AGILE

The evolution of agile frameworks

MANIFESTO FOR SOFTWARE CRAFTMANSHIP (2009)

As aspiring software craftsmen, we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

- Not only working software, but also well-crafted software.
- Not only responding to change, but also steadily adding value.
- Not only individuals and interactions, but also a community of professionals.
- Not only customer collaboration, but also productive partnerships.

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

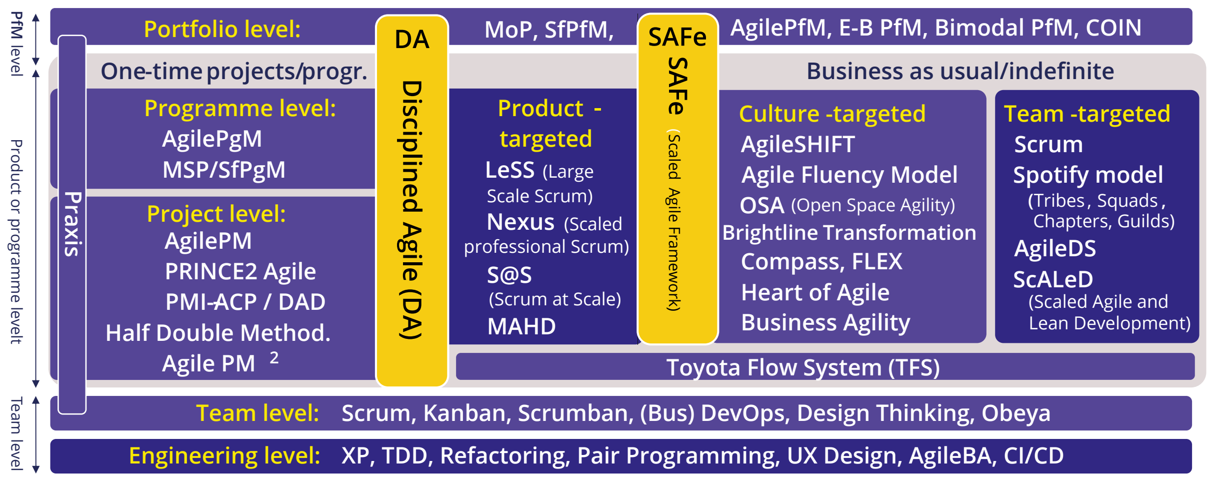
Introduction
My estimate is that there are at least 100 agile project, program, and portfolio management approaches. I have closely followed the development of agile approaches and build my 'Bird's eye view on the agile forest' (see figure) and found an evolution of approaches:

- Pilot phase: supporting one team
- Engineering phase: focus on skills within a team
- Scaling up to multiple teams: each team autonomous, permanently collaborating teams on one product or service, temporary structure with permanent or non-permanent teams
- Scaling up to organization: portfolio management
- Repair phase: focus on the agile mindset, the agile culture

Pilot phase: supporting one team
This phase already started at the end of the 20th century. The Manifesto for Agile Software Development, agreed by 17 people in Utah (2001), consisting of four values and twelve principles to formulate an answer to the much too slow software development. The Agile Manifesto brought structure and is the starting point for most of the agile approaches.

Some key characteristics for agile ways of working supporting one team are:

- Permanent or temporary. Is the team set up for a temporary period (as part of a project) or is it a permanent team using agile ways of working? In the last situation you could say "we bring the work to the team" and the product owner could put it on the backlog.
- The size of the team. If the team gets too big, cooperation between team members is made difficult by the number of possible lines of communication. A team size of 5 to 9 members is considered optimal.
- Team members. Team members must be among other things motivated to deliver, must be able to handle ambiguity and must have a high degree of agreeability.



A BIRD'S EYE VIEW ON THE AGILE FOREST. SOURCE: SCALING AGILE IN ORGANISATIES - HENNY PORTMAN.

- Team autonomy. How autonomous can the team be. What agreements are made regarding decision making. When can they make decisions on their own, when do the need to escalate?
- Timeboxed or continuous flow. Will there be a delivery heartbeat of one or two weeks (sprint or timebox) or will the team deliver piece of work by piece of work (backlog item by backlog item).

Some examples of frameworks or methods supporting one team are: Scrum (timeboxed), Kanban (continuous delivery), and DevOps.

Engineering phase:
focus on skills within a team

At a certain moment it became clear that the team's participation in a Scrum or Kanban training program was not enough to achieve the desired results. This was clearly stated in the Manifesto for Software Craftmanship.

To be successful you must have a suitable environment e.g., in the cloud with tools like Azure or Amazon Web Services. Your application architecture must support timeboxed or continuous flow delivery (ever tried to run a three-month user acceptance test in a two-week sprint?). This means you need a continuous integration and continuous deployment (CI/CD) pipeline and/or a microservices architecture.

Besides the suitable environment you must have team members who must be able to work in that environment. These team members, now often called engineers, must have the right environment skills as well as the skills to use all the supporting tools (e.g., GitHub, GitLab, Chef, Jenkins, Argo).

On top of these environment skills the team members must understand different way of working e.g., test driven development, pair programming, and refactoring.

Scaling up to multiple teams

Each team autonomous, permanently collaborating teams on one product or service, temporary structure with permanent or non-permanent teams.

Now an individual team can show the added value of using an agile approach you could make the next step to scale up to multiple teams. Here we find three groups of approaches. Each team is autonomous, different teams must work together to deliver a complete product or service and the situation that you still have a temporary (project) structure with temporary non-agile teams as well as (permanent) agile teams.

To start with the last one. What do you need to do when you want to bring a request to a team but there is no team to accept the initiative, or the team can't implement the change? You must build a team and bring the team to the initiative. And who can build such a team? I would say a project or program manager. I see many organizations making a transition to an agile way of working and firing their project and program managers. Within six months, project and/or programme managers are hired again, sometimes with a different job title, but doing the same thing. Some examples of »»

”Learn the rules like a pro, so you can break them like an artist” – Pablo Picasso

frameworks or methods supporting this temporary hybrid structure are PRINCE2 Agile, AgilePM, AgilePgM, and Disciplined Agile Delivery (DAD).

It’s also possible that you need more than 9 people to develop and maintain a product or service. In that case you must build a team of teams. But there is a limit how many teams can work together. The British anthropologist Robin Dunbar found a correlation between primate brain size and average social group size and that’s known as Dunbar’s number. Dunbar’s number is a suggested cognitive limit to the number of people (approximately 150) with whom one can maintain stable social relationships—relationships in which an individual knows who each person is and how each person relates to every other person. Agile approaches supporting team of teams offer coordination mechanisms between the teams and facilitate dependency management. To make integration of each team’s work possible there will be a cadence and synchronization between all the teams. Differences between the approaches are related to the involvement off all team members or only representatives during planning sessions, the usage of single or multiple timebox horizons (e.g., SAFe with team and program iterations versus LeSS or Nexus with a single timebox horizon), the usage of a single or cascaded backlogs (e.g., SAFe with epic, program, team backlog versus LeSS or Nexus with one backlog and the usage one product owner or hierarchy of product owners).

In case you need more than 150 people to build and maintain one product you need a team of teams of teams’ structure. I have seen in the medical equipment industry set-ups with 30 to 40 teams working on one product (physical parts, hardware, software, commercial material etc.). Examples of these approaches are SAFe, LeSS Huge, and Nexus.

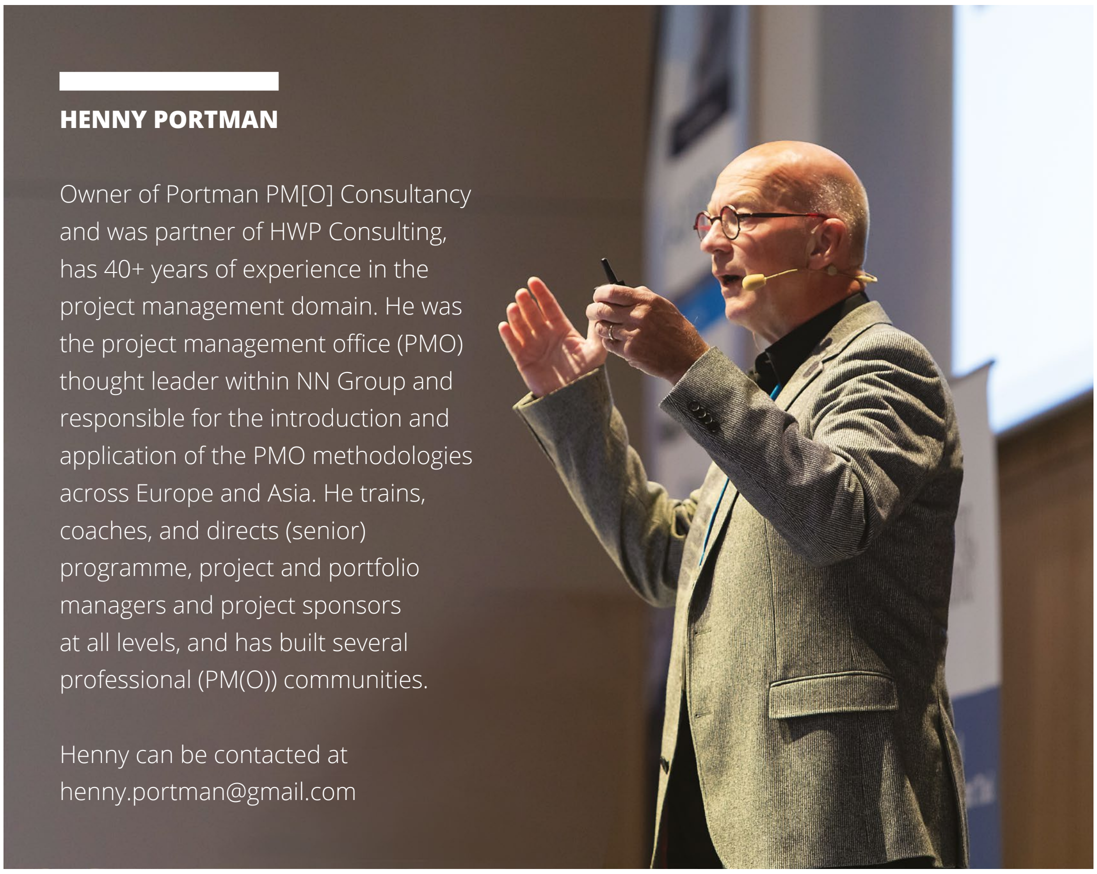
In case of autonomous, independent teams there is no need for synchronization, cadence, or dependency management. These teams can choose their own way of working and have their own product owner. Some examples of frameworks or methods to support these teams are AgileDS and Spotify.

Scaling up to organization: portfolio management

Now that it has been scaled up to the teams, the next phase is to look at what those teams are doing or should be doing. You also need agile portfolio management frameworks to facilitate this process. Probably an approach that supports both permanent agile teams and temporary projects is the most obvious. And actually, existing portfolio management frameworks or methods can handle this as long as the horizon over which commitments are made is not too long (e.g. quarterly) so that changes can easily be incorporated. Aspects on which a distinction is made are the manner of prioritization (e.g., multi criteria analysis, MoSCoW, WSJF (relative estimation) or the usage of an integrated framework/method (team, product, portfolio) or specialized framework/method. Some examples are MoP, AgilePfM, SAFe, and Disciplined Agile (DA).

Repair phase: focus on the agile mindset, the agile culture

Agile working methods are not always successful. 70% of agile transformations fail. You wonder: how is this possible while the neighbors seem to be working? And therein lies part of the answer. Every organization is different. What works for one may not work for another. Culture makes



HENNY PORTMAN

Owner of Portman PM[O] Consultancy and was partner of HWP Consulting, has 40+ years of experience in the project management domain. He was the project management office (PMO) thought leader within NN Group and responsible for the introduction and application of the PMO methodologies across Europe and Asia. He trains, coaches, and directs (senior) programme, project and portfolio managers and project sponsors at all levels, and has built several professional (PM(O)) communities.

Henny can be contacted at henny.portman@gmail.com

Kuva: Olli-Pekka Latvala

or breaks your agile transformation. Or as Spotify's CEO puts it: "You are all welcome to come and see how we work, but if you want to copy the Spotify model you have to become Spotify". One of the main reasons is the fact that the organizational culture does not match the necessary agile culture and mindset. There must be psychological safety within and between teams. Where possible, teams must be allowed to make their own decisions (decentralized decision-making).

This last phase supports this agile culture and emphasizes what ultimately matters: the cooperation between people. Without cooperation there is no team and therefore no result. These approaches also support the creation of uniformity, rhythm, and necessary roles. Some examples are AgileShift (Axelos) and OpenSpace Agility (OSA).

Future

In the coming years data, big data, data analysis and artificial intelligence will become more and more important. I foresee a next hype with data and AI driven project management methods.

Method war

Does this evolution justify the number of approaches, I would say no. Ivar Jacobson calls this the Methods War. Methods protect their own practices. Others cannot use them 1:1 but must be rewritten to fit within the method. They are trapped in their own method and fight tooth and nail by a 'guru'. In addition, not all methods are supported for practical application. Furthermore, I think that commercial interests also have an enormous influence on the development of approaches. An example: Ken Swaber and Jeff Sutherland, the founding fathers of Scrum, have always claimed that Scrum is enough to build and maintain products. However, I think, they saw the commercial success of SAFe and LeSS and each came up with their own approach with Nexus and S@S, including websites, guides, training courses, certification programs and so on. **PM**