Software for Oneself

The most efficient scenario for building software is when one builds software for oneself: The requirements and user acceptance are in the same mind as the designer and build and deploy engineers.

Reality: The need to scale

Obviously, a single person scenario has limited applicability, and even a team of 1 for a separate customer does not scale to more complex high quality software.

As size and complexity grow and the usage scenarios become more diverse, there is a need to enlarge teams, and the need for communication and documentation to hand off work and ideas resulting in additional overhead. A conceptual core of B/IT Velocity is that improving alignment of ideas without duplication, and with efficient tools, the size of the scope and complexity of work can be increased without increasing the size of the team. Further B/IT Velocity builds up domain knowledge and reduces cognitive load allowing individual practitioners to increase breadth allowing larger projects to become feasible.

Some of the need to increase team size is based on required domain knowledge, especially in requirements definition. Subject matter experts are not necessarily members of the team, but the work required to engage them grows with the complexity of the business need.[1]

There are additional inefficiencies that creep into work as team size increases such as difficulty accommodating the individual needs and goals of the participants.

There is much written about team size in the context of Agile.

- Two-Pizza Teams - Introduction to DevOps on AWS
- Development Team Size | Scrum.org
- https://www.centigrade.de/en/blog/the-number-seven-is-not-magical-part-1/
    - I have some doubt to follow up on regarding the connection between chunking in short term memory and team size. It seems like a stretch.

[1] Engaging SMEs to find important nuances in stakeholder needs is a form of customer learning that can take time, and should not be confused with a loss of agility. Quite the opposite, it should be considered a form of low cost “customer” learning. See Cheap Learning

Compressing Step Deliverables.

Prescribed documents B/IT Velocity tools should be flexible enough to allow for the communication of any adjacent steps to be compressed into a single deliverable, and content and ideas from one level must not be duplicated in works at other levels.