**Agile means “no planning.”**Detailed planning is as essential to the effectiveness of Agile as it is to Waterfall. The difference between the two approaches lies in timing. Planning is ongoing in Agile versus taking place primarily at the project outset in Waterfall.

**Myth 3 – Agile gives instant benefit** While a transformation to agile can deliver huge benefits, the reality is that the majority of transformations go through a learning curve.

**Myth 4 – Agile means no documentation** focus on producing working software instead of spending exhaustive amounts of time creating detailed documentation up front.

**Myth 8 – Agile only relates to software delivery** It is true that the *Agile Manifesto* describes agile in the context of software delivery, but agile can be applied successfully in business environments that are not exclusively software-related. In essence, agile is suited to any dynamic business environment that experiences variability, such as marketing or business change.

**Myth 9 – Agile should replace everything all at once in a big bang transformation**When agile is applied in a big-bang approach across large projects, programmes or across the entire organisation, there is a significant risk that the benefits of an agile operating model will not be realised or understood. Often the organisation and its staff will simply continue to do things as they have always done them while pretending – or believing – that they have moved to an agile method.

**Myth 10 – Agile means no planning, just do it**

The vast majority of agile frameworks involve frequent, regular and evolutionary planning. If a team is largely doing maintenance work or defect fixing, or there is no need for the customer to look any further than a couple of weeks when creating a product, they will probably only be planning in single iterations/sprints.

**C1: Project definition**: A project is a sequence of unique, complex and connected activities that have one goal or purpose and that must be completed by a specific time, within budget, and according to specification. **BAU definition:** Non-unique and permanently ongoing activity that does not provide new value. **Key:** specific time and cost.

**C3:** Leadership is a process whereby an individual influences a group of individuals to achieve a common goal.

Management is to exercise executive, administrative, and supervisory direction of a group or organization.

|  |  |  |
| --- | --- | --- |
| Category | Leadership | Management |
| Thinking Process | Focuses on people Looks outward | Focuses on things Looks inward |
| Goal Setting | Articulates a vision Creates the future Sees the forest | Executes plans Improves the present Sees the trees |
| Employee Relations | Empowers Colleagues Trusts and develops | Controls Subordinates Directs and coordinates |
| Operation | Does the right things Creates change Serves subordinates | Does things right Manages change Serves superordinates |
| Governance | Uses influence Uses conflict Act decisively | Uses authority Avoids conflict Acts responsibly |

**Benefits of managing projects: 1**.**Better Efficiency in Delivering Services**: Project management provides a “roadmap" that is easily followed and leads to [project completion](http://www.brighthubpm.com/project-planning/26374-smart-goals-in-project-management/). Once you know where to avoid the bumps and potholes, it stands to reason that you’re going to be working smarter and not harder and longer.

**2. Improved / Increased / Enhanced Customer Satisfaction**: Whenever you get a project done on time and under budget, the client walks away happy. And a happy client is one you’ll see again. Smart project management provides the tools that enable this client/manager relationship to continue.

**3. Enhanced Effectiveness in Delivering Services**: The same strategies that allowed you to successfully complete one project will serve you many times over.

**4. Improved Growth and Development Within your Team**: Positive results not only command respect but more often than not inspire your team to continue to look for ways to perform more efficiently.

**5. Greater Standing and Competitive Edge**: This is not only a good benefit of project management within the workplace but outside of it as well; word travels fast and there is nothing like superior performance to secure your place in the marketplace.

**6. Opportunities to Expand your Services**: A by-product of greater standing. Great performance leads to more opportunities to succeed.

**7.** **Better Flexibility**: Perhaps one of the greatest benefits of project management is that it allows for flexibility. Sure project management allows you to map out the strategy you want to take see your project completed. But the beauty of such organization is that if you discover a smarter direction to take, you can take it. For many small-to-midsize companies, this alone is worth the price of admission.

**8. Increased Risk Assessment**: When all the players are lined up and your strategy is in place potential risks will jump out and slap you in the face. And that’s the way it should be. Project management provides a red flag at the right time: before you start working on project completion.

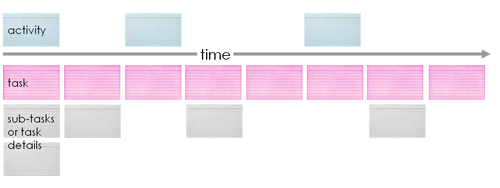
**9. Increase in Quality**: Goes hand-in-hand with enhanced effectiveness.

**10. Increase in Quantity**: I saved the best for last. An increase in quantity is often the result of better efficiency, a simple reminder regarding the benefits of project management.

**C4 C5: Benefits of User Story Maps：**

* allow you to see the big picture in your backlog,
* provide a better tool for making decisions about grooming and prioritizing the backlog
* promote silent brainstorming and a collaborative approach to generating user stories
* encourage an iterative development approach where early deliveries validate your architecture and solution
* are a great visual alternative to traditional project plans
* are a useful model for discussing and managing scope
* allow you to add a visual dimension to planning and real options for your project/product

**Creating a User Story Map**



To construct a User Story Map you

* Place activities – big user stories (UX), epics (Mike Cohn) – at the top

An activity is something people do –

has lots of steps, and

doesn't always have a precise workflow

* Eg. Building an email system

managing email

configuring email servers

Setting up out of office responses

* Activities provide the context
* Break activities down into user tasks i.e. smaller stories

Eg. send message, read message, delete message

A user task is something that someone does to reach a goal

* Move from left to right and top to bottom
* Arrange stories in the logical order in which they would be completed – grid form
* Test your map

Talk / walk it through

Find things you have missed

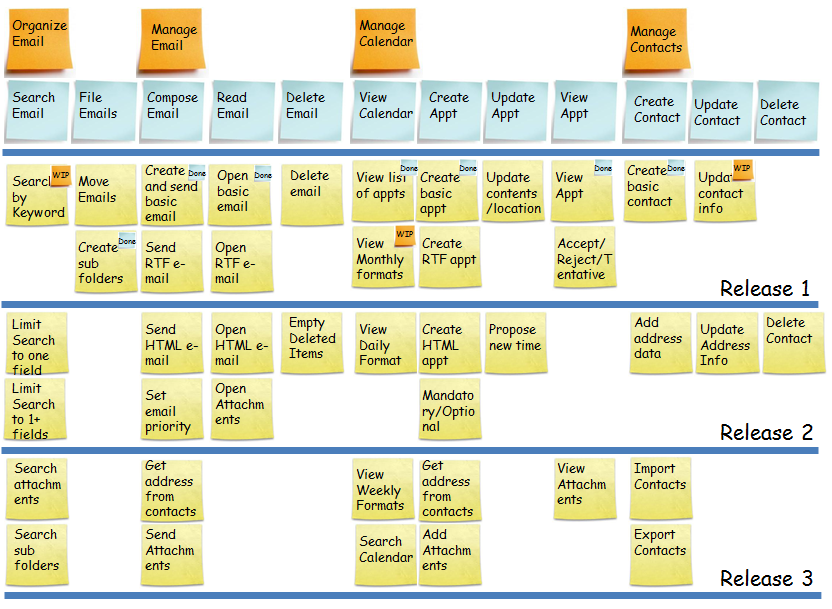
* Annotate it
* Use different colour for different levels
* Display it as an information radiator

Use for sprint or iteration planning

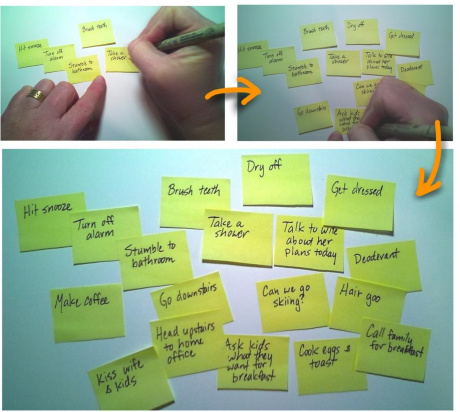
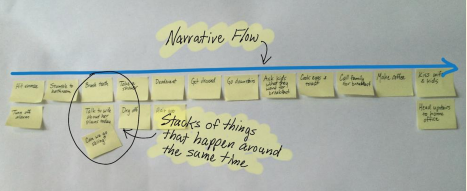
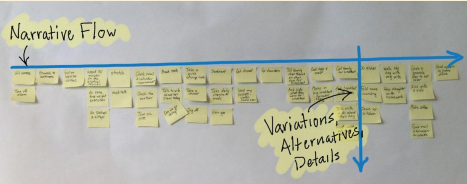
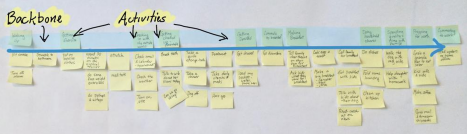
Mark of progress

Anatomy of a User Story Map

**The following image shows a possible User Story Map for the creation of a new email product.**

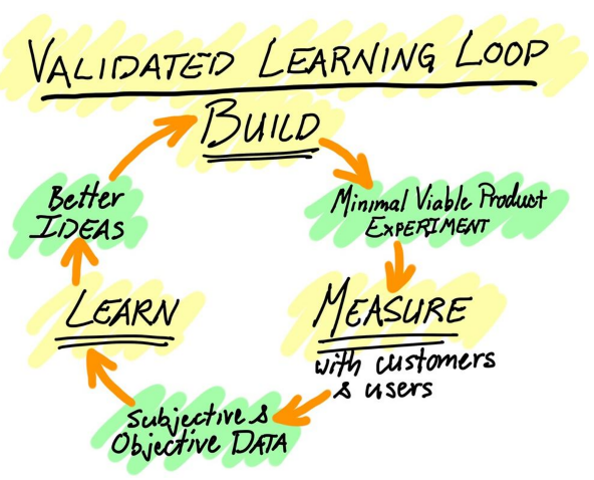


Example exercise for class - Getting to Uni

* Quickly note down what you did this morning when you woke up and got ready to come into uni. This might be, for example, turn off the alarm, take a shower, get dressed, have breakfast, catch the bus to uni.  
  Some may have more detail than others - don't worry about that at this stage. For example instead of make breakfast you might have said put bread in the toaster and pour a glass of juice as two separate tasks, while the person next to you may have said make breakfast. These are sub-functional or sub-tasks.  
  
* Decide which tasks you would complete before you'd intentionally stop and do something else, for example, take a shower. These are the functional level or user tasks. They can be broken down into small subtasks or user stories.   
  Think of tasks being like rocks. You can hit a big rock with a hammer and it will break into a bunch of smaller ones.   
  Tasks are the same. Frequently they can be broken down into a number of different size pieces
* Group sub-tasks with any associated tasks. For example, when taking a shower you might have included: adjust water temperature, wash hair, wash. Tasks can be aggregated into summary tasks. For example, everything you do in the morning in the bathroom might be called the "morning bath routine". These are goal level or summary tasks.  
  Goal level concepts help you aggregate small tasks or decompose larger tasks. Goal level tasks are often called epics - they are very large stories.  
  
* Now start creating the map  
  Organise your tasks from left to right with what you'd logically do first on the left. This allows us to develop what is known as "Narrative Flow", this represents the order in which you'd tell the story.  
  The visual nature of the story map - constructed using sticky notes - let you see the whole big picture. It helps you see the bits you have forgotten.  
  
* Fill in any missing bit and then explore alternatives.  
  What did you do yesterday morning? What do you do at weekends, is it different? What happens when something goes wrong, say you sleep through your alarm or you don't have anything for breakfast? What about your ideal morning? Add these tasks to the map. Is there anything more you need to add, anything you have missed?  
  
* Now still the map to make the backbone. You'll note you probably have bunches of stories that go together. Group these, because together they help you achieve a bigger goal. Identify that goal with a verb phrase. These are call activities or summary tasks. These form the backbone of your story map -- they tell the story of your product and so have "narrative flow".  
  
* Now think about what outcomes you need to reach. These must be specific. For example, think about something that didn't happen - you forgot to set the alarm! So add the story "get out the door in a few minutes". Think of slicing through your map left to right, move tasks below that line if they are something you wouldn't do when you need to get out the door in a few minutes. Likely you will only have a few stories left above the line. Add any missing tasks you would need to do if you were running late.  
  Use these "slices" to identify all the tasks and details relevant to a specific outcome.



**C6: Ready and done: ‘**ready’ is a set of rules that the team adopts as a guide for a story to be moved reasonably from the backlog to an agile project and ‘done’ is accepted by the team as a guide to ensure that user stories are done and wait for approval of the project manager.

**C7:**

**Test MVP:**

1.Communication with users

2.The page of login

3.A/B test, that is to design two different format for users and receive feedback.

4.Launch ads

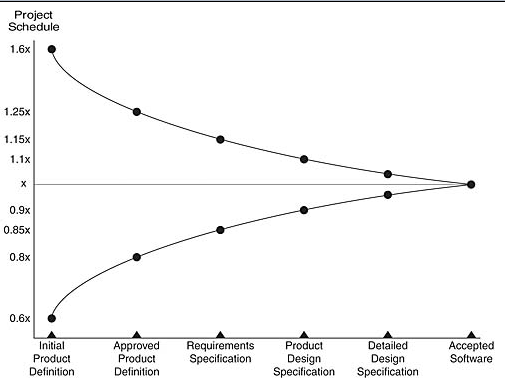
5.Raise fund

6.The introduction video for products

7.Based on the existing resource.

8.SaaS & PaaS, which means not to throw into too much at the beginning.

**C8:**

****

1.Imagine you've completed 3 iterations with velocities of 12, 15 and 16.

2.Calculate the average velcity for the iterations you have run, in this case, it is 14.3. Round it up to 15. (NEVER round down!)

3.For each completed iteration, move one step to the right on the cone of uncertainty, to a maximum of 4 steps (Product Design Specification). To calculate the range, multiply the average by the number on the y axis representing the step.

e.g.: Essentially if number of story points = a, velocity = b and iteration length = c then the formula is a / b \* c = number of weeks.

‘Calculate the number of iterations

100 / 15 = 6.6 so 7 iterations

**Planning poker:**

**1.**A Moderator, who will not play, chairs the meeting.

2.The Product Manager provides a short overview. The team is given an opportunity to ask questions and discuss to clarify assumptions and risks. A summary of the discussion is recorded by the Project Manager.

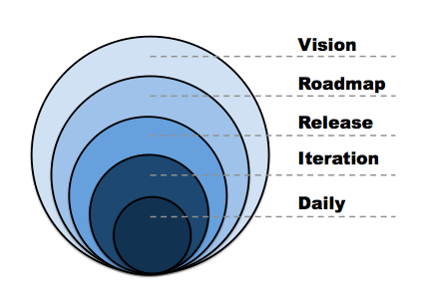
3.Each individual lays a card face down representing their estimate. Units used vary - they can be days duration, ideal days or [story points](https://en.wikipedia.org/w/index.php?title=Story_points&action=edit&redlink=1). During discussion, numbers must not be mentioned at all in relation to feature size to avoid [anchoring](https://en.wikipedia.org/wiki/Anchoring).

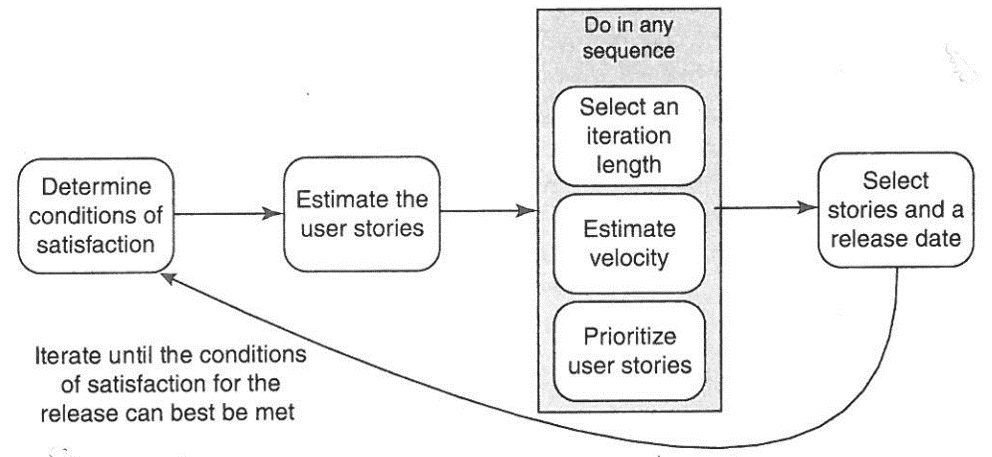
4.Everyone calls their cards simultaneously by turning them over.

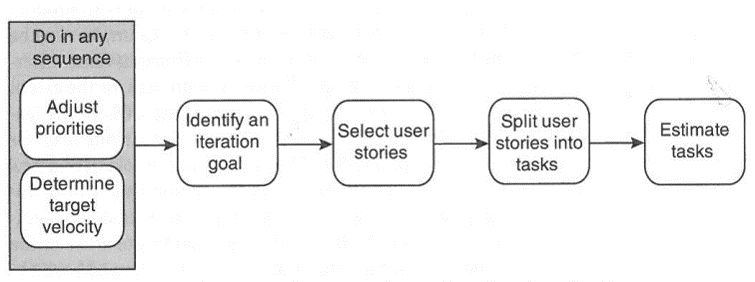
5.People with high estimates and low estimates are given a [soap box](https://en.wikipedia.org/wiki/Soapbox) to offer their justification for their estimate and then discussion continues.

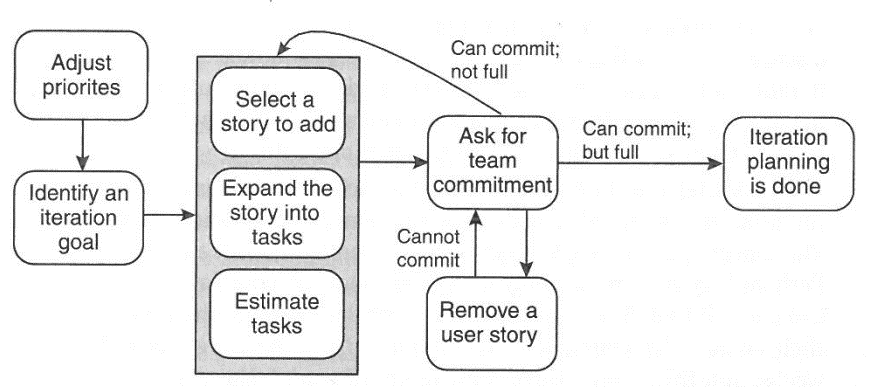
6.Repeat the estimation process until a consensus is reached. The developer who was likely to own the deliverable has a large portion of the "consensus vote", although the Moderator can negotiate the consensus.

7.To ensure that discussion is structured; the Moderator or the Project Manager may at any point turn over the egg timer and when it runs out all discussion must cease and another round of poker is played. The structure in the conversation is re-introduced by the soap boxes.

****

**C9: steps in planning a release**

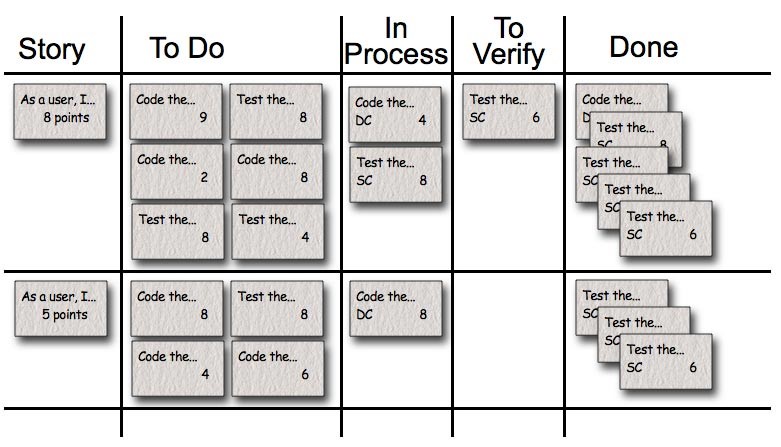
****

****

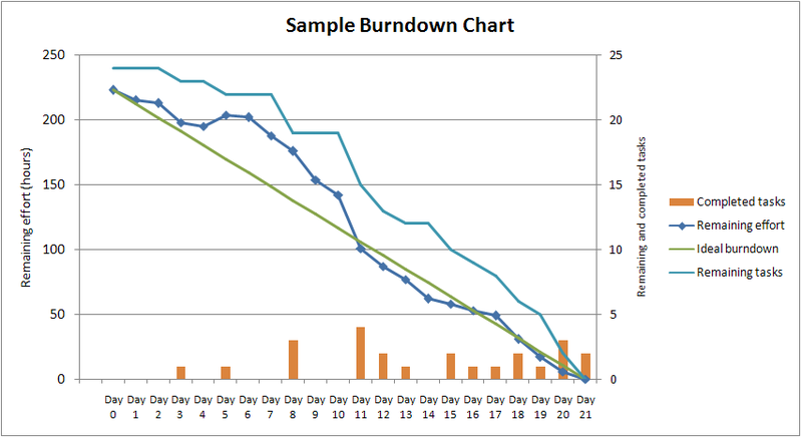
Metrics are what we collect as part of our monitoring and tracking activities.

**Tracking and Reporting Progress: Task Boards**

A "Task Board" is used to organise the work and shows at a glance how much work is left. The board must allow flexibility. For example, work isn’t assigned until the individual is read to start on it so until the end of the iteration, there may be many tasks unallocated. The board makes these tasks highly visible.



**Burndown charts**



Your progress report needs to summarise:

Schedule – original dates, authorised changes, and current estimated completion

Budget – original approved, authorised changes and current estimated budget requirements

Issues – any issues or risks triggered which have caused changes to scope, schedule, budget, quality or functionality

**C10 Core Governance Principles 2**

1.Single point of accountability for project success

2.Service delivery ownership determines project ownership

3.Separate stakeholder management and project decision-making

4.Separate project governance and organisational governance

**The Three Pillars of Governance**

Structure

Determines how the decision-making is organised (e.g. decisions about investment, business change, timing etc.)

People

Roles, skills, authorities, recognition, influence, commitment

Information

Lines of reporting, accountability, communication

**Key Governance Roles**

**Establish** the basis for project governance (roles accountabilities, policies, standards, management/reporting processes)

**Evaluate** project proposals

**Enable** projects through resourcing and harnessing business support

**Define** the desired business outcomes, benefits, and value from the projectKey Governance Roles (cont.)

**Control** scope, contingency funds, overall project value

**Monitor** progress, commitment, results

**Measure** outputs, outcomes, benefits & value

**Steer** the project, remove obstacles, managed critical success factors, remedy shortfalls

**Develop** the organisation’s project delivery capability

**What can go wrong**

Governance:

Does not understand projects

Does not understand its role

Does not have the skills or knowledge for the role

Is not visible, active and committed to the project’s success

Does not enjoy the respect or trust of the business

Is not adequately informed

**Warning Signs of Project Failure**

Undefined or poorly defined project requirements.

Lack of project planning.

Lack of or poorly developed budget forecast.

Lack of stakeholder involvement

Lack of executive support

Frequent or large changes to project scope

Lack of change management process.

Failure to establish appropriate client/user expectations

Unrealistic deadlines

Insufficient resources

**Ethical Issues**

There are a number of different components of managing a project. While conducting project management, profit and staff motivation are often paramount. However, a project manager must also remember his obligation to be an ethical, responsible employer, employee and corporate citizen. Some of the ethical situations that one may face in the duration of project management could be the admission of wrongdoing, focus of blame, and hard choices regarding contracts.

**Key attributes of a successful software team:**

**1. Analytical mind**

There is no strict definition of an analytical mind. One thing is for sure: if you have an analytical mind you are able to think, observe data, remember and basing on these activities, resolve problems.

Unfortunately, this is the most crucial and needed skill of a good software developer. Why “unfortunately”? Well this means that not everybody can become a good software developer, even if they really struggle. Having analytical mind is to a big extend an inborn ability. If you do not possess this skill it is better to change the field of interest.

Solving many algorithmic and logic problems can increase the basic level of analytical thinking ability. However, every person has their own border (“maximum level” – like in RPG games) which can’t be exceeded.

**2. Big-picture perception of a software**

Creating software is like playing chess – to win you need to predict a few moves ahead. You should not only look at the presence but also into the future while programming. To be a good software developer you can’t only be focused on a small piece of software that you are implementing. You don’t only need to know how to implement something, but also why. You need to see the software as a whole not just as the small part you are currently working on. You need to understand the impact of your small piece on the whole. All in all, you simply need to see the bigger picture.

**3. Business oriented approach to software development**

In my definition of a good software developer, it is not enough to be technically oriented. You can be a good coder and your code can be of a really high quality, but still you won’t manage to understand and satisfy your customers’ needs. If you are not business oriented, it can potentially cause a lot of problems: misunderstandings, usability lacks or late changes in the functionality.

So, what does it mean to be a business oriented software developer? There are many factors that influence this feature of a developer, but in my opinion the most relevant are the below ones:

Understanding software from a business perspective

Appreciation of client’s needs

The ability of converting business problems into technical solutions

The ability to cooperate and understand people from non-technical stuff

**4. Teamwork eagerness**

A good software developer is not a person who sits for the whole day in front of the computer and codes. If a project is supposed to be successful, the communication inside the team is crucial. Exchanging thoughts, ideas, knowledge and experience can boost the efficiency and the quality of the solution. A developer who is not eager or able to communicate with other team members will not be able to fit in the team and in the process of developing software. I am aware that there are many very introverted developers who are doing really a great job and produce high-quality code. Yet in Agile, which is more and more common amongst IT companies, pure coding is not enough. Communication inside a team is one of the major keys to success.

**commitment iteration**

1. Adjust priorities

2. Identify an iteration goal

3. Select a user story

4. Split the story into tasks

5. Estimate the tasks

6. Sum the estimates

7. Ask for team commitment

Repeat steps 3 to 7 until your iteration is complete

**Key features of a good user story**

1 Users Come First

2 Use Personas to Discover the Right Stories

3 Create Stories Collaboratively

4 Keep your Stories Simple and Concise

Write your stories so that they are easy to understand. Keep them simple and concise. Avoid confusing and ambiguous terms, and use active voice. Focus on what’s important, and leave out the rest. The template below puts the user or customer modelled as a persona into the story and makes its benefit explicit. It is based on by Rachel Davies’ popular template, but I have replaced user role with persona name to connect the story with the relevant persona.

As <persona>, I want <what?> so that <why?>.

Use the template when it is helpful, but don’t feel obliged to always apply it. Experiment with different ways to write your stories to understand what works best for you and your team.

5 Start with Epics

6 Refine the Stories until They are Ready

7 Add Acceptance Criteria

As you break epics into smaller stories, remember to add acceptance criteria. Acceptance criteria complement the narrative: They allow you to describe the conditions that have to be fulfilled so that the story is done. The criteria enrich the story, they make it testable, and they ensures that the story can be demoed or released to the users and other stakeholders. As a rule of thumb, I like to use three to five acceptance criteria for detailed stories.

8 Use Paper Cards

User stories emerged in Extreme Programming (XP), and the early XP literature talks about story cards rather than user stories. There is a simple reason: User stories were captured on paper cards. This approach provides three benefits: First, paper cards are cheap and easy to use. Second, they facilitate collaboration: Every one can take a card and jot down an idea. Third, cards can be easily grouped on the table or wall to check for consistency and completeness and to visualise dependencies.  Even if your stories are stored electronically, it is worthwhile to use paper cards when you write new stories.

9 Keep your Stories Visible and Accessible

10 Don’t Solely Rely on User Stories