively new. Not long ago the top person would be the Managing Director.

These models categorise development according to business model. Alternative categorisations could be by size – small or large – or according to project or product focus. As a general rule-of-thumb ISVs are considered as product focused and corporate IT as project focused.

At a technology level – Java, Windows and such – there is often little difference between the technology company and the corporate IT department. But in terms of managing there is a world of difference. Creating systems to support a business is very different to creating systems to sell are two different tasks.

Unfortunately it is becoming harder to determine which is which. As companies come to depend on technology to deliver their products they take more of the characteristics of technology companies, but if they continue thinking like a corporate IT department the results will be disappointing at best.

References

Highsmith, J. 2002. Agile Software Development Ecosystems: Addison-Wesley.