TEAMS

Idea in brief

Everybody talks about being a 'team player', but the reality is that teams don't actually work as well as we would like to believe. Improving how the team functions is essential in order to generate the collaborative and dynamic response demanded by software development.

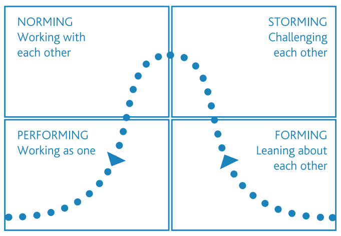
De Marco and Lister studied failure in software development projects and found that larger projects (with larger teams) failed more frequently. The cause of these failures was attributed to 'soft' problems – communication failure, high staff turnover, lack of motivation… These are team failures, not technological failures.

Although some subjects cause divided opinion, teams aren't one of them. Management experts and writers agree on a set of features necessary for building and operating as a successful team.

Teams are built around problems and as such should have an external focus. It follows that teams that are internally focused fail, whereas those that maintain an external focus succeed.

By setting up our teams correctly, we give them the best chance of solving the problem. We can then examine and improve the function of our team, maximising its effectiveness.

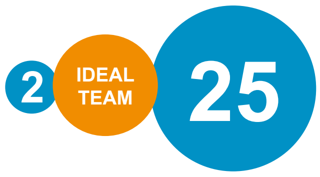
Ideas in practice



"Coming together is a beginning; keeping together is progress; working together is success."
Henry Ford

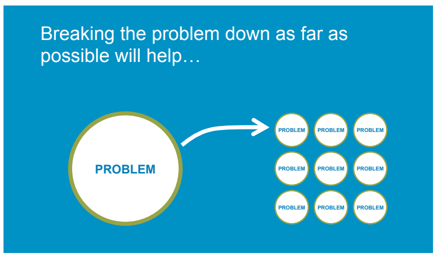
- The Tuckman Model shows how teams evolve.
- Teams need a chance to become teams and build their performance over time.

What's the ideal team size in software development?



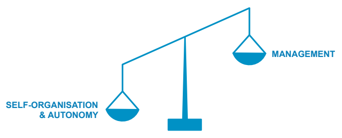
- Coordination costs rapidly rise to unsustainable levels as teams grow.
- Keep teams as small as possible.

"Virtually all effective teams we have met, read or heard about, or been members of have ranged between two and 25 people."
John Katzenbach



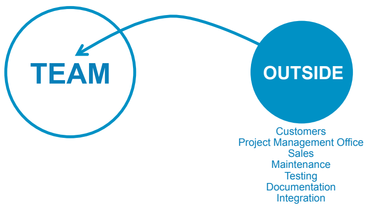
- Large problems are not best dealt with by large teams.
- Split the problem into smaller, well-defined pieces which can be tackled by independent teams that are not dependent on sequential elements.

Enable self-organisation



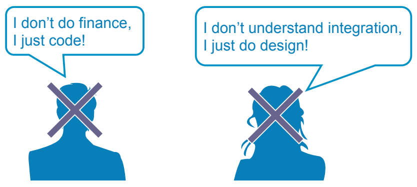
- Establish clear boundaries for your team.
- Teams can then avoid running into 'invisible electric fences' - the limits of their authority and operation.
- Allow teams to self-organise and be autonomous.
- Trust teams to do the work.

MANAGE EXTERNAL DEPENDENCIES



- We need to internalise external dependencies such as bringing an accounting function into the development team.
- The team then becomes more independent, autonomous and cohesive.

Create cross-functional teams


I don't do finance, I just code!
I don't understand integration, I just do design!

- Embed individuals from purely functional teams into cross-functional teams to give a consequential purpose.
- Once a team has a consequential purpose, the people will be driven in a compelling direction.