CSE-322

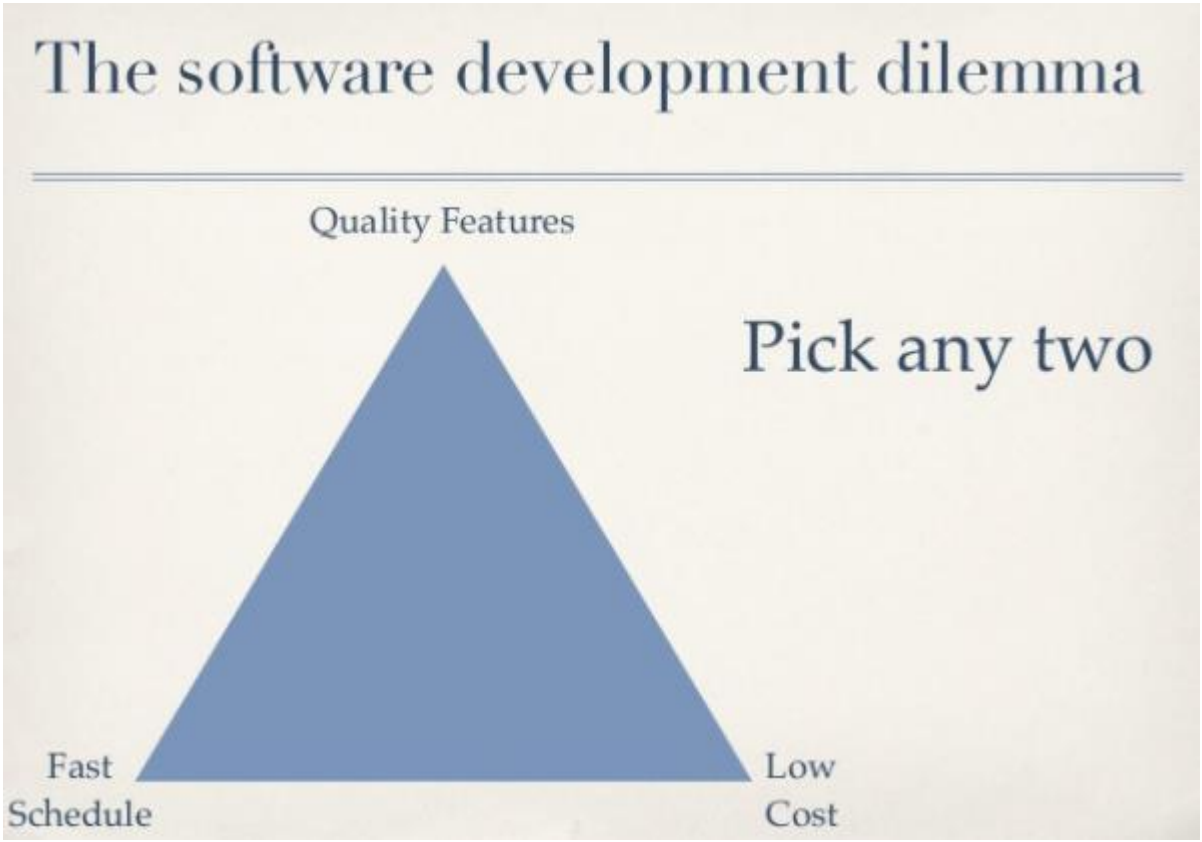
Software Engineering Laboratory

Common Problems in Software Engineering (and a few solutions)

Reference1, Reference2

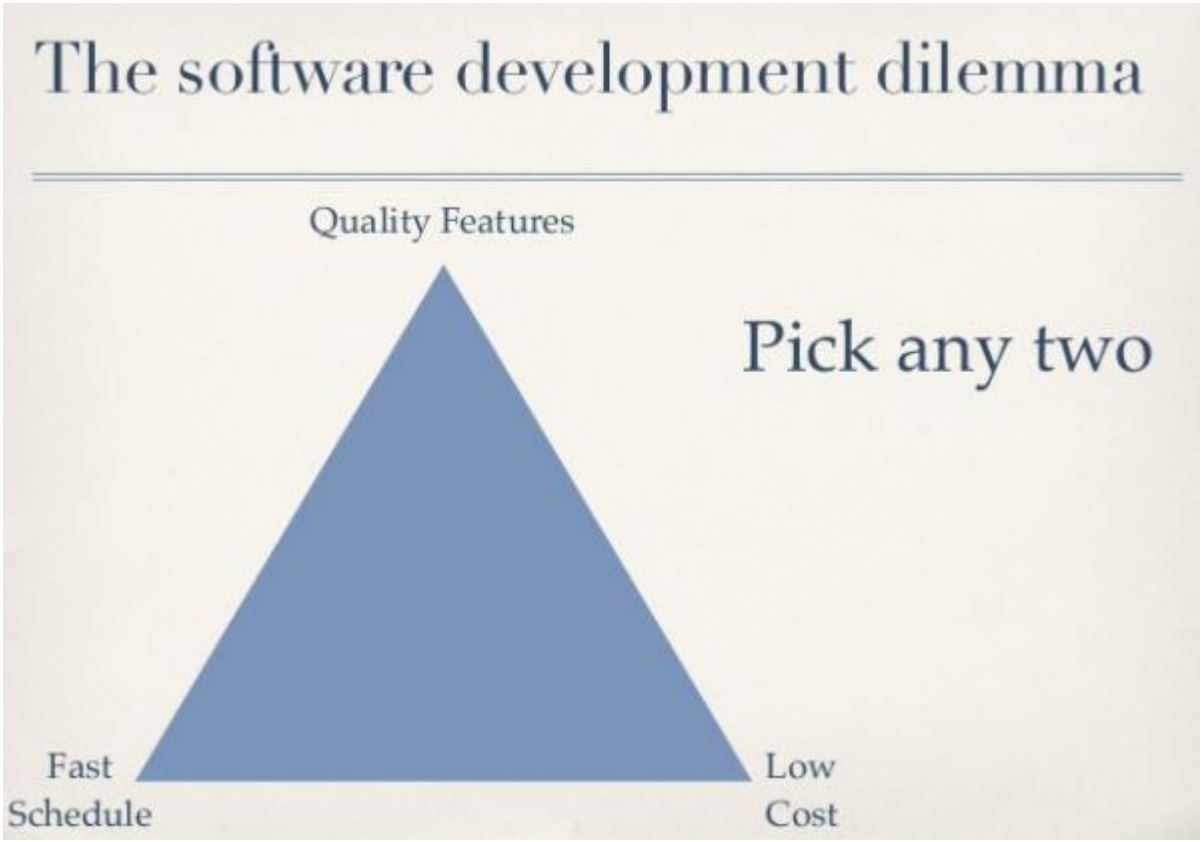
Md. Saidul Hoque Anik
anik@cse.uiu.ac.bd

1. The Iron Triangle



- Programming is not magic
- If client chooses all three, ___?___ suffers

1. The Iron Triangle



- Programming is not magic
- If client chooses all three, quality suffers

1. The Iron Triangle

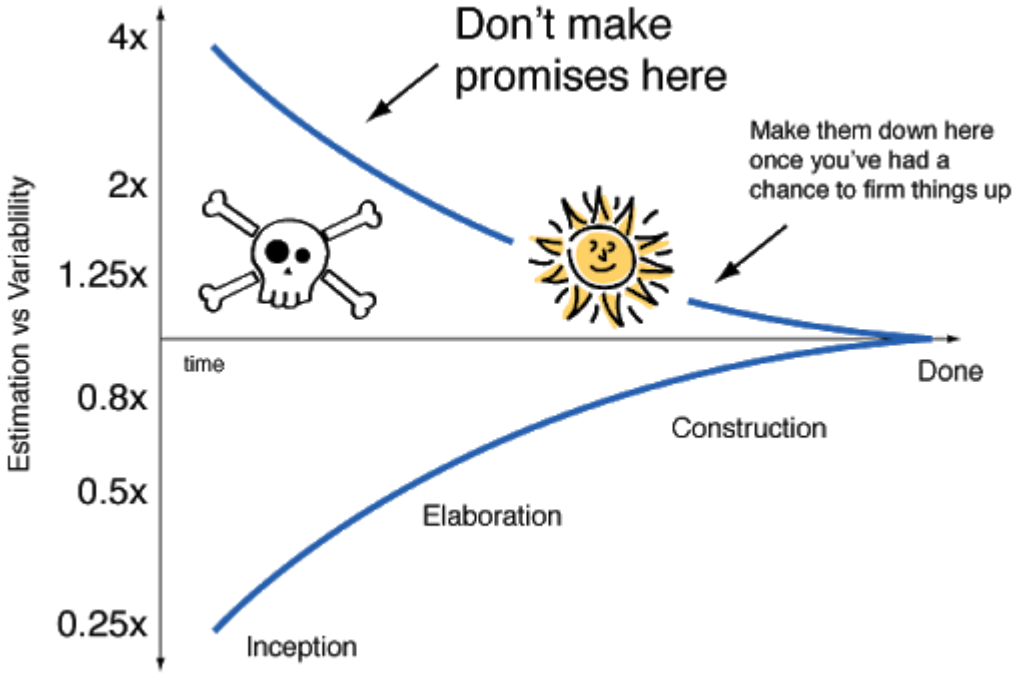
"Never delegate authority; delegate liability."

"To deliver the best possible combination of features by a given date, there must be control over the resources, and over the feature list. There's no way out of this. If you go to the store with a huge shopping list and twenty dollars, you need the authority to go to the money machine for more cash, or the authority to make changes to the list. And shopping is a lot easier than software development." - *Ron Jeffries*

2. Multitasking

- Multitasking is an illusion. It takes away 20% of the workers time to switch context.
- You might need to inform the client if their requirements are causing you to multitask

3. The cone of uncertainty



- After feeling the sting of underestimating, one common reaction is to double or triple the estimate the next time round. This definitely lowers the upfront risk, but padding the numbers is harder than it sounds.
- Give too big a number, and sponsors will balk and not approve your project. Give too low a number and you risk running out of money. This gets doubly dicey when you are bidding on fixed bid contracts where there is even more pressure to keep the numbers down.
- Most projects add some kind of padding onto the final numbers to give themselves sufficient wiggle room. Another things teams can do is compare this project with others.

3. The cone of uncertainty

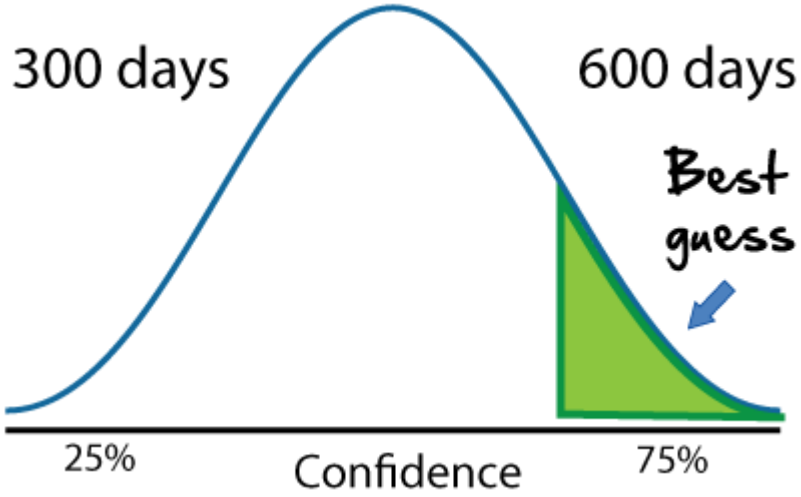
Size the project relatively

Humans are really good at sizing things relatively. We can't tell you precisely how big a rock is. But we can tell you how big it is compared to something else. We can use this when sizing projects too.

Agile's default mode of operations is transparency and visibility. So it should be no surprise that the Agile way of dealing with the cone is to be upfront and honest.

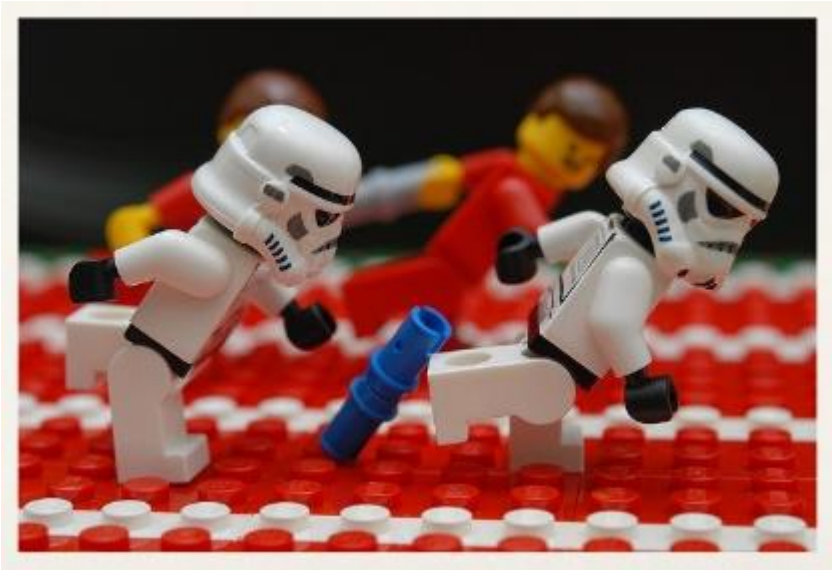
Say, ***"Look. We don't know how long this is going to take. This is our best guess. But if you give us a couple iterations, we can build something, measure how long that takes, and then tell give you a much better sense of how big this thing is."***

3. The cone of uncertainty



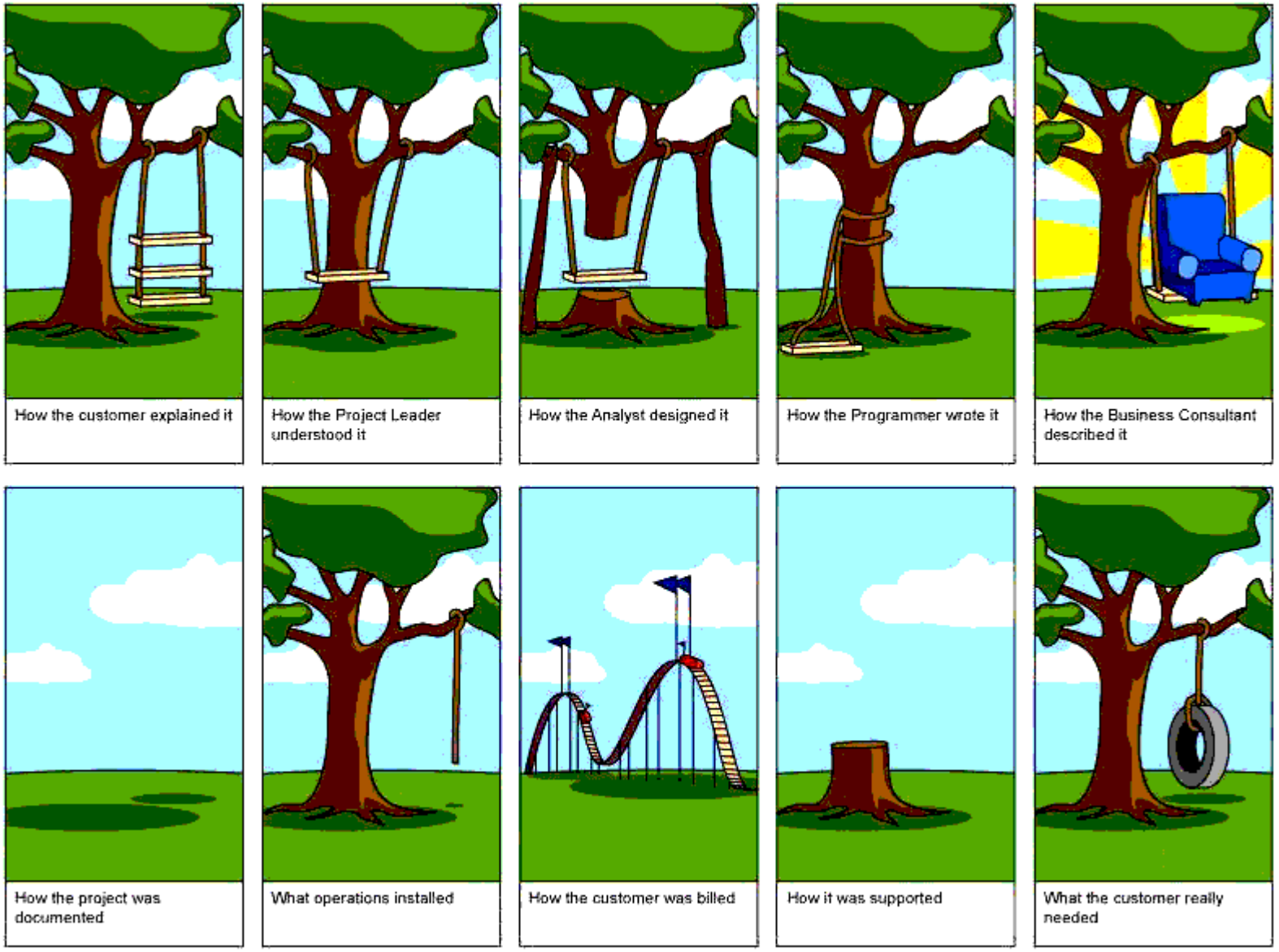
Another approach some Agile teams will take to help communicate the uncertainty is to present the estimates as a range. This has the advantage of visibly showing sponsors the uncertainty that comes with the project, and then letting sponsors decide how much risk they can afford to take on.

4. Information is lost in handoff

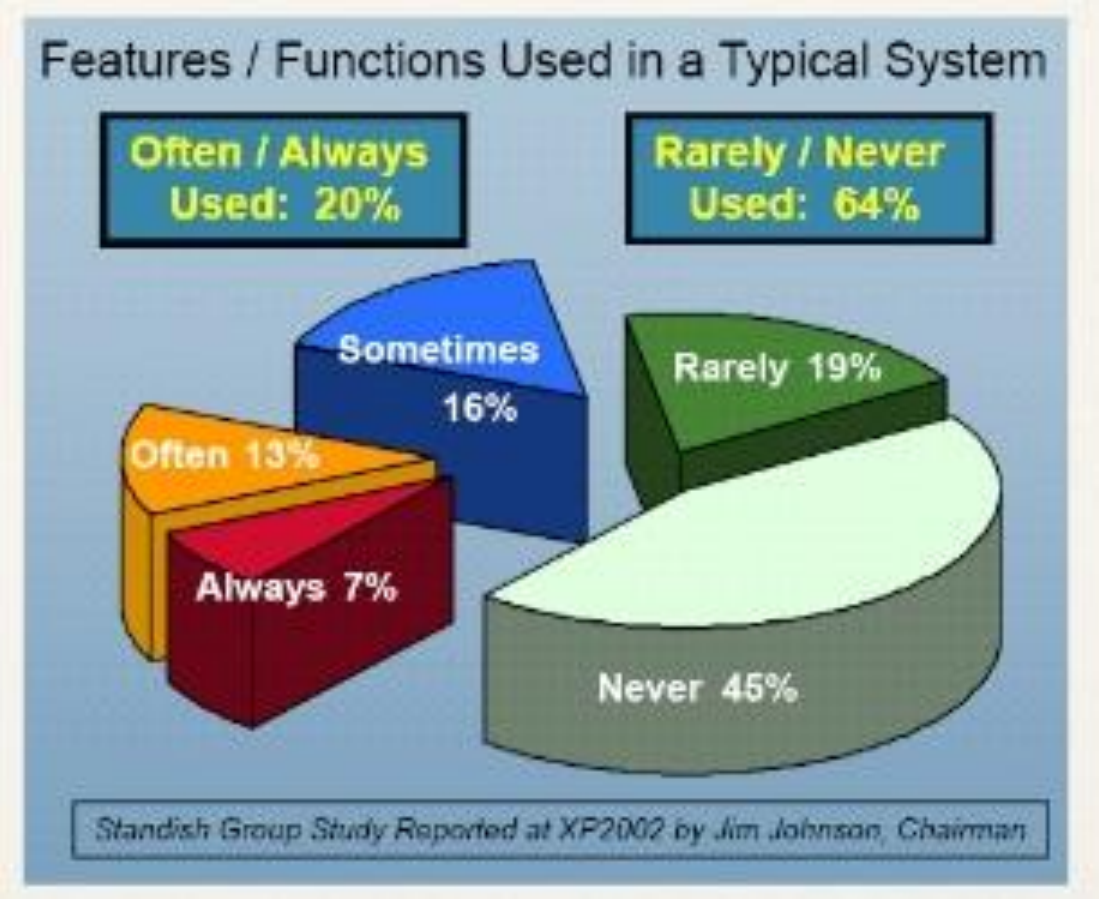


Miscommunication between clients and developers can be as common as miscommunication between two developer teams.

4. Information is lost in handoff



5. No Opportunity for feedback



- In combination with the cone of uncertainty, this is deadly
- Ask customer what they want in the beginning
- Penalize them for adding things later

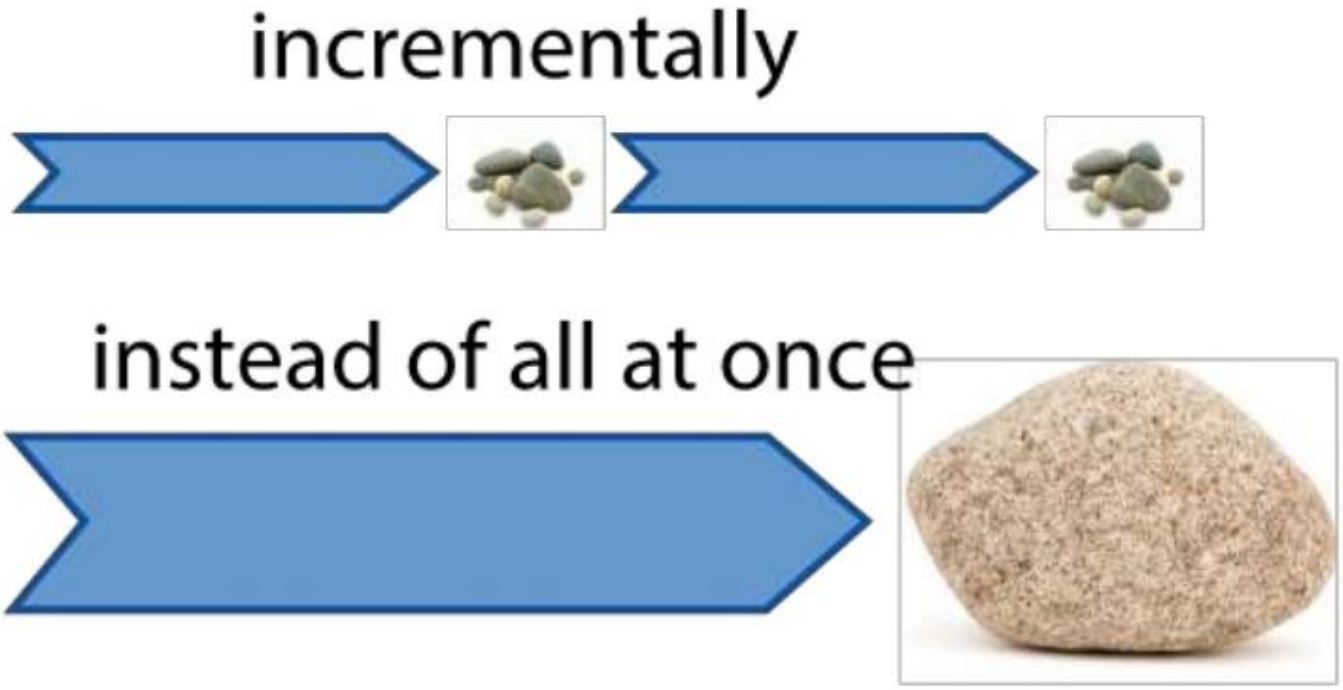
Result: Low Success Rate

"The main thing that pushed Agile and Scrum was that the success rate on the traditional projects terrible, only 45%. If that was a car manufacture company, that would mean you are throwing every other car you build."

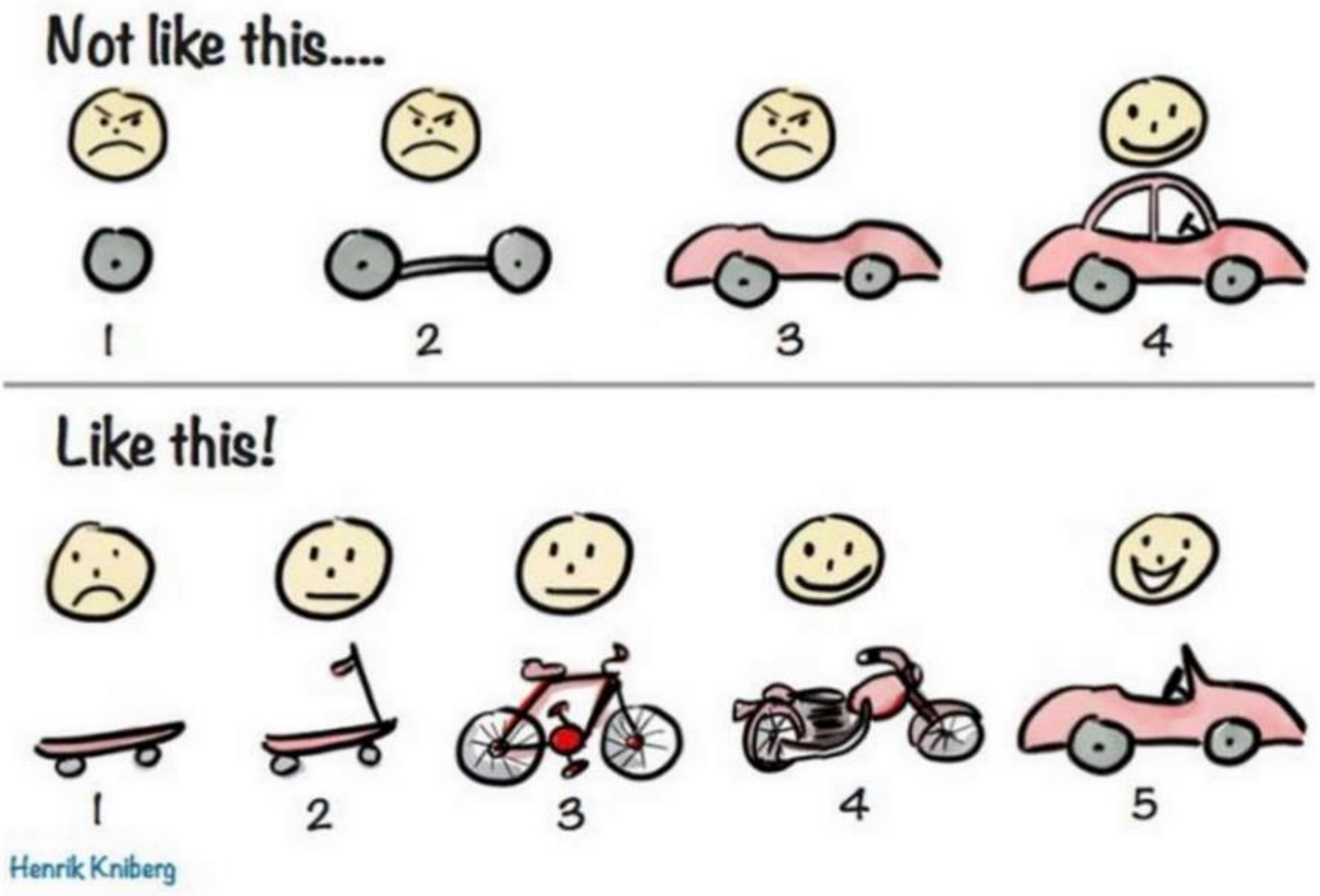
– Ken Schwaber (Co-creator of Scrum)

Introduction to Agile

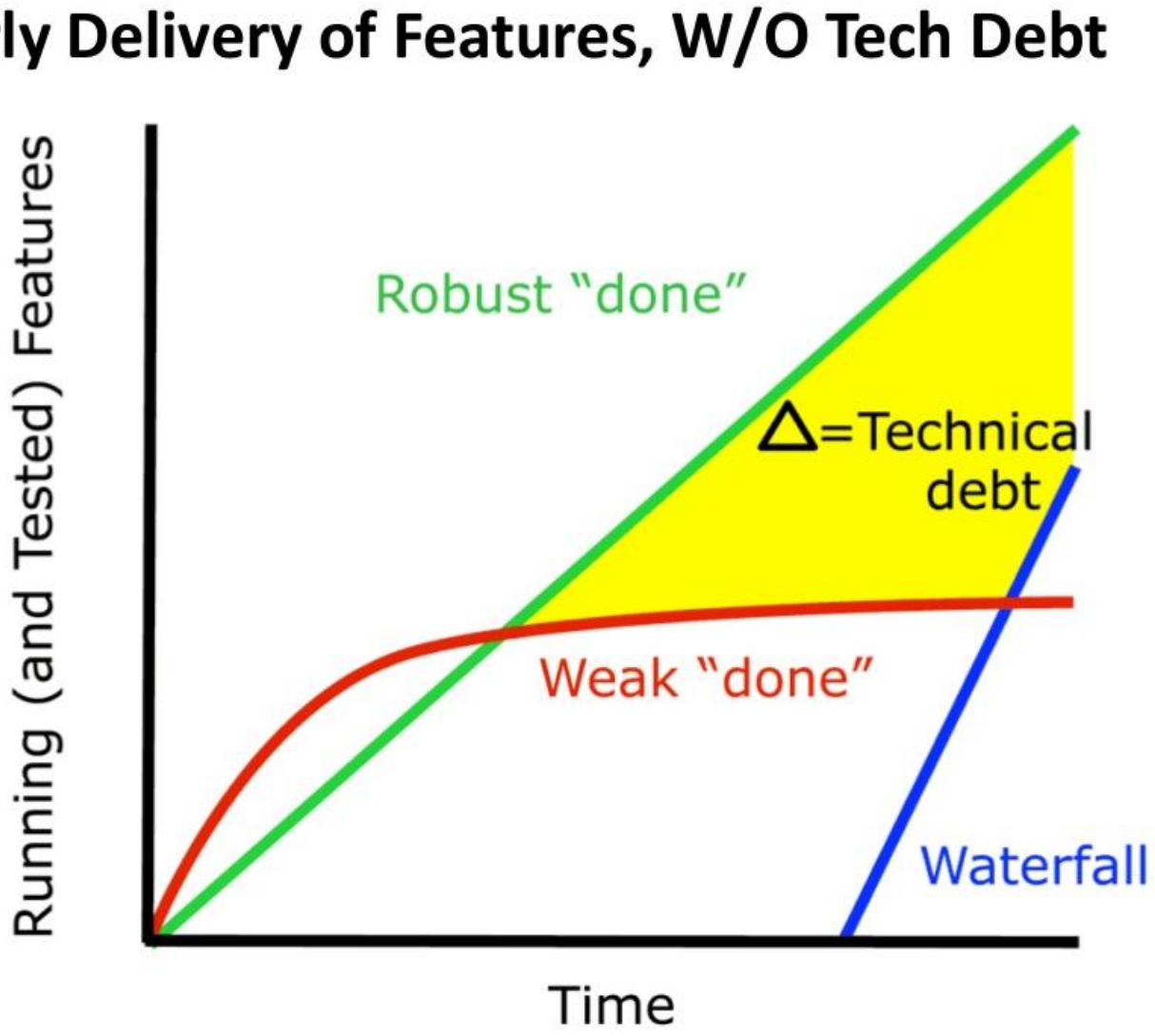
Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.



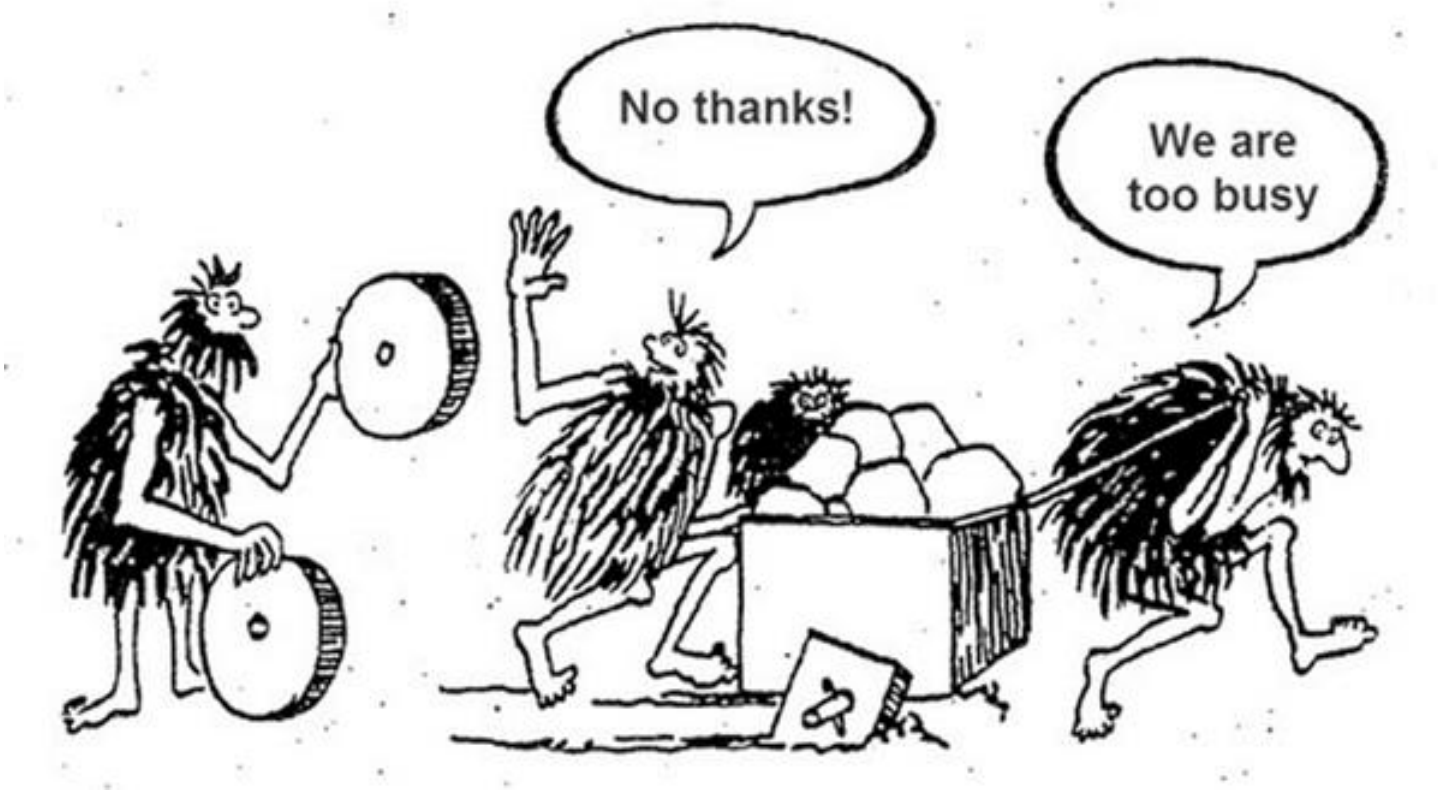
Building Product the Right Way



Building Product the Right Way



Technical Debt

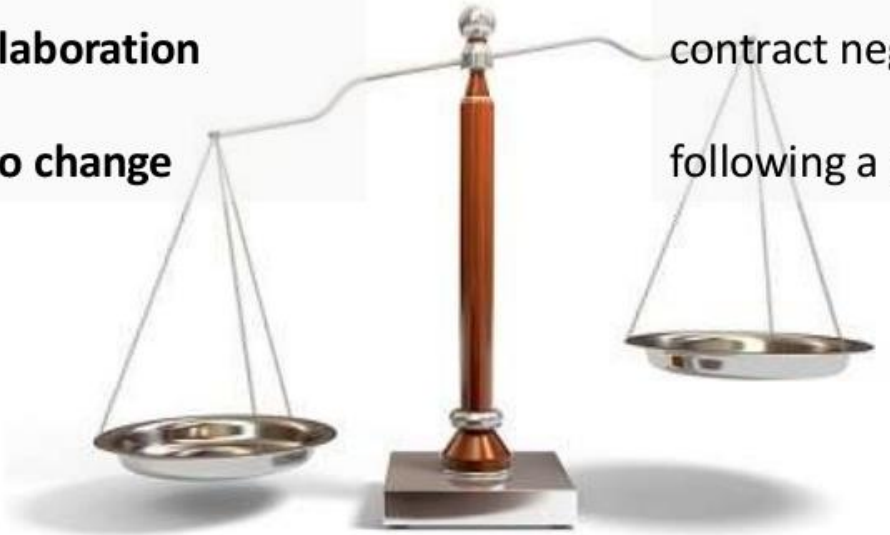


Technical debt is essentially, a series of bad business and technical decisions that have a snowball effect if not addressed swiftly.

The Agile Manifesto

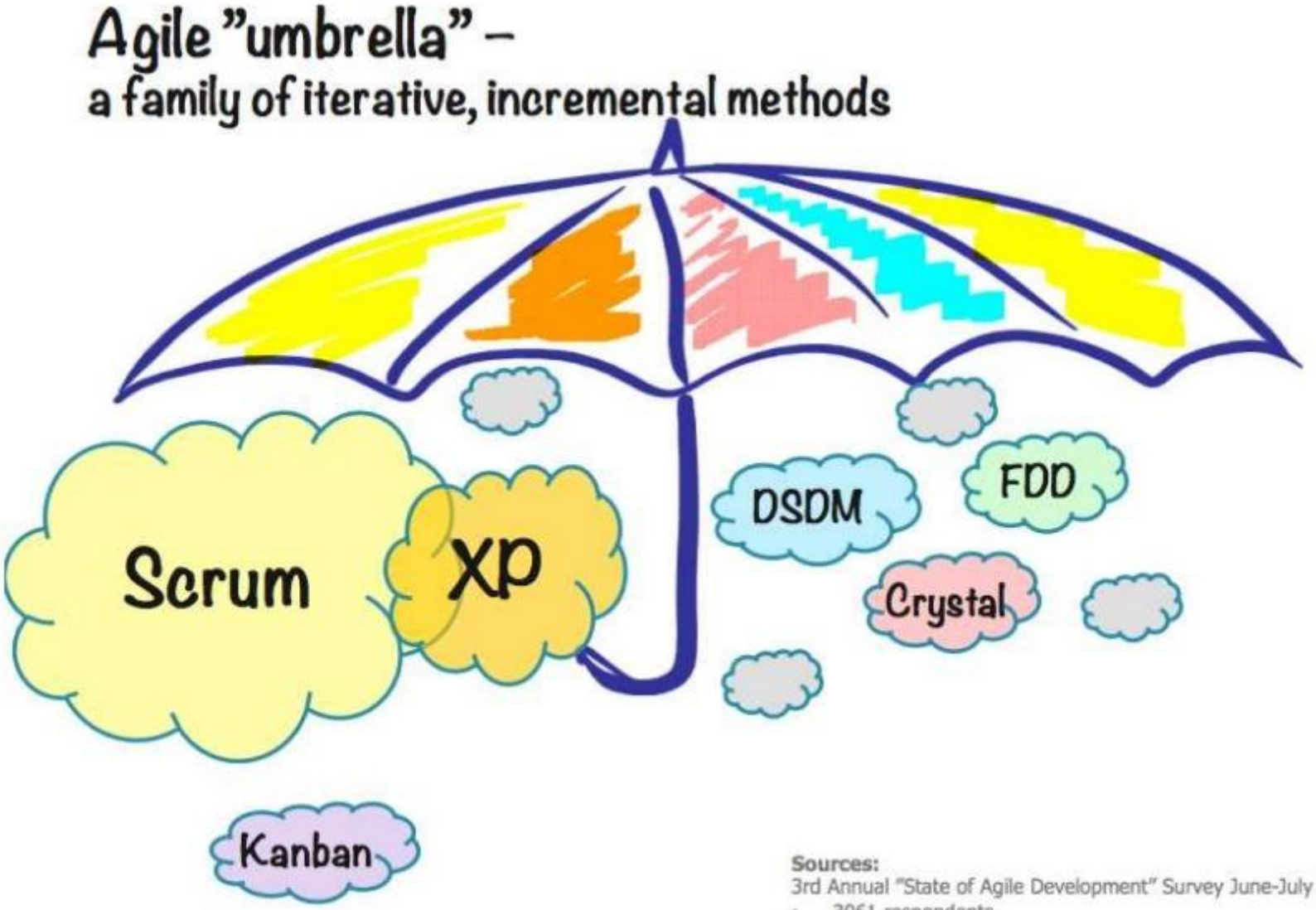
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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



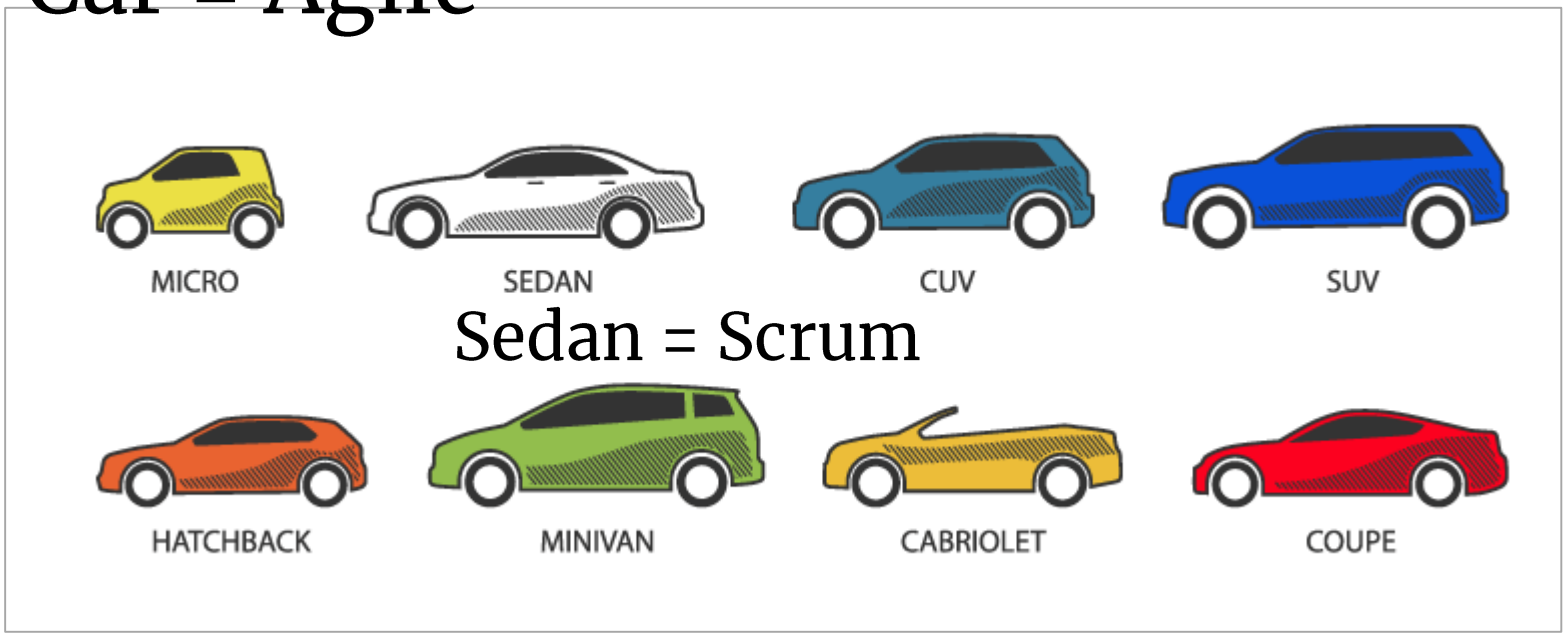
That is, while there is value in the items on the right, we value the items on the left **more**.

The Agile Manifesto

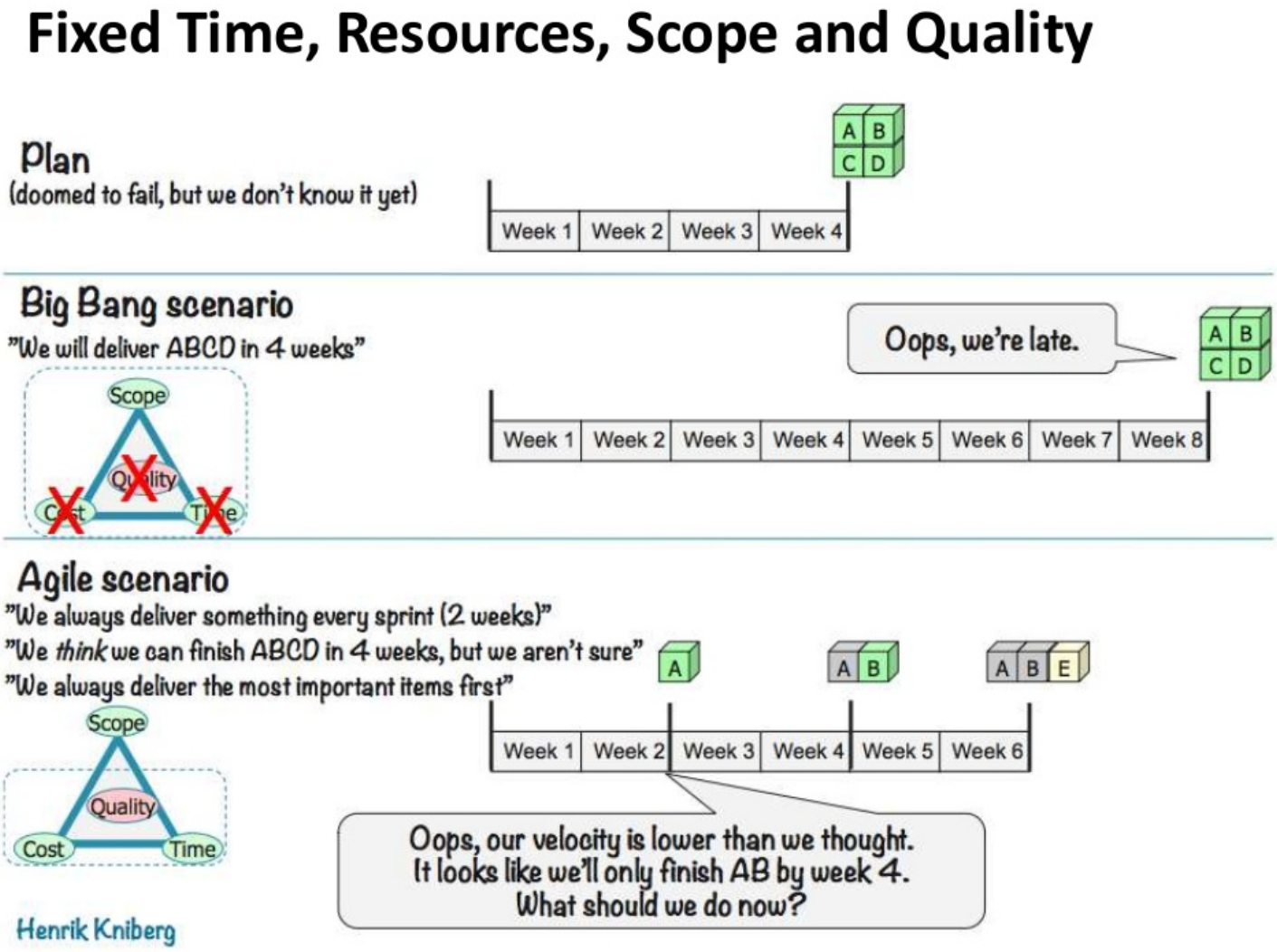


Agile vs SCRUM

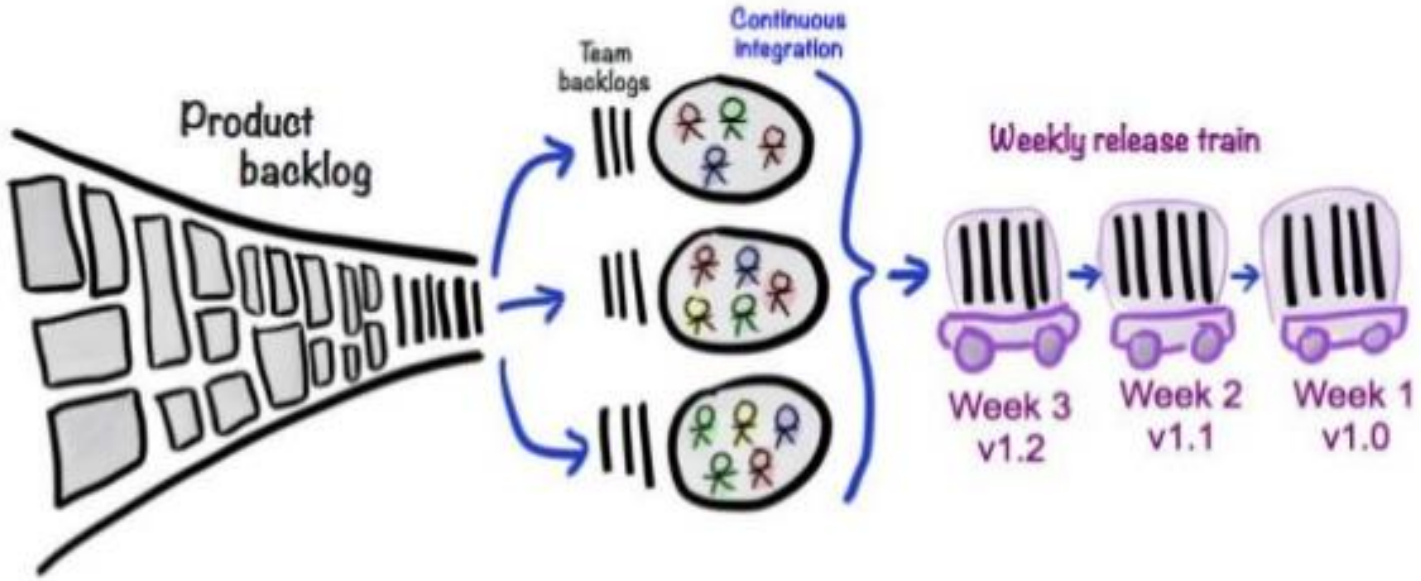
Car = Agile



Agile Process

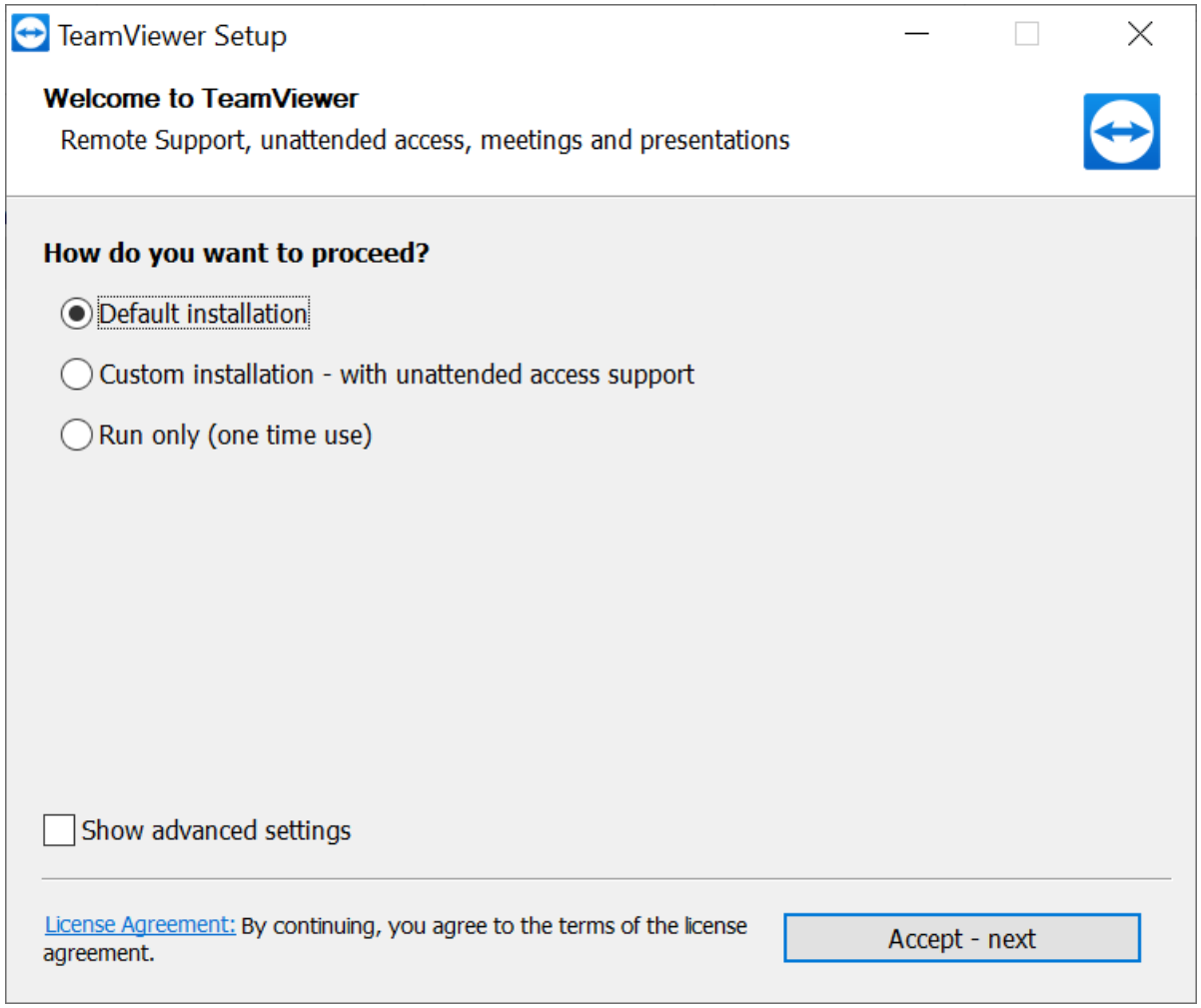


Agile Process with Multiple Team

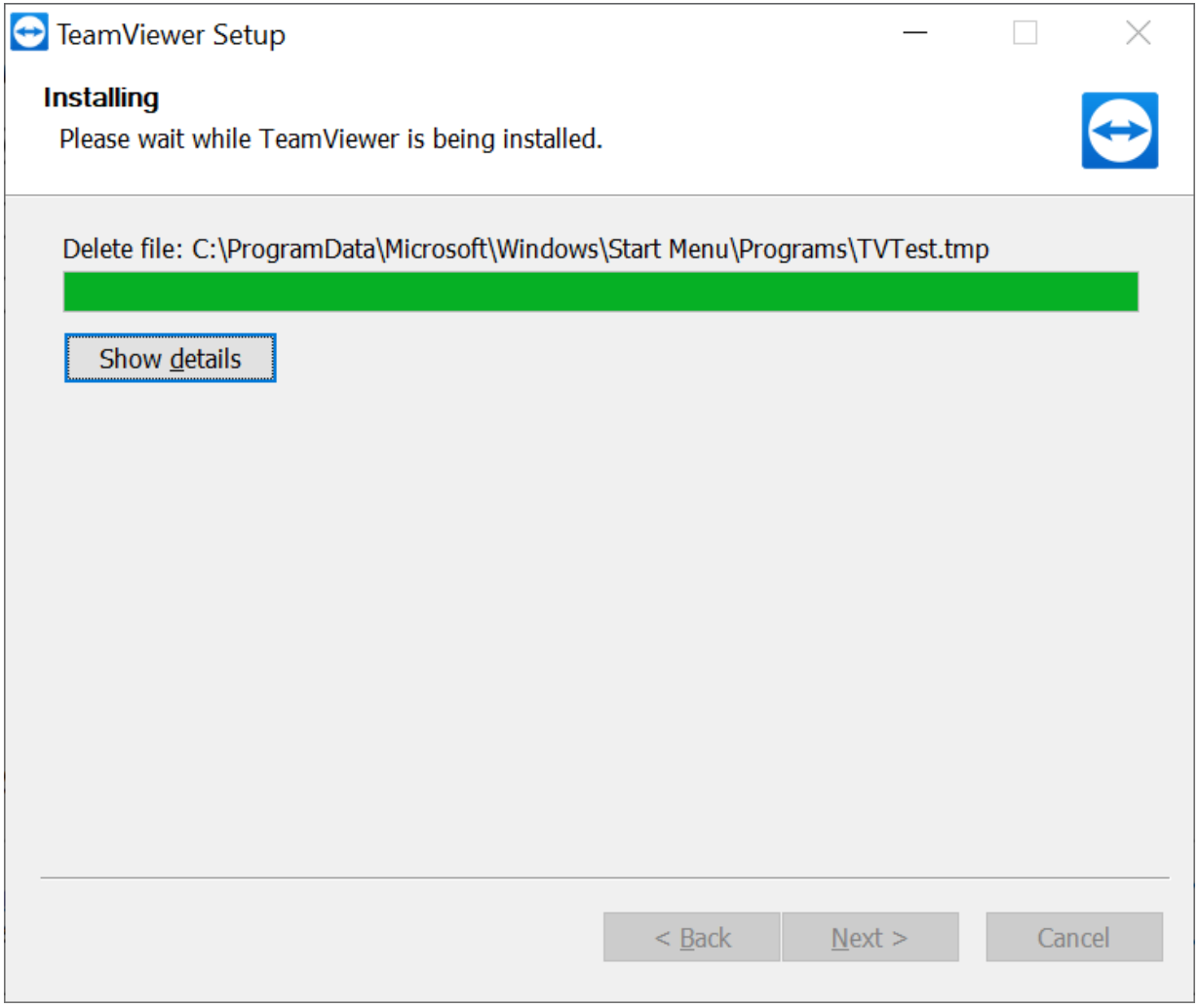


TeamViewer: A Collaboration Tool

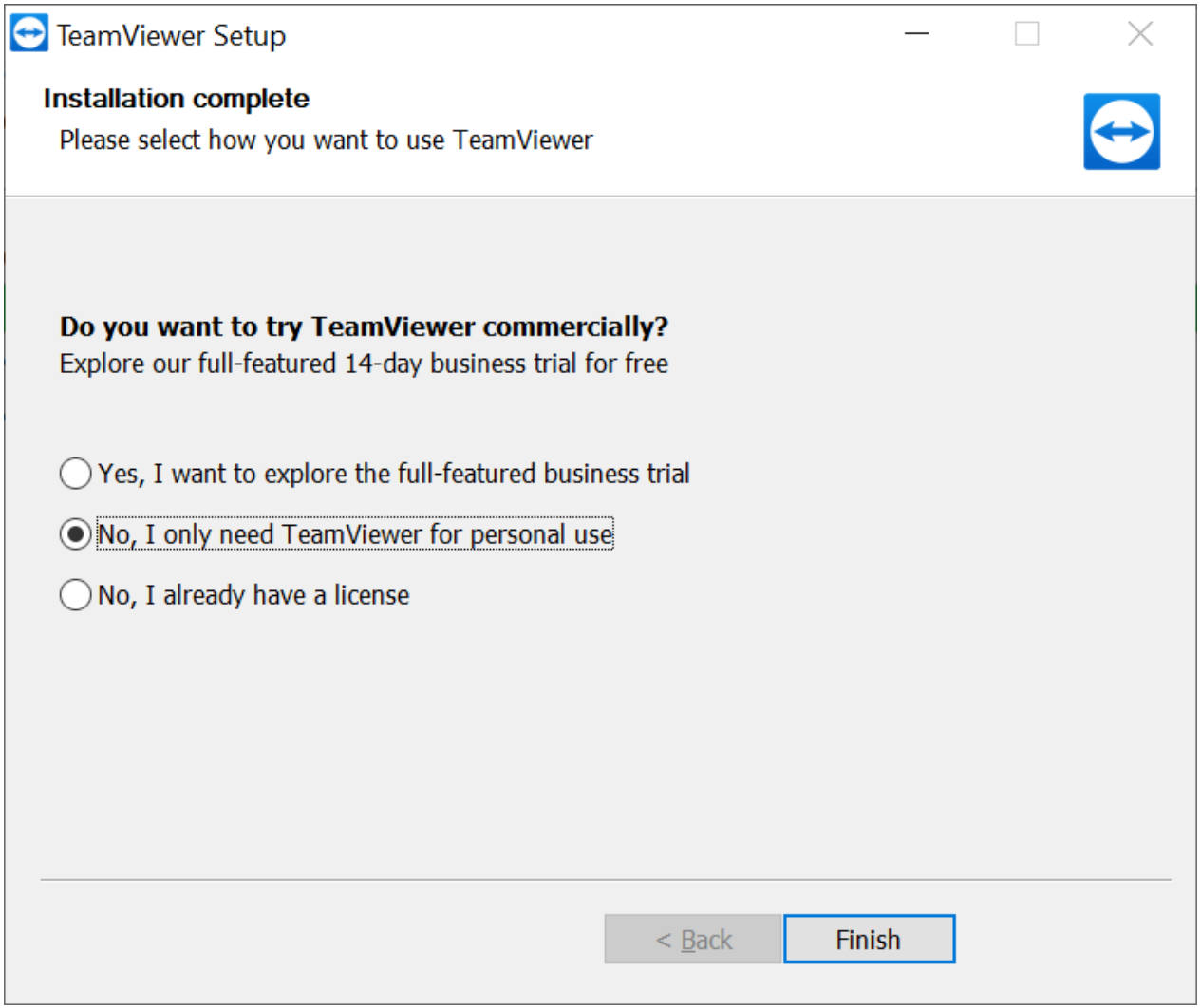
Download from https://www.teamviewer.com/en/download/windows/



TeamViewer: A Collaboration Tool



TeamViewer: A Collaboration Tool



TeamViewer: A Collaboration Tool