"We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:
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 Agile Manifesto*

12 Principles Behind the Agile Manifesto

Aside from the base manifesto (above), the Agile philosophy also includes 12 principles that flesh out the core concepts. Each of the 12 principles are interconnected, they flow naturally into one another and cross-pollinate. They create a mesh of ideas that rely on the foundations of being open, creative, and *empathetic*. Agile is an accommodating guide, an ethos that understands the value of the individual in relation to the strength of the team. Agile extends a sense of adventure and iterative non-attachment. Agile is an invitation to a non-dualistic[1] approach to development, an opportunity to advance a culture of sharing and prosperity founded upon the ability to grow, learn, and be flexible in the face of inevitable challenges and a rapidly changing world.

The 12 principles are as follows:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.

[1] No more "us and them" mentality.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Commentary

A few notes of extrapolation have been provided below to demonstrate some of the ways these principles may be applied practically.

Involve the customer
It is better to include the customer early than to assume you know what they need. This not only provides valuable insight and context for the design and development team, but also involves your customer in a way that promotes their personal *investment* and *loyalty*; allowing them to feel valued, heard, and understood. To be understood is such a basic human need, and to include your customer in the design process is an incredibly powerful way to both humanise and promote the company.

Be flexible
Situations change, people change, and software needs to be able to evolve and change too. A good piece of software exists to solve a problem, and it is natural that over any given amount of time new problems arise, while others evolve. Without problems that need solving, there is no need for a solution to exist - therefore, the development process must be as fluid, as Agile, as life demands. It needs to be able to flow with, respond to, and hopefully (via correct iteration) *forecast* the kinds of challenges a customer or sector is likely to experience.

Deliver frequently
Releasing software frequently is advantageous, and encourages;

a) The development team to try new features.
b) The ability to gain feedback on features more quickly.
c) The opportunity to engage the customer throughout development.
d) The ability to iterate and grow organically with feedback.

As a product grows, the creative / experimental elements and the need of the customer naturally become homogenised via the process of release, insight,

and iteration. It is important to continue to deliver features frequently. This allows both for the ability to "fail fast" - to adapt in real time as a product is built and delivered[2], and to successfully remain Agile enough to iterate solutions based upon customer feedback. If the working software hasn't been shipped, it's not finished and so no progress has been made. Unreleased software is inventory. And inventory is a cost, not a piece of income or value.

Communicate often and in the open
Each extra step in communication leads to lost information, therefore it is not only useful to involve all relevant parties in conversation, but to enable teams to communicate frequently and fluidly. This allows staff to feel included in stakeholder level decisions, draw new context and meaning, and take ownership over their product while understanding the impact their work has.

Empower your team
With the correct coaching, environment, and tools, developers will feel motivated and enjoy a sense of investment that will add a dimension of meaning to their work. If projects are built around people that aren't motivated or have been demotivated due to a lack of trust or support, that project isn't likely to succeed. Even worse, developers may leave - taking important historical learning and context with them - which will destabilise the team foundations and the stability of the product.

Maximise the amount of work not done
Promote simplicity, minimise complexity. A complex developmental process will likely generate complex software. Remove procedures that are no longer relevant, automate manual work, use existing libraries instead of writing your own, et cetera. It not only saves time and money, but frees up valuable development resources to work on new features.

Promote technical excellence
Allowing a team to "fail fast" allows them to grow and incubate technical experiences via rapid proliferation of innovations and ideas. If teams neglect a good technical design for too long, their speed and time-to-market will suffer as they become bogged down in refactoring. As a result, their ability to grow the product as a reaction to a changing market will diminish and agility will be lost.

Learn and adapt
Utilise both a daily stand-up, and release meetings to discuss what is working, and what isn't. Brainstorm together as a team, and constantly reassess processes to come up with new ways of working together as the team moves forward. It is important to encourage seeing setbacks as opportunities to grow and advance.

[2] It is also possible to A/B test features in this way, collect the feedback, then deploy the most suitable feature.

Glueing it all together: a deeper look at empathy and iteration.

Agile, and its 12 principles are as simple or as complex as needed. Like the development processes they promote, the concepts are self-iterative, and can be fed back into themselves to define more complex use cases.

There are two fundamental concepts that lie at the heart of an Agile design process. These are *empathy* (which when properly distilled becomes *insight*), and *iteration*[3].

Empathy

Empathy can be summarised in a creative context as the *meaning* and *insight* gained via the sympathetic distillation of data between nodes; a flow of communication rooted in understanding and reciprocation.

Empathy, used correctly, creates a bridge[4] between two or more previously isolated nodes. For example; customer and company, problem and solution, designer and developer, team and product, et cetera[5].

This empathetic connection between nodes allows a sphere of influence to grow which includes all parties in an orientation that fosters *meaning*, *direction,* and a shared *focus*. Empathy is about listening, understanding, and venturing above all else to create the conditions from which *genuine insight* can be rendered. Genuine insight plants the foundations for *authentic* mutual relationships which in turn create strong and resilient structures of community, sharing, learning, and growth. Empathetic insight is also the foundation from which you are able to reasonably predict the direction of your product, see setbacks as growth opportunities, and distill valuable truths about your processes without bias or blindspots.

Agile disowns the “us and them” mentality that inhibits growth. Agile is an invitation to put aside the singular and create a community around a product that engenders all involved with a sense of ownership. From business stakeholders to designers, developers, and the customer - all are initiated into an empathetic process that allows ideas to be collected, prioritised, and enacted upon based upon the needs of the customer in a cycle of *frequent iteration*.

---

[3] Iteration is the ability to act upon insight, to take what is known and feed it back into the creative process indefinitely.

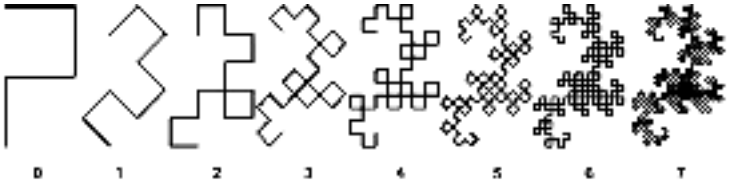
[4] This can also be imagined as an *entanglement* or threading together of two differently traveling paths, this creates a harmony or resonance - “striking a chord”.

[5] Naturally, empathetic connection between nodes aren’t limited to pairings, but can extend to an endless network of inter-connections.

Iteration

Through the process of iteration, complex systems and designs can emerge that transcend even the original concept, becoming what they are needed to become in a given environment, morphing and growing with the customer, consistently adding value over time. Agile, underscored by empathetic insight and iteration takes low-resolution ideas, and progresses them naturally towards high-resolution realities.

Empathy > Insight > Iteration > ∞ :



Agile iteration provides a preferable outcome over older "a > b" release patterns. In such a waterfall release model, a product is shipped when it is finished, likely prohibiting adaptability and denying software the ability to evolve and grow. Worse still, a product may no longer meet the requirements of the customer by the time it is shipped, leading to a possible breakdown in communication, and a corruption in the relationship the customer has with the product.[6]

Modern systems are designed and built continuously, dynamically, and are shipped frequently, directed by customer need. In this way, there is a distinct and profound correlation between Agile processes, component-based development frameworks, and empathetic development and learning. The Agile system is a *direct reflection* of the people that utilise it. The same process of customer interaction, sharing, and learning are reflected in the technology itself - via shared libraries, components, and open source culture. This is true too of the way data is shared at large, or how people communicate via social networks. You can extrapolate the iterative concept as far as you wish; what is true at the macroscopic level reflects itself into the microscopic - from systems and software, to individuals and people using that software. To understand this is to understand why the Agile framework, and the systems that spring from it, are so successful.

To simplify, people aren't business processes, they aren't unidimensional and don't exist in a neat line that traverses in an "a > b" manner like the waterfall approach of old. Software should reflect the user as to become virtually invisible. People are diverse, changeable, complex, empathetic beings - so it makes sense that the systems they interact with should reflect these qualities.

[6] This can also lead to losing a customer entirely.