coding {the} architecture

Project Management for IT People

A short guide to the fundamentals of IT project management

Kirstie Brown

Project Management for IT People

A short guide to the fundamentals of IT project management

Kirstie Brown

This book is for sale at http://leanpub.com/projectmanager

This version was published on 2015-09-10

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

For my husband Simon, who gave me the courage to write this.

Contents

1. About the book

This book is mainly aimed at anyone who is working within the IT industry that has an interest in IT project management and how it affects their role on a project. It is an ideal read for either new or experienced project managers, project team members or management as the aim of the book is to be small, compact, easy to pick up and flick through or to read cover to cover

The book covers the main principles of project management and the software development lifecycle with scenarios, tips and examples drawn from real life experiences that are used to explain how to overcome some of the challenges IT projects face in an ever-changing technical environment.

Why am I writing this book?

My background is in software development and, having made the decision to become more involved with managing projects, I noticed that most websites, articles and books discuss project management from a very generic perspective. There seem to be very few books that take a more holistic view of *IT* project management including what skills you need, problems you may come up against due to technology and general tips on project management principles. After all, technology projects have their own unique activities and challenges.

In addition to this I have worked within a project management office, where I provided guidance and support services for project managers who worked within a complex and changing IT environment. In undertaking this role, I found that the advice, support and document reviews I undertook started to show some common trends. These include a lack of understanding of how to define and present IT projects to the business and common mistakes made in day-to-day project management.

This trend led to me sounding like a “broken record”, repeating the same advice over and over again to different project managers. That started me thinking about writing a short book that is clear, concise and quick to read, covering the skills, principles, tools and activities that will help you to successfully manage an IT project in the real world.

About the author

I live in the beautiful island of Jersey[1] (the largest of the Channel Islands) with my husband and family. Prior to that I lived in Kent, where I grew up. I studied Computer Science at the University of Reading and have since worked for a number of IT organisations in London. I have worked for one of Jersey's largest organisations where I was responsible for running the Information Services Project Management Office. My interests are in project management, agile and change. With a background in software development, I have an understanding of the technical issues faced in project delivery from first hand experience.

[1] http://www.jersey.com

2. Why is IT project management different?

Project management has been around for ages and it is used in all sorts of industries, such as construction, engineering and manufacturing. Even though there are some common areas between IT project management and other industries such as they all cover initiating, planning, executing, controlling, reporting, and closing out a project, there are some differences within IT that makes being an IT project manager more challenging.

Some reasons why IT project management is set aside from the more traditional project management are:

- In IT the major focus is the development and implementation of systems or products, therefore an IT project manager is also responsible for the development, configuration management, and quality assurance activities.
- There are rarely two IT projects that are exactly the same, even a hardware upgrade project will have new and unknown elements to it which makes it different. This could be due to environment, infrastructure, hardware or people skills.
- Most IT projects will need to integrate with existing systems that are currently being used in the business.
- People find it hard to visualise the end product on IT projects, so gathering requirements is key.
- Software systems are not just about the functionality, they need to also address areas such as security, performance and scalability.
- IT Projects are usually higher risk projects, due to new technology being used.

I have heard many project managers who have taken on an IT project say *"I didn't realise we needed to do that"* or *"How do we release the system into production?"* the week before the go-live date. I have even seen issues and faults logged on a system released for user testing being managed within emails! A big part of managing an IT project is knowing what technical decisions and environments need to be in place and at which stage of the projects life cycle.

Regardless whether the company that you work for produces software, buys software or uses a consultancy company to build software, the delivery of the project will have the same distinct stages:

1. **Understand** the business problem and why we are doing it.
2. **Define** what is needed.
3. Work out **how** we can deliver it.
4. **Build** or implement what needs doing.
5. **Test** that it actually does work and meets the business need.
6. **Release** the software to the business so they can start using it as part of their daily business.
7. **Support** the system in the business environment.

An IT project will go through these stages regardless whether you are running a traditional waterfall, hybrid (a combination of agile and waterfall) or an agile project, the difference is you may have multiple iterations of stages 2 - 6 with agile approaches.

Over the years I have seen so many agile projects lose track on what the project purpose due to the project proceeding with little management. There are active discussions on whether a project manager is needed on agile projects and can be seen, by some, as an unnecessary expense as the development team has it all in hand. The project manager role on an agile project is even more important than in a more controlled approach (such as Waterfall) however, the questions you ask your team and how you manage the project are different but the role of a project manager is essential.

IT project management is no more or less important than project management in any other industry, but in order to manage an IT project successfully you need to be aware of how the development, configuration and software quality processes work.

Is it a project?

Many times I have heard the phrase “it’s not a project, it’s business as usual” when really the activity *is* a project. Another example of not understanding projects is where large projects are defined as a “programme”, just because it is complex, has many elements or because it sounds more important.

There are many definitions of what characterises a piece of work as a project and you may find that your organisation has a standard already set. If this is the case then you should adopt that approach. If not, then the following is a good starting point.

A project is a set of activities that are not repeatable and you are likely to only perform them once. An example of a project would be upgrading a specific server versus a "business as usual" activity such as adding a new user account to the network.

A project will:

- satisfy a business need, solve a problem or increase efficiency.
- have a defined scope.
- have funding approved.
- have risks.
- have a defined set of stakeholders.
- have a defined set of people responsible for delivery.
- have a defined start date.
- have a defined end date.
- have a temporary team.
- it will have an IT element.
- it's unique.

What's a programme?

A programme differs from a project, as it is normally made up of a number of different streams of work that have been grouped together to deliver a specific set of business objectives or goals. The streams of work that can be included in a programme are:

- projects in their own right.
- changes to business processes.
- business as usual activities.

Objectives or goals are normally grouped together due to a strategic decision, where the organisation wishes to improve their business benefits, processes or efficiencies. Programmes are normally part of an organisation's vision for the future.

What's a portfolio of projects?

A portfolio of projects differs from a programme because it is purely a collection of *projects* that a project manager is responsible for delivering. The projects are grouped together for one or more of the following reasons:

- the projects relate to a common business objective or goal.
- the projects are dependent on each other.
- the projects complement each other.
- to improve overall efficiency.

What's portfolio management?

In Information Technology it is fair to say, that there are more projects ideas created than there is time or money, to physically to complete them all at once.

This means your organisation needs to have a way of prioritising and selecting a subset of projects that will increase their profit, make efficiency savings and get the best return on investment for the business. This is what portfolio management is about. Projects are prioritised against a set of criteria that allows your organisation to get the best value for money, at the right time and meet the business demands.

Portfolio management is different to project management, but a project manager can be responsible for ensuring a portfolio of projects are delivered successfully.

What's "business as usual" (BAU)?

"Business as usual" is any task or activity that is carried out as part of a person's normal work duties. It will normally have the following attributes:

- it can be repeated multiple times.
- it follows a pre-defined process or set of steps.
- has very low risk associated to it.
- does not require additional funding, it is budgeted for as part of the organisations operating costs.

Why start a project?

Some people may think IT projects just appear on their list of work to manage, and they don't need to consider why they have arisen, but there is always a reason why a project has been formed. Most projects are commenced due to a business idea that has been factored into an organisation's business plan. If a project isn't driven from a business need, then you should be questioning why you are doing it.

Regardless of whether your organisation undertakes their projects in-house or through an external IT supplier, the reason why a project has come about is the same. Most projects are started because:

- there is a common need to do something.
- a strategic decision has been made to invest in a change to the business.
- there are efficiency savings that can be capitalised on.
- there is a need to keep up with changes to technology.
- existing systems or infrastructure have come to the end of their life.

IT projects can fail, but why?

In my experience I have yet to find someone who can hand on heart say that one of their projects went exactly according to plan with no surprises on the way (if you are the first then please let me know). Now, a completely failed project is easy to identify as the project has been canned and the team have been sacked, but that is quite dramatic. There are more subtle indicators that you are managing or are involved in a failing project.

- **Your project is costing a lot more that it should have**. At the end of the day cost is one area that the management team will be looking at and it needs justifying why it cost more. This could be for a number of reasons such as, an unknown cost was identified, more people were needed in order to bring the project in on time or because not enough was known up front before the team started working on the solution.
- **It is taking a lot longer than expected**. If things take too long to see results then people loose interest, also the longer it takes the higher the cost. The reason why some projects do take longer can be due to an inappropriate deadline imposed at the start of the project that was impossible to meet in the first place. If this is the case, then it is questionable whether the project should have actually started. Estimates can also be at fault for things taking longer due to your team guessing how long tasks could take

and believing giving a figure is better than saying they don't know or if an accurate estimate was given by your team, but it has been "tweaked" along the way as "surely it can't take that long to do?"

- **The deliverables do not do what they are supposed to do**. If a system doesn't do what the business needs then it has no value and will not be used, therefore it is an unrecoverable cost. Rushing through the business requirements and not having a clear understanding of the business problem will be contributors to this failure.
- **No one is happy about working on the project**. If your team is unhappy then there is normally a reason for this that relates to, excessive workload, clarity on progress is needed or a dysfunctional team.
- **Functionality and tasks are being cut out in order to make a deadline**. Although IT projects have a change control process, from experience this is normally used when more is requested rather than less. If features and quality tasks such as testing is being reduced to make a deadline the alarm bells should be ringing loudly. Cutting out planned tasks to meet a deadline normally means a substandard product will be released to the users.
- **Lack of process used**. We all use processes in our life, such as driving to work each day. IT Projects are no different and if your project is not following a methodology or process then it is likely to go off the rails. People understand clearly defined processes and most adhere to these. Having defined processes in an IT project is key to them running smoothly, there should be processes for how the project is to be managed, how the development team are to work on one code base and testing processes to name a few.
- **Too much change**. Don't get me wrong change is a good thing, but sometimes we can either change too much at once or be constantly changing things that we have just implemented. Change is good, but only when is managed and controlled appropriately.

How to succeed!

IT projects can succeed and be very successful, so whats the key?

- **Ensure change is managed** - Change happens in most projects, if not all, but it's how you deal with the changes that can either make or break a project. Change management is needed regardless of what project management methodology you are using and needs to be controlled. Having change control processes, accountability, authorisation processes and principles, outlining when changes can be reviewed and introduced into the project, will help your team know what to do when a change is suggested. If you

have no processes or rules then change on your project could turn into a bun fight and deliver nothing or a solution that doesn't do what the business wanted because the vision was lost. You also need to remember that changes will have an impact on cost, people and the delivery schedule.

- **Do not let the project be micro managed by the boss!** - This is sometimes easier said than done and who is the boss on the project? Well, the answer kind of depends on what your role is. If you are the project manager then the boss is likely to be the project sponsor or customer(the person funding and driving the vision), but if you're a developer then this could be your lead developer, software architect or the project manager. So, this point actually applies to a number of people. If you are responsible for a team, then you need to remember that each person on the team has a purpose and role, otherwise why are they there? You need to have an overall understanding of how they are managing their time and tasks, but you need to also allow them to have some flexibility and creativity on how they achieve this. This also applies to the project manager, they will need information from the whole team on the progress, direction and problems that are currently happening, but I wouldn't expect a project manager to start rolling up their sleeves to carry out a code review. Micro management normally creeps in when someone is concerned or not getting the correct information about the teams or projects progress and therefore feel that to get an understanding must get more involved. If this is the case then, it is worth while sitting down with the person concerned and understand what the problem is and how you and your team can provide this information without them getting more hands on.
- **Don't rely solely on gut instinct** - I am sure most people have in their time on a project thought "hmmm, I am not sure about this", but can't figure out why? This is your gut instinct talking and although some of the time it is proved right, you shouldn't rely on it solely. If you are worried about some aspect of the project then this should be flagged as a concern and then proven or worked through with the team. This way the gut instinct has been formally logged and addressed. Thus, if someone asks about it later you or the project manager can show how the team came to your outcome.
- **Ensure the business is working with you**- Whether you are working on a traditional waterfall project or an agile one, it is key that the business is represented on the project *team*. For projects to be successful they need to meet a *business* need, so if the business is not working as part of the project team, how are you going to know what they want or whether is works as they expect? Remember your expertise is in IT not the business, so this needs to be represented.
- **Eradicate tunnel vision, see the BIG picture** - I think anyone can suffer from tunnel vision on a project, it is easy to get wrapped up in what you are working on and forget how it all fits together. However, for a project to succeed everyone must understand

what the project is trying to achieve, how it interacts with the current business and how the project is going to deliver. Understanding the bigger picture doesn't mean you need to know all the details of what each team member is working on, but you do need to understand how it fits together. One way that tunnel vision can be prevented is by the introduction of daily stand up meetings or 10 at 10's where the whole team meet for around 10 minutes to discuss what is happening for that day and any issues or help needed raised. Visual prompts around the room also help, such as a vision sketch showing the business requirements that need to be delivered or the architecture diagrams on how the system fits together. You'll be surprised by how many times your team will refer to these posters on the wall when talking to each other. Having the key information on the walls (flip charts or whiteboards will also do) makes it much quicker to refer to than finding the correct document and diagrams, plus they seem to get updated when things change.

- **Have transparency** - Nothing should be hidden from anyone in the team. Good communication is key to success. This actually goes further than the team that is implementing the change, it needs to cover the business and anyone who is affected by the change. By having transparency within the project usually means that problems are captured and dealt with earlier and it is a known fact that the earlier in an IT development a problem is found the less it will cost.
- **Outline how the project is to run and ensure everyone understands this**- This is something that the project manager will undertake, however it should be done in conjunction with the team. In the past, I have had some project managers say, "well we use Prince2 and thats how we are going to run the project" To me, this makes a huge assumption that everyone on the team understands the methodology and also the key elements to running an IT project is how the software development, configuration management and quality assurance of the solution is to be carried out and Prince2 does not cover these. The approach of how your team are to deliver the project needs to be understood and agreed by all. If you do not have your teams buy-in to how the project is to run then implementing the plan will be tricky.
- **Tell the TRUTH** - This goes in hand with having transparency. Telling someone bad news or something you know that they are not going to like is hard, however, not telling them early enough in order to do something about it is much worse. So many times on IT projects I have heard the words "We are CODE COMPLETE" but when you ask has it been unit tested and did it pass, I hear "Umm not yet", to me that is not code complete. Code complete is that is has passed the tests and it does what is meant to do and is ready for release for system testing.
- **Don't guess how long something is going to take** - Its pretty obvious really, but it is a situation that a lot of developers are put into. The problem with this is once a date is

given you are committed to it. It is better to say we don't know because xyz…, rather than put your finger in the air and guess.

- **Make sure estimates have contingency if there are unknowns** - If you have to give an estimate and you are not comfortable that you have enough information or certainty from your team then add contingency to the task to cater for unknown issues.
- **Question deadlines if they seem unrealistic** - I'm sure most people who have worked on an IT project have heard the comment "Surely that can't take that long?" or it has to be done by a certain date. First question is, where has the date come from? Most of the time you will find that someone has just decided that that's when they want it by, but there is no real business demand for a particular date. Sometimes, however a project can be date driven and if this seems unrealistic then, rather than try to squeeze as much in a possible, you should be addressing the unrealistic deadline and either reduce what is delivered or harder still, determine whether the project is actually viable.
- **Make sure non-coding tasks don't get missed off or cut** - When deadlines get closer in IT projects and development is still underway the non-coding tasks start to disappear or reduce in time. Coding and development is only part of an IT project and the quality assurance part (testing and making sure it does what it should) is key to the project being a success. Remember you only get one chance to introduce the new functionality to the user, make sure you and your team are proud and confident in what they release. If they are not, then you know it's not ready and you or your project manager is going to need to manage expectations.
- **Don't rush into coding, understand what the actual problem is first and what the customer wants** - You may think that by showing the customer what you can do through a prototype is great (and sometimes it is), but coding without understanding what the business problem is and what they need, could lead to delivering the wrong thing or creating unmanageable non-supportable code. Even agile projects need to have high level requirements before any sprints can be planned and coded.
- **Understand what you are coding by having the architecture defined**- Any developer will need to start building their code on a firm architecture, if this has not been defined and understood then you can end up with workarounds, spaghetti code or worse, the need to start again at a cost. The system architecture is driven from the requirements and needs to be in place and understood by the development team before coding and should be kept up to date throughout the development phase.

IT project management is about **control** and **vision** and the job of the project manager is to ensure that the project is controlled and all the team have a clear understanding of what the project is trying to achieve and how they are to get there i.e. the vision.

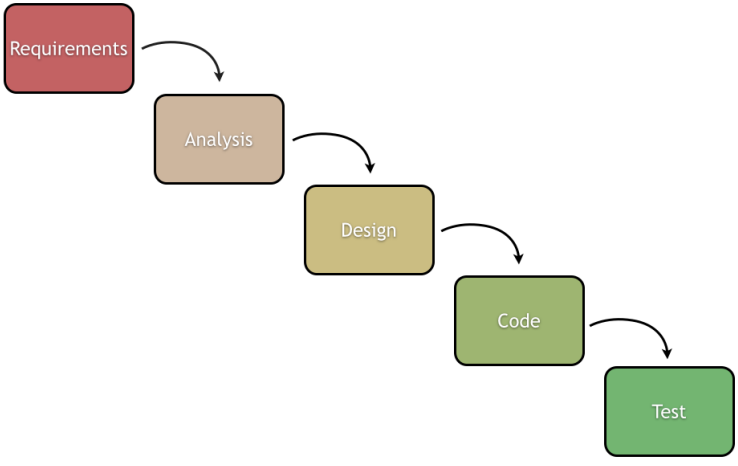
3. Methodologies

There are a number of different ways that an IT project can be run and they usually follow a methodology, which one depends on the type of project and also what is the standard within your company. Methodologies are in fact a set of defined processes and principles that a project follows in order to deliver something that the business needs.

There are three main methodologies used to run IT projects, although there are many more around based on these three.

The Waterfall Model

This used to be the most common method for delivering projects and was taken from the manufacturing sector. The principles within the waterfall model is that the project will move through each stage sequentially and the new stage cannot start until the end product from the earlier stage has been completed. This means that what the customer requested/wants will only be delivered at the final stage of the project.



Key points to note about the waterfall model

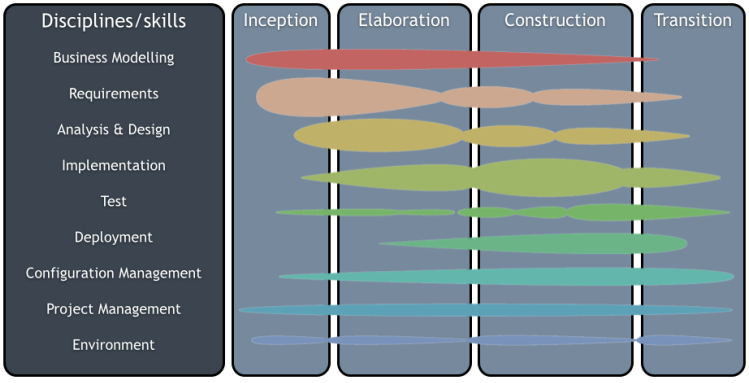
- It uses the big bang approach to delivering the solution to the customer.

- The customer is not involved all the stages.
- Each stage tends to work within their own silo and produce a stage output, such as a requirements document is the output of the analysis phase. This is then handed over to a team of software architects or designers for them to design the system from the collected requirements. This in turn, would produce a design blueprint that at the end of the stage is handed over to a development team who will then code the system from the design and so on.
- Once a stage has been completed it is common for that team to move onto a different project. This therefore provides additional pressure each stage to ensure that the outputs of the stage are well documented and correct.
- If at a later stage it becomes clear that a mistake was made in an earlier part of the project, it will be dealt with as a change request that will happen at the end of the project.
- Waterfall model is best suited for IT projects that are repeatable and are low risk, such as software or hardware upgrades. Within these types of projects there are known steps that the project needs execute, but also a few unknowns due to new hardware of software.

RUP Model

RUP takes a progressive approach in developing IT solutions, which has been shown in practice to be far more effective than the traditional, serial *waterfall* model. Although this method has a sequential element to it due to there being four distinct stages, it also takes advantage of iterative principles by delivering outputs via incremental releases throughout the project's life, whilst following proven best practices, such as Stop/Go decisions at the end of each stage.

The four phases RUP projects follow are:



- **Inception phase**- Within this phase the project's business case is defined and work is carried to determine if the project is worth doing or possible. It concentrates on what the project scope is, the requirements and puts plans in place for who is required and how the project will be run.
- **Elaboration phase** - This phase will specify the requirements in more detail and prove the architecture for the system. This is carried out by taking some of the higher risk requirements and doing analysis, design, build, testing on them. This is sometimes confused with prototyping, however the end products that are developed in this phase will be used within the end solution and therefore are not just proof of concept.
- **Construction phase**- The focus of the construction phase is to develop the system to the point where it is ready for deployment. If necessary, early releases of the system are deployed, either internally or externally, to obtain user feedback. It is common to have multiple iterations within this phase. This allows the customer to see the progression of the end product.
- **Transition Phase** - This phase is about delivering the system into production. It concentrates on system testing and user testing and any final adjustments are be based on user feedback, relating to usability or installation issues. Alongside this, training of end users, support, and operations staff is done.

Looking at the phases you may be wondering how RUP is different to waterfall; on the surface it looks like the only difference is less phases or stages. The power of RUP and where it differs is, that even though there are set phases, skills and people do not work in silos. Throughout each phase a number of skill sets will being working together even in the early inception stage and they are referred to as *the nine disciplines* of RUP:

- **Business Modelling** - The goal is to understand the business of the organisation, usually confined to the scope of the business that is relevant to the solution being developed.
- **Requirements** - To identify, document, and agree upon the scope of what is and what is not to be built. This information is used by analysts, designers, and developers to build the system, in addition by testers to verify the system, and by the project manager to plan and manage the project, so getting it clearly defined is key.
- **Analysis and Design** - To analyse the requirements and design a solution to be implemented based on the requirements, constraints and any relevant standards and guidelines.
- **Implementation** - To transform the design into executable code and to carry out a basic level of testing, in particular unit testing.
- **Test** - To perform an objective evaluation to ensure quality. This includes finding defects, validating that system works as designed, and verifying that the requirements have been met.
- **Deployment** - To plan for the delivery of the system and to execute the plan to make the system available to end users.
- **Configuration and Change Management** - To manage access to the projects outputs. This includes not only tracking versions over time, but also controlling and managing changes to them.
- **Project Management** - To direct the activities that take place on the project. This includes managing risks, directing people (assigning tasks, tracking progress, etc…), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget.
- **Environment** - To support the rest of the effort in terms in ensuring that the proper process, guidance (standards and guidelines), and tools (hardware, software, etc.) are available for the team as needed.

Key points to note about the RUP model

- It is does have sequential phases.
- It does follow defined processes, such as end of phase stop/go decisions.
- It groups work into smaller iterative work packages.
- It is collaborative. The team needs to work together and are involved for most, if not all, of the project.
- Documentation is still produced, but it is continuously improved when more is known at each phase.

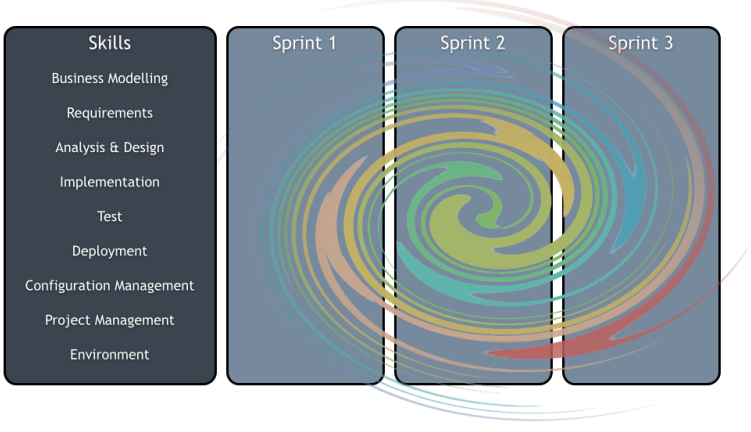
- It needs to incorporate the nine disciplines in each phase and iteration.
- By selecting the high risk requirements first means that if these prove to be too difficult then decisions on whether the project is viable or if more people are needed on the team can be made a lot earlier in the project.
- Most IT projects that say they use Waterfall, but have iterative deployments, should be using RUP instead.

The Agile model

Although the concept of agile has been around for some time now, it seems to have taken a while for companies to adopt it. One of the challenges that agile has faced is that it changes the way IT projects are delivered and also how project teams work together, therefore for some companies who are used to a more regimented approach agile seems high risk and a lot of change.

Agile has been seen as an approach that is organic and unstructured, but this is not true. Good agile projects do have structure and processes, but how the solution is planned and delivered is more flexible.

Looking at the agile diagram you will notice that agile projects do not have phases or stages assigned to them as standard. What agile has is sprints, what you do in each sprint is defined by the customer and what they believe is the most important elements the solution must deliver.



Like within the RUP model a good agile team needs to have expertise for all of the skill sets for the duration of the project, in agile projects the team are normally working on one project

at a time and can cover multiple roles if not fully utilised. In agile rather than separating out the analysis, design, build, test etc… these are done at the same time during a defined sprint and at the end of the sprint element of the solution will be ready to deliver into production for the business to test or start using.

Key points to note about the agile model

- Agile projects do have processes and principles.
- They are structured and they do have documentation.
- They include a number of sprints that include a certain amount of requirements.
- Sprints are time-boxed to typically 2 - 6 weeks.
- At the end of the sprint completed requirements are deployed for the users to use.
- Uncompleted requirements are put pack on the pile of remaining work to complete at a later date in the project, this is commonly called the backlog.
- Sprints should never be extended in duration in order to finish something.
- Sprints should be of a set time that is constant throughout the project.
- Planning and business requirement gathering is done before the sprints are planned.
- The customer is part of the project team and has an important role in deciding what requirements are to be delivered first and are involved in testing.
- Sprints are typically made up of 3 parts, *start* - planning what is to be in the sprint, *realise* - the implementation of the tasks and *evaluate* time to reflect on what was completed, what is outstanding, what went well and what did we learn.
- By having the customer involved and short deployments allows the team to respond effectively to change.