Agile

A radical revolution or a simple evolution

Mats Nyman
Wenell Management AB

During the last century, a number of agile methods have reinvigorated the field of system development. These methods are often described as a completely new way of thinking and working in projects. In reality, they are a collection of principles that has proven successful.

Do you have to abandon your current project model and completely embrace an agile method, or can you apply the agile principles in your current model?

In this article, the agile principles will be explained in relation to traditional project models.

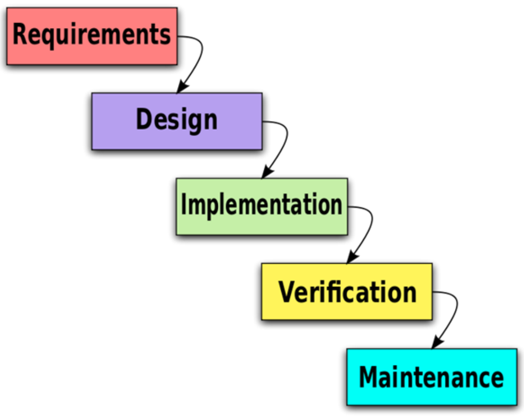
What are Agile Methods?

The word agile means to be “nimble” or “dexterous”. Agile Software Development (or “Agile”, for short) is a perspective shared by a group of agile methods. The fundamental idea behind Agile is that an ever-changing world demands methods for development that can handle change as a part of reality, rather than avoiding changes or trying to regulate them. More rules won’t result in more successful projects. We need flexibility – not rigidity.

In other words, agile methods can be considered an umbrella term. It’s not a method for system development but rather a collection of values, attitudes and principles. Within the agile methods, you’ll find several different methods for development, all of them considered agile. A method is considered to be agile when the author of the method recognizes the agile values.

History and Background

For many years, the development of IT has been based on different versions of the waterfall model, meaning that the point of entry when working on a project is to round up all the requirements that are relevant for the system. Following this, a solution is designed, and when this is done the solution can be developed and implemented. After implementing, you test and validate/verify. Lastly, the solution is launched and you devolve into maintenance.



There are many situations where agile methods can work better than the waterfall method. The issues that arise when you try to find and document all the requirements in advance, and then create specifications as a basis for the development, can be handled in a better way using agile methods.

This gets highlighted particularly in environments where demands are constantly changing.

Further, there are many cases where projects delivered in accordance with specification, and maybe even before deadline and within given resource restrictions, but where the results ended up being disappointing in regards to what the business needed.

The importance of being able to deliver partial results early on, which can be used to the advantage of the business far earlier than when the entire project has been completed, is an important motive for changing from a waterfall-like way of development.

During the 1990’s, several different ways of working were created as a reaction to waterfall-like methods, which were felt to be too slow and too cumbersome. Methods like Scrum and DSDM were developed independently and gained in popularity. These methods had a number of things in common. In particular, they had their foundation in experiences of what works and what the common causes behind failing IT projects were. Essentially, agile methods can be perceived as being common-sense packaged in different ways.

Something good did come out of these different initiatives. You usually see everyone fighting for “their” method, but instead, seventeen leading people in agile methods gathered in January of 2001 at a ski resort in Utah to discuss what they all could agree on and what set them apart. They ended up agreeing on certain principles and values. This perspective ended up being called Agile. During this meeting, the network Agile Alliance was created with the purpose of collectively developing and sharing the values, principles, and ways of doing things that are fundamental for every agile method.

The shared principles and values were documented in a common manifesto – The Agile Manifesto.

The Agile Manifesto

These bullet points are the pillars of the Agile Manifesto

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

“That is, while there is value in the items on the right, we value the items on the left more.”

The bullet points above should not be interpreted to mean that methods, processes, and tools lack value and that they should be ignored. Everything in the bullets above is important, but the words that are bolded are the most important.

At Wenell, we share these values and for many years we have been spreading the word regarding the importance of:

- Methods, processes, and tools are needed, but they are not enough. Motivated individuals and working communication, preferably face-to-face, are required to reach success.
- Prioritize. Make sure that you are getting useful results first and do reasonable amounts of documentation. Avoid value-lacking, comprehensive documentation.
- Assume that plans will change. Implement mechanisms that make change easier. Apply rolling wave planning and make it clear, using milestones, when different results should be ready.

- Collaborate continuously with the customer/user and receiver to make sure that the results of the project are in line with customer expectations. Focus on creating customer benefits by clarifying the business objectives and leading the project with those in mind.

The Agile Principles

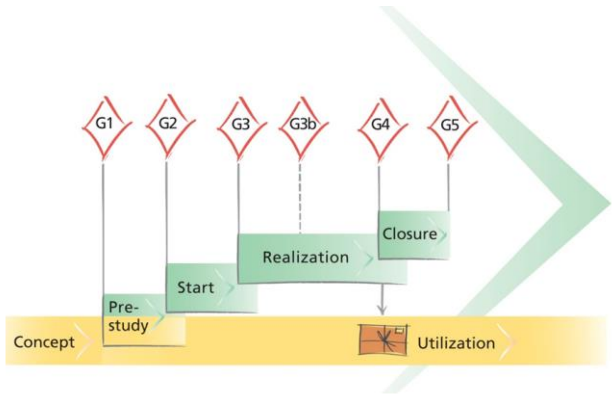
The agile principles should be present in all projects where the intent is to work agile. In the Agile Manifesto and in the original statement of the agile principles, the starting point was on software projects, and they talked in terms originating from system development. However, the principles are universal and applicable in many different kinds of projects, albeit not every project. Below, you'll find our interpretation of the agile principles that also work in projects that are not related to system development.

- Our highest priority is to satisfy the customer through **early and continuous delivery** of valuable project results (software).
- **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
- **Deliver** results (working software) **frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- **Business people and developers must work together** daily throughout the project.
- Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
- **Working results** (software) is the primary measure of progress.
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to **technical excellence** and good design enhances agility.
- **Simplicity** - the art of maximizing the amount of work not done - is essential.
- The best architectures, requirements, and designs emerge from **self-organizing teams**.
- At regular intervals, the **team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly**.

From now on we will be discussing how the agile principles can be inspiring, even if you're using a traditional project model as a foundation.

Agile Application of a Generic Project Model

A model is a way of simplifying a complex reality. In our opinion, a project model must be adapted and applied in the way that best supports a certain business as well as the individual project. Let's talk about how one can apply and combine what's good in traditional project models with the sound and easy to understand principles that form the basis for agile project management.



The figure illustrates the project flow in Wenell's project model and shows in a transparent way the course of the project, from concept to utilization. Similar descriptions are found in most established project models.

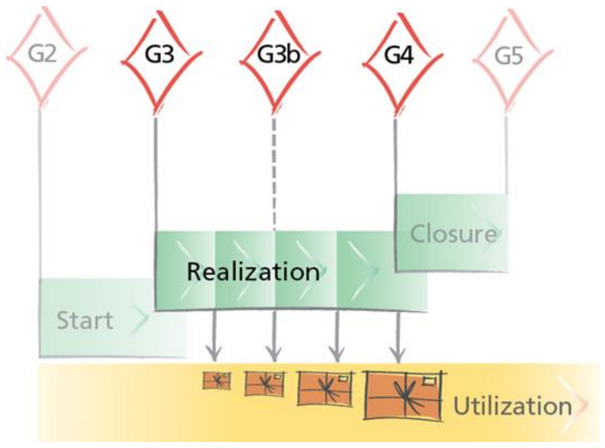
The project life cycle consists of a number of phases and decision points, so called gates. The phases contain different elements of work that are to be performed and the gates are forward-looking decision points where business benefit, available resource, etc. is assessed. From experience, this division of the workflow enables a more measured structure for the project and creates higher quality results with the work focused on the most important activities. As the image shows, the following phase is often started before the previous one is completely finished.

The idea behind this division into phases is to, during the early phases of Pre-study and Start, in a systematic way, promote knowledge about the project, illustrate the project from a business perspective, and create a sound basis from which to decide if it's a good idea to progress further with the project or not. This fundamental idea is valid even when you apply agile project management.

However, in certain kinds of projects, it is very hard early on to specify in detail what needs to be accomplished and delivered, which is an important insight that characterizes agile methods. The uncertainty is often substantial and it is almost impossible to describe clear objectives and exclusions before you've reached a certain point in the project. From experience, it is also clear that in projects where choice were made to ignore this insight, there is a tendency for the projects to consume considerably more time and resource than initially planned.

Many of the agile principles address this problem, and one way of dealing with this uncertainty is to divide the realization phase into several shorter parts to deliver concrete, usable results early and often. These different parts are called different things in different agile models. A couple of examples are Iteration, Increment, Timebox, and Sprint. We have chosen to call it an Increment. A regular recommendation for the duration of an Increment is about a month or less.

The Increment starts with Increment planning, where we define the contents in terms of prioritized functionality/deliverable results, time estimates, etc. During the Increment, the team gets to work without interruptions and with high intensity focus. At the end of the Increment, delivery is prepared for by having a well-tested and quality-assured result that can be demonstrated and reviewed. In conjunction with this demonstration/delivery, it's very clear what the project has accomplished and other forms of status rapports become redundant. At the end of the Increment, the team will conduct a Retrospective, where the team discusses what has worked well and what can be improved. This discussion results in continuous improvement of the way the team works.



Dividing a long realization phase in several shorter parts, Sprints, gives many benefits:

- The group gets clear results to aim for along the way, giving clarity and focus.
- The group continuously delivers parts of the end result, benefitting the business far earlier than a single big delivery at the end.
- The customer/user continuously receives deliveries, utilizes the results and can benefit from this early on.
- The customer/user gives feedback to the project team, which continuously gets input for adjustments and improvements for upcoming deliveries.
- Progress and status in the project are very clear since the delivered results are shown and these demonstrations can mostly replace normal status updates.
- In conjunction with every delivery, there are possibilities to reflect on the work of the team and develop proposals for improvements.

Stakeholders

Since an important agile principle is to deliver benefits, it's wise to early on strive towards clarifying what this benefit looks like. An analysis of stakeholders is needed in agile projects as well, since the project needs knowledge regarding individuals, groups, and organizations, and their important insights and opinions about what the benefit means to them. In the early phases, you should create a mutual understanding regarding this within the development team. The work in clarifying the stakeholders and their expectations are led by the product owner/person in charge of benefits (see the organization below) and this role is tasked with the responsibility of clarifying which of the expectations the team needs to satisfy by priority.

During implementation, this role is responsible for defining the results that are to be developed in every increment. Understanding that expectations and needs change, and that they will get clarified during the project, should influence the priorities. When the project is finishing up, you will have results that give the most benefit in relation to time and resources spent.

Defining Objectives – The Project Arena

In the early phases, it's important to try to clarify project objectives. The Project Arena consists of a summary of both the intended effects of the project and the result that is to be delivered when the project is finished. The exclusions also play an important part in the arena, as they clarify the results that are not to be delivered. Time and cost limits complete the goal setting, giving us a feel for when the project should be done and what resources it will consume.

When working agile, we must realize that the goal definition is founded on the knowledge we have about the project during the early phases. This knowledge will develop and be refined during the course of the project. We must be prepared to use the new knowledge and readjust our objectives for them to mirror the reality of the project and the demands and expectations of the project environment.

One way of handling this balancing act is to define the overarching objectives of the project at a less detailed level than normal, and then later, per Increment, prioritizing harshly to get the most important results for delivery, giving the highest possible value and benefit.

There are many different techniques in the different agile methods when prioritizing. One of the good and usable ones is called MoSCoW and stems from DSDM/ATERN. MoSCoW can be used to prioritize the importance of different deliverables for the entire project and primarily when prioritizing in different Increments.

The letters M, S and C in the word **MoSCoW** stand for different levels of prioritizing.

- **M**ust Have - these provide the Minimum Usable Subset (MUS) of requirements which the project guarantees to deliver. It is no point in delivering on target date without this; if it were not delivered, there would be no point deploying the solution on the intended date. Not legal without it. Unsafe without it.
- **S**hould Have - also important requirements. Important but not vital. May be painful to leave out, but the solution is still viable. May need some kind of work-around, e.g. management of expectations, some inefficiency, an existing solution, paperwork, etc.
- **C**ould Have's - Wanted or desirable but less important. Less impact if left out (compared with a Should Have).

It is the recommendation of DSDM/Atern that no more than 60% of the development initiatives in one increment should be defined as Must Have.

W stands for **W**on't Have and can be used as exclusions, clarifying what should not be included in a specific delivery. The two occurrences of the letter o can be viewed as markings that show that the increment is controlled by time and resources (**o**n time and **o**n budget).

To sum it up, you should aspire to have clear objectives for the entire project, but at the same time realize that your knowledge is limited about what's to come. When working agile, it's of the highest importance that we are prioritizing actively, where we can benefit from increasing knowledge and changing circumstances. This means that we know when to deliver and what resources we have available, but the exact content of the delivery remains dynamic. The fundamental idea behind this approach is that we get the highest possible benefit in the delivery in relation to time and resources consumed.

Visualization

Visualization is a way to create understanding for what you want to accomplish. There are many ways of visualizing, including everything from different kinds of tree structures and simple sketches to models and prototypes. It's important to choose a visualization method with the need of the project in mind. Every project benefit from visualizing, not only agile projects.

"User stories" and "storyboarding" are two techniques used to describe what the results of a project should be used for.

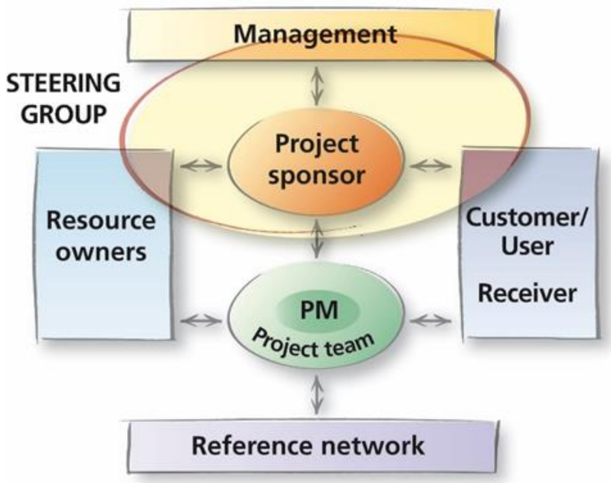
An important principle in agile methods is to visualize status and to identify possible obstacles early. A usual way of doing this is by using a task board. If we imagine that we've broken down the delivery of an increment in descriptions as user stories, each one of them will get one row on the task board. On the row we can see the activities that belong to the corresponding user story written on post-it notes. On the board, we also have columns for "To do", "Started", "Ready for verification", and "Done". As work progresses, you move the notes, giving you an up-to-date visual of the status of the work.

There are other versions of this, but the principle is to make it clear what you are going to do, what you are working on right now, what's done etc.

It's also possible to supplement with a graph, where the Y-axis, for example, symbolizes total amount of man hours that the team has undertaken at the start of the increment. The X-axis shows the calendar days of the increment. Every day, you plot the estimated remaining development hours. This helps in showing how your work is progressing, and if you're risking not having time for some user stories and need to take action.

Organization

At Wenell, we use this figure to illustrate interaction and cooperation in the project. We use it as a kind of checklist in order to clarify different roles important to the project core organization and in the project environment.



The idea is to give everyone a good and clear image of which individuals are filling the different roles in this particular project. Visualization, clarity, and transparency are important in every project.

In order to make agile work possible, an extremely important role is played by the individual who represents the benefit perspective and possesses deep knowledge about how the results should be utilized and the expected benefit of the project. In Scrum, this role is called the Product Owner. In DSDM/Atern, there are more roles with an order of escalation, but the primary role for prioritizing is called the Business Ambassador.

It's important to define responsibility and authority for this role, and to clarify who has it. One of the most important reasons why an agile working method isn't working out is when this role isn't clearly defined, and when the individual responsible for it isn't present or available.

Another important principle in an agile organization is to enable commitment, focus, and a healthy working environment. In order to make this possible, you should aim for a smaller team (about 5-9 people) where everyone has a high level of availability in the project. We aim to fill the project with people who can work on it full time, and whose attention is not split between several different projects. From experience, this is the most efficient way to work and you get the most benefit per working hour if you can work like this. An important task for the project manager/team leader is to make sure that the members of the team can devote themselves and remove other obstacles from the project.

In agile methods, it's important that the members of the team are active in planning, that they are allocating their own time, and that they commit to tasks and are responsible for the different results, rather than having a project manager controlling who is doing what and when. The co-workers on the team are considered responsible individuals and act as such when given the correct conditions and support. “One for all, all for one” should be the motto for a working agile team. Collectively, you are responsible for structuring the work in a manner where you can have a delivery at the end of every increment. This approach works hand in hand with theories and research on group dynamics and motivation.

Planning

Time estimates and planning are very important in most development projects. Without a good plan, we’re exposing the project to a host of different problems. Still, planning is hard and plans are often wrong. Project teams tend to react to this fact by choosing one of two extremes. Either you don’t plan at all, or you choose to spend so much time and effort planning that you manage to convince yourself that the plans are correct.

Planning is all about reducing risk. The biggest risk for many projects is to develop the wrong products/results. The definition of a successful project is often seen as the ability to deliver on time, within given resource limits, and where all features/results are included in the final delivery. However, this is a dangerous definition, as it doesn’t take into account the fact that a feature that seemed relevant when the project began might not be worth the effort it takes to develop once we gain knowledge of all the details. In agile estimation and planning, we continuously reconsider, utilizing increasing knowledge we gain along the way. A customer or user will probably not be very happy to know that fantastic ideas or new smart features are rejected in favor of mediocre features just because these were included in the original plan.

In agile projects, as we’ve mentioned earlier, we aim for early, successive deliveries with prioritized content. This will create trust between project members and the customers/users that continuously receives new deliveries with results they can start using and gain benefits from.

A rather big difference in planning an agile project compared to more traditional projects is that the planning in the agile project is done on a broader level, initially. At the start of the project, we realize that the knowledge about the project is low, and that it will increase throughout the duration of the project, and that there will be lots of changes and clarifications along the way. Trying to plan in detail from the start is, simply put, not very efficient. Also, it’s a way for many projects to fool themselves. Just because the plan is detailed doesn’t mean that it’s correct. In other words, we are aiming to plan continuously during the entire project. The most important part is the planning, not the plan. The knowledge we gain during planning can last well beyond where the current plan has become obsolete and been replaced by another plan.

The broader planning can be done using traditional techniques like breakdown structures, logical network diagrams, and Gantt charts. The important thing is to work broader and not to try to, for example, estimate in detail how much time something will take when we don’t have sufficient knowledge. Planning using milestones representing achievements along the way works very well and mirrors the thought of early and successive deliveries.

The detail should be handled and planned at the start of every increment. The team and the product owner gather to process the prioritized list of requirements and define the contents in the upcoming increment. The product owner will prioritize the content in the delivery, and the team is responsible for time estimations and the scope of the prioritized list of demands that they are taking on for development during the term of the increment.

Used in some agile methods, planning poker is a popular way of doing estimations. Simply put, every result that's supposed to be included in the delivery of the increment is described in a clear and understandable way, e.g. a user story (As a <role>, I want <goal/desire> so that <benefit>). The team is gathered and every member is given a set of cards with different estimates printed on them (e.g. 0, 1, 2, 3, 5, 8, 13, 20. 40 and 100). The unit can be man hours, or "story points" etc. Every member of the team makes their own estimation of how much effort certain results will take to develop. When everyone has chosen their cards, you display your estimates and compare. If there's a big difference between the lowest and the highest estimation, those who have made these estimations get to explain their line of thinking. These clarifications are then discussed and afterwards, everyone makes a new estimation. You do this until everyone can unite behind a reasonable estimation, acceptable by everyone. From experience, this works because the estimation gets done by individuals with different expertise. Further, this is an example of a technique that focuses on learning and collective understanding. The planning is more important than the plan! And it's fun!

To sum it up, agile planning should be characterized by:

- Focusing on the planning rather than the plan.
- Embracing change - if we come to the conclusion that it's smart and creates an increased benefit for the customer.
- Aiming for plans that are easy to change.
- Project members being active in the planning, and that they are independently making time estimates, and committing to perform tasks.

Dealing with uncertainty

The ability to live with uncertainty and to handle it is built into the agile perspective. "Embrace Change" is an often-used motto. The thought is that we learn more and more during the project, and the more knowledge we have, the wiser the decisions we can make. In order to give users the most valuable results possible, it's important to have an attitude where you continuously see, and even embrace, change. This is made possible through the product owner's priorities and feelings for what is most important and what creates the most value/benefit. If new requirements and expectations arise in the project, it is possible to add these to an increment ahead of us, and of course, other content deemed less important can be excluded from the project or the increment (see MoSCoW).

It has great value and it creates knowledge about the critical success factors for a project if you analyze the risks and possibilities in the early phases. Mini-risk is a simple, common technique for performing these analyses with associated action plans.

Another mechanism that most agile models advocate is a daily, short stand-up meeting. The technique is simple and very rewarding. The entire team gathers in a U-formation so that everyone has eye contact with each other. Everyone in the team answer the following three questions:

1. What have I done since the last meeting?
2. What am I doing until the next meeting?
3. Do I/we have any obstacles/problems that I/we will have to deal with?

The meeting is tightly timed and shouldn't take more than 15 minutes. If there are issues that arise during the meeting that need to be discussed/dealt with, the concerned individuals will have to do that after the meeting. During the meeting, the status is updated, for example on an activity board, making it clear what we've finished, what we're working right now, and what's yet to be started. Further, you can graphically visualize how worked hours are relating to the created results. This will make everything visible and understandable.

By applying a meeting agenda like this, the team is aware if there are any problems or if something is about to derail.

Project Realization

An obvious difference between traditional projects and agile projects is the division in many shorter phases (increments), where we're creating a small part of the total delivery in every phase. When the Increment is done, the result is delivered, demonstrated to stakeholders, and can start being utilized by the users. The users give feedback to the team who absorb it and benefits from it during the upcoming Increments. This way, it's easy to spot if misunderstandings have arisen, etc. The risk of delivering a lot of unusable or faulty parts at the end of the project is reduced radically.

In the coming Increments, you continue to build on the earlier results and continuously deliver more parts of the total result. Apart from the feedback telling us that we're on the right track, an important benefit to working like this is that the users can start utilizing the results way earlier. The business gets to enjoy the benefits before the entire project is done. Also, you constantly have the possibility to discontinue the project and yet still have big parts of it finished and usable.

As we've mentioned earlier, an important part of the work of the team is to constantly improve. In conjunction with every delivery, you can sit down together and talk about how the work is progressing, "sharpen your tools", thus reaching new realizations that can be used to adapt the work of the group and improve efficiency.

One way of easily implementing a reflective meeting like this is to make everyone on the team think about their experiences of the last increment. For example, you can write what has worked well on green post-it notes and things you've experienced as not so good on red notes. Every team member attaches their notes on a timeline of the last increment and gives a short summary of their experiences. The project manager/team leader facilitates this and sums up the observations of the team on a list, detailing what needs to be improved in the next increment.

Project Closure

When the project has delivered its last increment, it's wise to finish the project in a structured manner. Summing up the project and talking about the experiences with relevant parties are two important activities at the end of the project. However, most of the work related to summing up experiences and "lessons learned" has already been done in agile projects as you carry out smaller deliveries and retrospectives for every increment.

Further, we want to finish up some formalities in a tidy fashion, like closing temporary accounts, time tracking codes, etc. Dissolving the organization, giving each other feedback and celebrating the successful results are other important activities.

Another thing that's a bit different is the measurement of project objectives. These were broadly defined from the start, and might have changed several times during an agile project.

In conjunction with partial deliveries after every increment, the act of handing over the results to different receiving roles has already been done a couple of times during the project. Thus, the formal ending in a well conducted agile project is easier than the one big delivery associated with more traditional projects.

Lastly

The agile principles are based on “best practice”, principles that, from experience, have proven to work well, especially in projects pertaining to system development. However, the principles are very sound and should be able to be utilized in many different kinds of project types. Every project might not be able to use every principle, but some parts are always applicable.

The answer to the question whether agile methods are a radical revolution, or a simple evolution, will vary depending on how used you are to managing projects and your own leadership style. Many experienced project managers work in line with agile principles as a natural way of working although they do not explicitly say they work Agile.

Think about these principles and imagine which of them could create benefits in your projects. I have a firm belief that this common-sense approach is applicable in many different projects.

Good luck!

A Couple of Agile Links

www.agilemanifesto.org
about the Agile Manifesto

www.dsdm.org
about DSDM/Atern

www.scrumguides.org
about Scrum

www.extremeprogramming.org
about XP

www.poppendieck.com
about Lean Software Development