Agile Software Process

By
Sunil Kumar(Master of Sc.)
Bangalore, India

Agenda

- What is Agile?
- History of Agile
- Agile Manifesto
- Agile Methodologies Overview
- Roles within in Agile Team
- Summary

What is Agile

- Software evolve over time.
  - Agile software development is a group of software development methods in which requirements and solutions evolve through collaboration between self-organizing and cross-functional teams.
- Adaptive planning, evolutionary development, early delivery
  - It promotes adaptive planning, evolutionary development, and early delivery for your highly iterative and incremental approaches to software development.

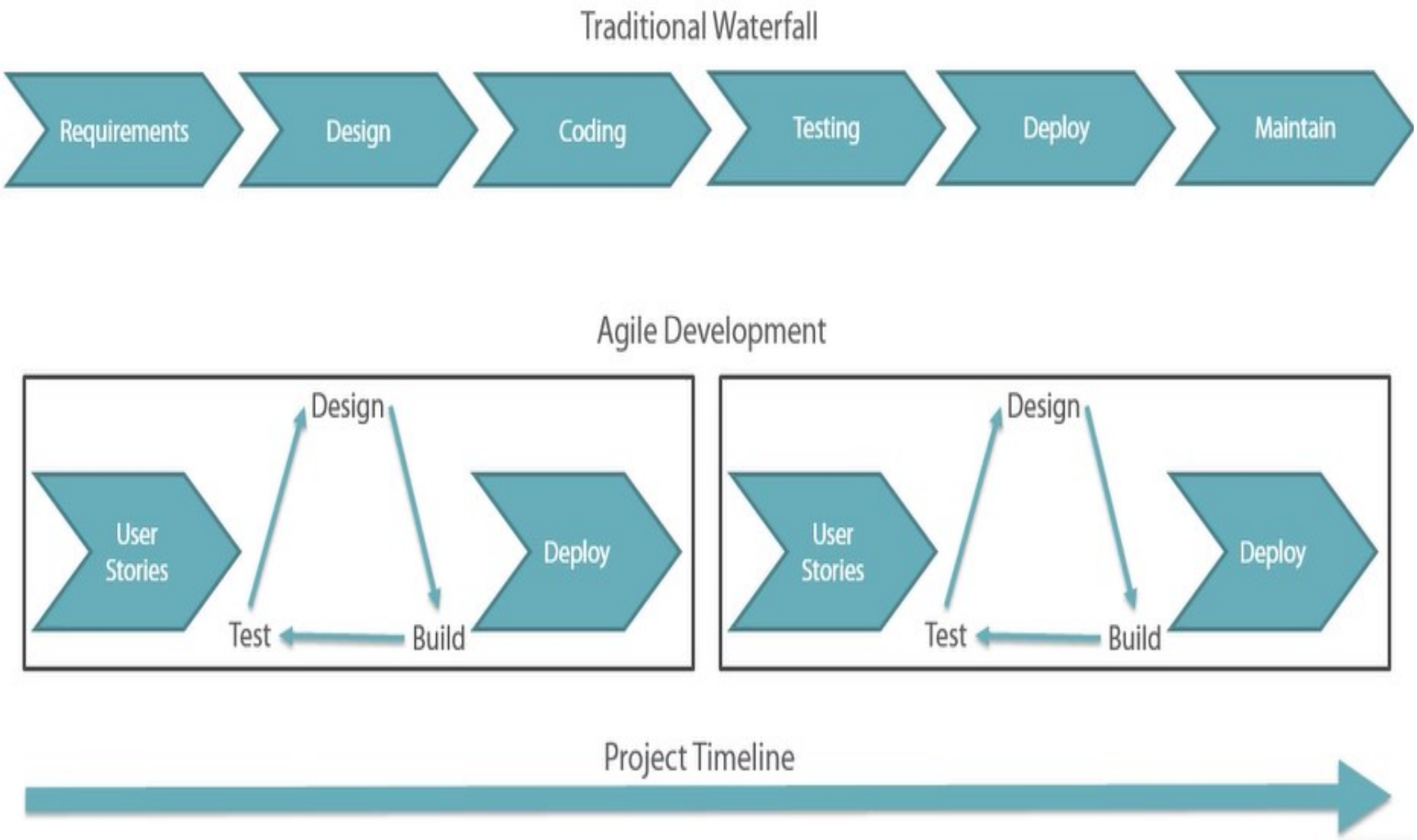
What is Agile

- The term agile was first used in 2001 when the Agile Manifesto was published as a unifying charter by many of the leading visionaries in the software field at the time, who were fed up with traditional approaches to software development.
- Unlike traditional software development practices, Agile development methodologies such as Scrum, Extreme Programming, Lean, Feature-Driven Development, Crystal, Adaptive Software Development, and others, incorporate close cross-functional collaboration, and frequent planning and feedback as fundamentals tenets inherent in the evolution of a software system.

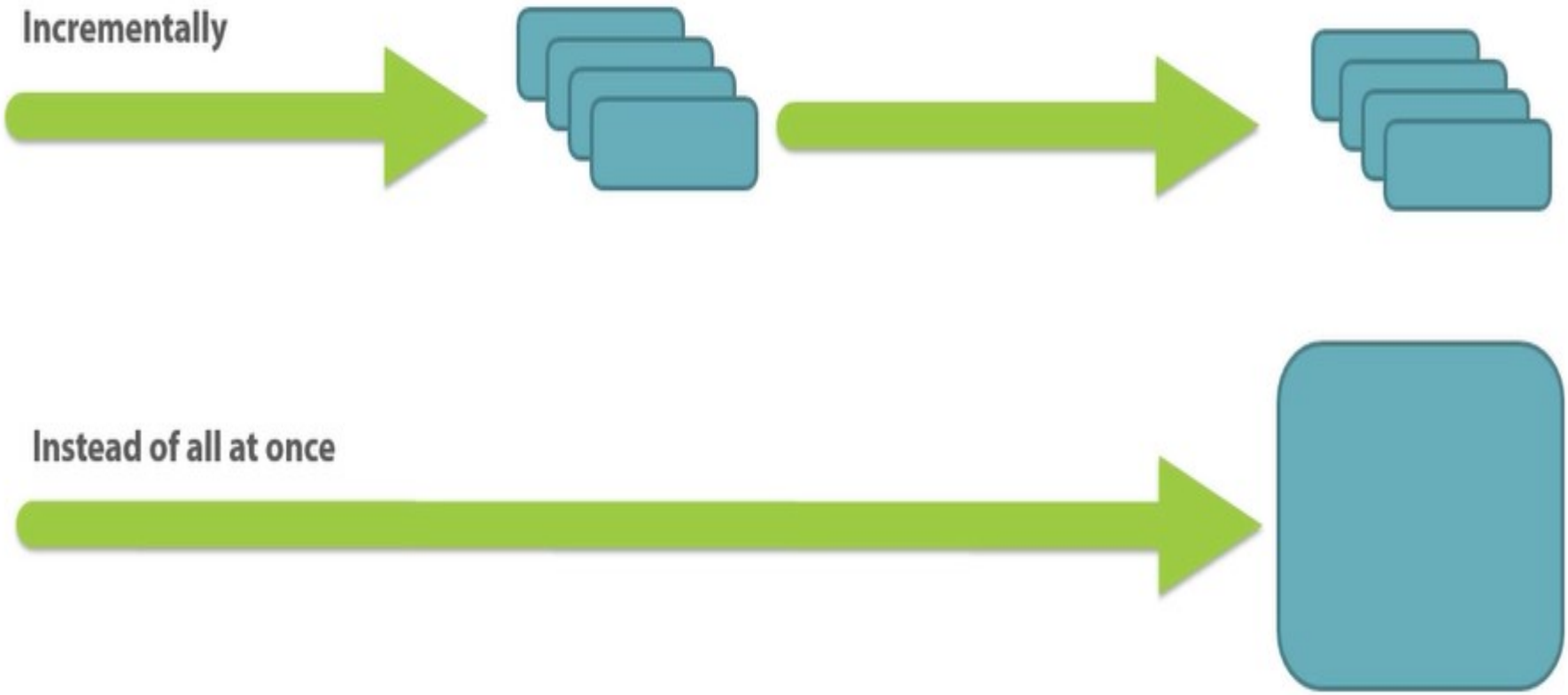
What is Agile

- Deliver value to business sooner
  - One of the key ideas behind Agile is that instead of delivering a "big bang" at the end of the project, you deliver multiple releases of working code to your business stakeholders. This allows you to prioritize on features that will deliver the most value to the business sooner, so that your organization can start to realize an early return on their investment.
  - The number of deliveries that we do depends on how long and complex a project is, but ideally you would deliver working software at the end of each sprint or iteration.
- Flexible response to change

Agile vs Traditional Model



What is Agile



What is Agile

- As per this diagram, it shows that with Agile we deliver incrementally instead of all at once. You should hold this thought in your mind as we progress through the rest of the material about Agile development.

History of Agile

- Many attempts to improve software development
- Small group of thought leaders met to discuss software process.
- This term was used as an umbrella reference to a family emerging lightweight software development methods like
  - Scrum
  - Extreme Programming (XP)
  - DSDM (Dynamic System Development Process)
  - FDD (Feature Driven Development)
  - Crystal
  - Adaptive Software Development

Agile Manifesto

- 17 people met to find the common ground about software development.
- The formed “The Agile Alliance” and signed the agile manifesto.

  http://agilemanifesto.org
- 4 core values
- 12 Principle behind the agile manifesto split into 3 groups.
  - Regular delivery of Software
  - Team Communication
  - Excellence in design

Agile Manifesto 4 Core Values

More Values Items	Valued-but less than left items
Individual and interactions	Over processes and tools
Working software	Over comprehensive documentation
Customer collaboration	Over contract negotiation
Respond to change	Over following a plan

Agile Manifesto Values

Values	Explanation
Individual and interactions over processes and tools	• Strong collaborating team > process tools • Coherent team > Individual gurus • Put more effort in building coherent teams
Working software over comprehensive documentation	• Documentation is important • Too much documentation is costly and documents become outdated • Generate just enough high-level structure/dynamics documents.
Customer collaboration over contract negotiation	• Continuous customer feedback is essential • Customer work closely with team • Feedback derive future iterations

Respond to change over following a plan	• Plans must be flexible to change
	• Change can and will happen
	• Multi-level planning.
	  • Weekly detailed plans
	  • Iteration plan
	  • Overall plan

Agile Methodology Overview

- Scrum
  - Scrum is a lightweight management framework with broad appeal for managing iterative and incremental projects of all types.
  - Ken Schwaber, Mike Beedle, Jeff Sutherland, and others, contributed significantly to the evolution of Scrum over the last decade and a half.
  - Over the last few years in particular, Scrum has garnered increasing popularity in the software community due to its simplicity, proven success, and improved productivity, and its ability to act as a wrapper for various engineering practices promoted by other Agile methodologies.

Agile Methodology Overview

- Extreme Programming (XP)
  - XP or Extreme Programming was originally devised by Kent Beck, and has emerged as one of the more popular and controversial Agile methods.
  - XP is a disciplined approach at delivering high quality software quickly and continuously.
  - It promotes high customer involvement, rapid feedback loops, continuous testing, continuous planning, and close teamwork to deliver working software at very frequent intervals, typically every 1-3 weeks.
  - The original XP recipe is based on four simple values: Simplicity, communication, feedback, and courage.
  - There are also 12 supporting practices. These are the planning game, small releases, customer acceptance tests, simple design, pair programming, test-driven development, refactoring, continuous integration, collective code ownership, coding standards, metaphors, and sustainable pace.

Agile Methodology Overview

- Crystal Methodology
  - The Crystal methodology is one of the most lightweight, adaptable approaches to software development.
  - Crystal is actually comprised of a family of methodologies like Crystal Clear, Crystal Yellow, and Crystal Orange, whose unique characteristics are driven by several factors, such as team size, system criticality, and project priorities.
  - This Crystal family addresses the realization that each project may require a slightly tailored set of policies, practices, and processes in order to meet the project's unique characteristics.

Agile Methodology Overview

- DSDM
  - Dynamic Systems Development Method, or DSDM as it is most commonly known.
  - DSDM dates back to 1994 and grew out of the need to provide an industry standard project delivery framework for what was referred to as rapid application development or RAD for short.
  - While RAD was extremely popular in the early 1990's, the RAD approach to software delivery evolved in a fairly unstructured manner.

Agile Methodology Overview

- As a result, the DSDM consortium was created and convened in 1994 with the goal of devising and promoting a common industry framework for rapid software delivery.
- Since 1994, the DSDM methodology has evolved and matured to provide a comprehensive foundation for planning, managing, executing, and scaling agile and iterative software development projects.
- DSDM is based on nine key principles that primarily revolve around business needs and values, active user involvement, empower teams, frequent delivery, integration testing, and stakeholder collaboration. DSDM specifically calls out fitness for business purpose as the primary criteria for delivering acceptance of a system, focusing on the useful 80% of the system that can be deployed in 20% of the time.

Agile Methodology Overview

- FDD
  - Feature-driven design was originally developed by Jeff De Luca. The first incarnations of FDD occurred as a result of collaboration between De Luca and a thought leader, Peter Coad.
  - FDD is a model-driven, short-iteration process. It begins with establishing an overall model shape. Then it continues with a serious of 2-week, design-by-feature, built-by-feature iterations. The features are small and useful in the eyes of the client.
  - FDD designs the rest of the development process around feature delivery using the following eight practices: Domain object modeling, developing by feature, components and class ownership, feature teams, inspections, configuration management, regular builds, visibility of progress, and results. Next we have Lean software development.

Agile Methodology Overview

- Lean Software Development
  - Lean software development is an iterative methodology originally developed by Mary and Tom Poppendieck.
  - Lean software development owes much of its principles and practices to the Lean enterprise movement and the practices of companies like Toyota.
  - Lean software development focuses the team on delivering value to the customer and on the efficiency of the value stream and the mechanisms that deliver that value.
  - The main principles of Lean include eliminating waste, amplifying learning, deciding as late as possible, delivering as fast as possible, empowering the team, building in integrity, and seeing the whole. Lean eliminates waste by selecting only the truly valuable features for a system, prioritizing and delivering them in small batches. It emphasizes the speed and efficiency of development workflow and relies on rapid and reliable feedback between programmers and customers.

Agile Methodology Overview

- Kanban
  - Kanban is applied to software development as a based planning and execution method. Rather than planning work items up front and pushing them into the work queue of a team, the team signals when they are ready for more work and pulls it into their queue.
  - Kanban historically uses cards to signal the need for an item. For software development teams, these cards are kept on a Kanban board, which is organized into columns and rows.
  - The columns represent the different states of a work item, from initial planning through customer acceptance. The specific columns a team uses should meet the needs of the team and be tailored to their context.
  - The rows on a Kanban board represent work items. Work items are sometimes grouped within areas such as feature sets and category types.

Agile Methodology Overview

- Kanban focuses on maximizing the throughput of a team. One of the ways it achieves this goal is through the application of work-in-process limits, or WIP limits, in each of the states of a work item.
- Under a Kanban, or Lean approach, of work in any state are seen as waste. The work-in-process limits enable the team to focus on the optimal flow of work items through the system, minimizing any associated waste. Kanban allows teams to achieve process optimizations while respecting and maintaining at a sustainable pace

Roles in Agile Team

- Agile teams are software development team first and members of a department second.
  - Everyone is a team member committed to a common vision and goal. In order to achieve these goals, team members do what is needed at the time it is needed.
  - This can be a bit unnerving for a number of folks used to strict roles and titles, but it is critical for the common good of the team.
  - Statements like I can't write tests, I'm a developer, have no place in an Agile team. Naturally everyone comes into a team with their own area of expertise and experiences.
  - Outside of the common project, the team still needs to fit into the larger organization, so you do still need to keep some of these titles, but if the entire team can make the shift in thinking, the results can be amazing.

Roles in Agile Team

- A team should have a product or domain expert.
  - There are team members with product and domain knowledge.
  - These people interact closely with users and/or customers, and in a commercial software scenario attempt to understand and research what benefits the market.
  - They serve the team by creating and prioritizing work, along with helping understand when work is complete. Commonly called a product owner in Scrum, this group can also consist of product managers, business analysts, and customers, and anyone with deep domain knowledge.
  - A critical success factor for this person or group is to act and speak as a single deciding voice of software product decisions.

Roles in Agile Team

- A team has members with cross functional skills
  - There are also team members who possess all the cross-functional skills necessary to build, validate, and deploy the product, and to determine how, if necessary, it interoperates with the larger system.
  - This group of core team members can consist of developers, testers, user interface designers, architects, business analysts, team leads, and technical writers.
  - This group works as a single team to design, code, test, and implement the software system.
  - The skills required to do so are extensive, and as a result, it is critical that everyone on the team can work together and act as a single unit. This does not mean that the team does not consist of Agile developers, Agile testers, or Agile usability experts, etc., just that they all work and collaborate together on the delivery of a software system. As a result, team members often perform multiple roles in addition to their primary responsibility.

Roles in Agile Team

- A team should have some form of leadership role.
  - In Scrum, this role is typically referred to as the Scrum master.
  - In some project, this may take on the project manager, program manager, team lead, or other title. On Agile teams, the primary goal of this person or group is simply to enable and ensure the success of the team.
  - This servant leadership role is in sharp contrast to the traditional task master and resource management focus of a Waterfall project.
  - Additional responsibilities may revolve around managing stakeholder expectations, company communication, progress reporting, facilitation, motivating team members, budget management, and staff management.
  - Just because a team is operating in an agile fashion, does not mean that all organizational requirements disappear, but every effort is taken to ensure that they have minimal impact on the ultimate success of the team.

Roles in Agile Team

- A team member can benefit from an agile coach or mentor.
  - Often referred to as an Agile coach, this person ideally does not have specific project deliverables, and instead maintains a focus on continuous improvement and engineering discipline. One goal of an Agile team should be to become better every single day, and the coach's job is to help guide the team to this direction.
  - The larger the organization, the more complex team structures can get. Cross-project teams, shared services, operations, configuration management, and database administration can all come into play, but the goal remains the same, define a software project and cross-functional team capable of delivering on that project and empower the team to do so.

Summary

- What is agile and how it differs to the more traditional Waterfall software development process. The main visible difference with Agile is that we deliver a business value sooner to the business by following an iterative development process, as opposed to the more "big bang" approach of Waterfall. We deliver incrementally instead of all at once.
- History of Agile with the formation of The Agile Alliance, who defined the Agile Manifesto.
- The Agile Manifesto represents a series of 4 core values and 12 principles that should guide us on how we develop software.
- Series of different Agile software development methodologies that are in use today in different software development teams.
- The most common are Extreme Programming and Scrum, but Lean development and Kanban are also increasing in popularity.
- Different roles we would find in an Agile team, from the development team itself to the product owners and domain experts, team leadership, and Agile coaches.

Common Agile Misconceptions

- Agile is ad-hoc, with no process control
  - When a team is new to agile, it can be hard for them to adjust to a new way of working, especially if they're used to working under a Waterfall-based methodology.
  - Agile is ad-hoc, with no process control. To be agile, you need to adhere to the Agile Manifesto, but following the manifesto doesn't mean you are using a defined process.

Common Agile Misconceptions

- Agile is ad-hoc, with no process control.
  - The manifesto describes a set of ideals. There are various different processes and project management templates that we can apply to our projects to help them become agile. Extreme Programming and Scrum are the two most popular, but Lean and Kanban are also becoming very popular.
  - When we try to implement the manifesto items, we generally need to apply lots of common sense and pragmatism to help us get to our goal, but if we want to warp a more formal process around the "how" of agile, as opposed the "why", then we would need to apply something like Scrum or Extreme Programming, which gives us more formal processes like: stories, iterations, standups, demos, retrospectives, test-driven development, and pair programming.

Common Agile Misconceptions

- Agile is faster and/or cheaper
  - Running an Agile team, doesn't mean we will finish a project quicker or for less money.
  - It isn't a direct money saver in that respect. What being agile about is, delivering value to the business sooner. You head towards working versions of the software quicker.
  - At the end of each development iteration, you're supposed to have working software to demo to the business. You may not have all their requirement in place, but what is there will work. This means re-thinking about how you plan your workload in each iteration.

Common Agile Misconceptions

- Agile is faster and/or cheaper
  - This means you deliver defined pieces of functionality in an iteration that may encompass work on the user interface and data access layer. It's a mind shift change that I've seen teams struggle with if they are used to working horizontally, but when they finally get it, the efficiency of a team is increased remarkably. Being agile is also about being able to respond to change. Requirements can change, and business can change partway through a delivery. I've worked with teams who treat this is a real negative thing. If you want to be agile, you need to expect and embrace that things will change. The tools and processes of Scrum, for example, are designed to help you react to these changes in a more efficient manner. "Agile teams do not plan their work or write documentation.

Common Agile Misconceptions

- Agile teams do not plan their work or write documentation.
  - Being agile is not an excuse to avoid appropriate planning or writing documentation. Agile is an on-demand or just-in-time approach, and encourages continuous planning and documentation, but only when it is needed for specific customer requirements.
  - Depending on what type of company you work for, formal documentation may not be something that you can avoid. For example, if you work in a very heavily regulated environment, then there's lots of upfront documentation that may be needed for evidence and submission to a regulatory body. If this is the case, then the delivery team will need to take this documentation into account.
  - Large diagrams instead of large documents of text is helpful in visualizing requirement quicker.

Common Agile Misconceptions

- An agile project never ends.
  - This might be true in some situations. You should continue to work on a project while the customer continues to get business value, and that value is worth more than the cost of developing it. Most products in any industry have a point of diminishing returns. This is the ideal time for an Agile project to end.

Common Agile Misconceptions

- Agile only works for small organizations.
  - Agile works for projects, teams, and organizations of any size, not just small projects. This doesn't mean it will necessarily work for all organizations, but size is rarely a factor. Large and complex projects and organizations are often excellent candidates for an Agile transformation, where it is difficult or impossible to know all of your customer's requirements in advance.

Common Agile Misconceptions

- Without upfront planning, Agile is wasteful.
  - This assumes that your customer knows the details of all their requirements in advance. If this is true, then by all means, undertake comprehensive upfront planning. However, in reality this is rare, and usually leads to greater waste of having undertaken design and development work that was ultimately unnecessary.

Common Agile Misconceptions

- Agile is not the solution to all your problems.
  - Agile is a change in approach and culture that comes with its own sets of benefits and issues. If you're working in a well-established team that has not been following any agile processes, then changing them over will not be an instant transformation. You need to do it slowly, and make sure everyone has a say in the decision-making process. Otherwise, you may get resistance from team members who fear change, which is a perfectly normal human characteristic. Convincing your team isn't the biggest hurdle though. Your biggest challenge is making sure that your leadership team understands and wants to adopt Agile as a way of working. Once you have achieved this and have leadership buy-in, then the rest of the adoption just takes time and patience as everyone adjusts.

Common Agile Misconceptions

- Agile means no commitments.
  - Some people think that agile teams don't make commitments and there are just 6-8 developers working until someone declares "we're done“. In all agile flavour deferments, or not completing an item that was promised, is treated as an exceptional occurrence and not treated lightly. This commitment to making a cadence of delivering software in small increments builds a high degree of trust between the team and other interested parties such as stakeholders, users, and customers.

Common Agile Misconceptions

- Agile development is not predictable.
  - Successful Agile teams bring predictability to software development because every step of the way they're communicating, deploying real code, and adapting to change, and keeping any documentation current.

Common Agile Misconceptions

- Agile is a silver bullet.
  - Avid supporters of Agile will sometimes claim that moving to Agile will fix all of your problems. This isn't the case. It's important to stress that the Agile Manifesto is a set of values and principles that define a core attribute for software development. These values point to collaboration, rapid feedback loops, and quality. In this way, Agile exposes your problems. Before any issues can be addressed, they need to be surfaced. Once exposed, teams can work to eliminate friction and any blocking issues.

Common Agile Misconceptions

- There's only one way to do agile.
  - The Agile Manifesto consists of four core values and 12 principles. It doesn't document any actual implementation details. There are many interpretations of Agile that form different methodologies, like Scrum, Extreme Programming, Kanban, Feature-Driven Development, to name a few.
  - Each style has its own benefits and weaknesses, and you must evaluate your own situations to decide which methodology is the best fit for your team.

Common Agile Misconceptions

- Agile doesn't need upfront design.
  - There's a misconception that Agile teams do all their design work on the fly. This is not true. Some elements of design will be done long before any code is written, while other elements of design will be completed as sprints i.e. prior to when they're needed.
  - Some design will be done as the code is being written. This is commonly referred to as emergent architecture and design. It emerges as needed, not before and not after.
  - Upfront design tries to answer every single question prior to the questions even being surfaced. Agile teams design the highest priority features first, then medium priority features, and then finally the lowest priority features. This process correctly identifies the questions as they become clear, therefore providing the correct answers when needed.

Common Agile Misconceptions

- Being Agile is pain-free.
  - Transforming from a traditional Waterfall team to a successful Agile team is not an insignificant effort. There are serious growing pains involved with such a transformation.
  - You won't just completely change everything about your company's development culture overnight. It will take study, practice, courage, and commitment to make the change.
  - The goal is to make sure that you fail fast when you try new things, so that you can quickly try a new approach. If you can work your way through the learning curve, you'll end up much better off in the end.

Common Agile Misconceptions

- We're doing scrum, so we don't need to pair program, refactor or do test-driven development.

Summary:Common Agile Misconceptions

- Agile is ad-hoc, with no process control
- Agile is faster and/or cheaper
- Agile teams do not plan their work or write documentation.
- An agile project never ends.
- Agile only works for small organizations.
- Without upfront planning, Agile is wasteful.
- Agile is not the solution to all your problems.
- Agile means no commitments.
- Agile development is not predictable.
- Agile is a silver bullet.
- There's only one way to do agile.
- Agile doesn't need upfront design.
- Being Agile is pain-free.
- We're doing scrum, so we don't need to pair program, refactor or do test-driven development.