The Agile Knowledge Projector: Enhancing the Adaptive Creation of Knowledge

Tiziano Villa
The Project Management Lab®, Monza, Italy

tiziano.villa@pmlab.it

Abstract: Agile is knowledge-sensitive. Knowledge feeds the entire adaptive life cycle from envision to transition. Main agile frameworks (i.e., SAFe®), toolkits (i.e., DA® – Disciplined Agile) and standards (i.e., PMBOK® 7th Edition) underline in different ways and at different levels of detail the relevance of knowledge for generating business value. So, all right? Everything is fine? Are we on the same page? In principle yes, but there are three main knowledge lacks: implicit knowledge, knowledge framework, knowledge roles/responsibilities. Therefore, negative effects are determined at different ontological levels: project performance, team dynamics, personal development. Presently, the agile landscape shows a relevant awareness of the topic of knowledge. However, this awareness is fragmented, ambiguous in key concepts, poorly addressed, even misleading in some cases. Knowledge in the agile environment is too often a fuzzy topic. The self-dimension of an agile team (self-directing, self- organizing, self-learning, self-empowering) erroneously assumes that knowledge is well addressed, wasting time, money, resources, decreasing performance and results. This paper provides a practical framework and tips for avoiding these negative effects. The AKP® – Agile Knowledge Projector might be a practical solution for better addressing knowledge in an agile project environment, improving value. AKP® is an organizational framework that fosters the creation of a dynamic mix of knowledge, combining in a continuous way explicit organizational artifacts, implicit team patterns and tacit individual expertise, for the entire duration of the agile project. AKP® consists of nine processes, focused on main agile dynamics of a complex and innovative project. AKP® is a powerful but not self-regulated framework: thus, it needs to be carefully designed and administered, according to specific knowledge roles and rules.

Keywords: Agile, Self-organizing team, Knowledge framework, Implicit knowledge, Agile knowledge creation

1. Sketching the Scene

Agile is knowledge-sensitive. Knowledge feeds the entire adaptive life cycle from envision to transition. Main agile frameworks (i.e., SAFe®), toolkits (i.e., DA® – Disciplined Agile) and standards (i.e., PMBOK® 7th Edition) underline in different ways and at different levels of detail the relevance of knowledge for generating business value. The concept of knowledge is embedded in the definition of agile, in that agile is "The art and science of facilitating and managing the flow of thoughts, emotions and interactions in a way that produces value outcomes under turbulent and complex conditions" (De Carlo, 2004). And again, agile is "The work of energizing, empowering and enabling project teams to rapidly and reliably deliver business value by engaging customers and continuously learning and adapting to their changing needs and environments" (Augustine, 2005). Thus, knowledge is an essential ingredient of this flow towards value. A topic, commonly pointed out by all references, is the concept of tacit knowledge. Barry Boehm states that "Agile methods derive much of their agility by relying on the tacit knowledge embodied in the team, rather than writing the knowledge down in plans" (Bohem, 2002). According to PMBOK® "Tacit knowledge is comprised of experience, insights and practical knowledge or skill. Tacit knowledge is shared via networking, interviews, job shadowing, discussion forums, workshops" (PMI, 2021). As for many other topics, also for what concerns knowledge Agile draws from Lean. For example, DA® recalls the principle "Create Knowledge" of the Lean Mindset: "Planning is useful, but learning is essential. We want to promote strategies such as working iteratively, that help teams discover what stakeholders really want and act on that knowledge. It's also important for team members to regularly reflect on what they're doing and then act to improve their approach through experimentation" (Ambler Lines, 2020). Similarly, the Scrum Guide states that "Scrum is founded on empiricism and lean thinking. Empiricism asserts that knowledge comes from experience and making decisions based on what is observed. Scrum engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed" (Schwaber Sutherland, 2020). So, all right? Everything is fine? Are we on the same page? In principle yes, but there are some knowledge lacks.

2. Knowledge Lacks

Sketching the current scene of the agile landscape, there are three knowledge lacks.

2.1 The Lack of Implicit Knowledge

It is probably the most significant. The current focus is limited to the dichotomy “Explicit VS Tacit”, not taking into account the implicit dimension of knowledge. To clarify, it is worth briefly summarizing the profile of the three types of knowledge:

Explicit (represented). Ontological level: organization. Explicit Knowledge (EK) is based on formal logics, codified, written, context independent, asynchronous (who produces EK doesn't know in advance if it will be used, when, by whom, in which way), it's hard to modelize, easy to communicate and transfer. EK must considered as a stock, as a "universal rule" for the entire community. An example of EK is the formula for computing the area of a rectangle (width times length). Other examples of EK are documented best practices, formalized standards, codified rules by which a customer order must be processed by the company, the official theory test for car drivers, the PMBOK® 7th edition in the world of project management (from predictive to adaptive and everything in the between). EK is already available before its use (i.e., criteria for structuring the WBS - Work Breakdown Structure to be included into the scope management plan, or prioritization techniques such as WSJF - Weighted Shortest Job First for prioritizing the backlog), It formalizes the progressive outputs, produced during its use (i.e., scope baseline, including WBS and WBS dictionary of the specific project, or user story syntax “As a… I want… So that…”), it documents the final results after its use (i.e., formal documentation for accepted deliverables and user manuals of the new solution carried out by the project).

Implicit (embedded). Ontological level: team. IK - Implicit Knowledge (IK) is a collective interpretation of EK in a specific context by a specific community that share a common goal. For example, the community of car drivers is asked to put in action a collective IK for driving successfully in the traffic of a certain city at a certain time. Driving in the traffic of New Delhi at rush hours is quite different than driving in the traffic of Zurich at midnight. IK is less collective and more dynamic than EK. IK resides in systemic routines, organizational culture, codes of conduct, ethics, social behaviour, habits, social norms, heuristics. IK is a property of the system as a whole, rather than a personal feature of individuals belonging to the system. IK is a collective knowledgeable answer to a contextual challenge. It's an unwritten and unspoken agreement for coping with a shared situation or accomplishing a result together. According to an adaptive (agile) approach, for estimating user stories, development team and product owner play a planning poker session, starting from planning poker rules and Fibonacci sequence (EK). During the session, development team and product owner implement planning poker rules based on collective patterns (IK). If similar situations come into view, the same pattern of IK will be typically put in place. If project environment changes (i.e., less time for planning, unavailability of the scrum master, large compound stories), implicit practices change consequently.

Tacit (embodied). Ontological level: individual. Tacit Knowledge (TK) is personal in nature, subjective, intuitive, based on physical experiences ("flight time"). It's context dependent and emotional (beliefs, insights). It's a continuous flow, it's easy to protect, but hard to gain, tell, write down and transfer. TK bodily lives with the individual and is deeply embodied in it. Expert judgment is a classic example of TK, widely applied for managing a project. For example, a senior specialist will troubleshoot a big problem based mainly on his/her experience, insights, intuition and secondly on codified standards. At the same time, it would be quite difficult for him/her to translate his/her TK into a document (EK), to be shared with a junior specialist (the document is the top of the iceberg, the expertise is the iceberg). The amount of TK owned by an individual is directly proportional to the amount at stake (time, money, effort, trust) for replacing the individual. TK is fully contextual: it’s might come to mind only against an intentional need of knowledge. TK might be shared through physical proximity, F2F interactions, learning by intrusion, observation, imitation, joint practices, osmotic communication, but first of all through a real trust and willingness to live common situations closely together. TK is “in the doing”. Refer to the famous example of TK used by Polanyi “Being able to recognize a person’s face but being only vaguely able to describe how that is done” (Polanyi, 1966). TK cannot be articulated; it consists of a set of personal models, fully played, less told and more less written.

Summing-up, PK – Project Knowledge is “A dynamic combination of explicit artifacts, implicit patterns and tacit expertise, applied for developing options, taking decisions and implementing actions throughout the project life cycle" (Villa, 2015). This definition underlines the "contextual mix" of different types of knowledge, to be used intentionally to achieve project objectives. Four are the key features of an effective PK: a) contextual (tailored on project peculiarities, individuals and interactions included), b) intentional (focused on project options, decisions and actions), c) dynamic (progressively elaborated and shared within the project environment), d) combined (based on different project cognitive elements). Assuming that one of the four core values of Agile Manifesto is still valid, “Individuals and interactions over processes and tools”, then

implicit knowledge is essential in that agile team feeds on and thrives on implicit knowledge. Agile ceremonies are founded on implicit knowledge. For example, the effectiveness of the daily stand-up meeting depends only marginally on ground rules (EK) and number of daily meetings attended by each member of the agile team (TK). Effectiveness mainly depends on the implicitpattern that the team as a whole put in place collectively, adapting it from time to time to individual circumstances. Without implicit knowledge, the performance of the team is very similar to that of an organ in fibrillation, where each part moves uncoordinated from the others.

2.2 The lack of a Knowledge Framework

Agile movement has addressed the "knowledge" topic since its origins. Some examples:

referring to the Manifesto of Agile Software Development, 2001, the principle n.12 declares that "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly" (PMI, 2017). While the principle n.6 declares that "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation" (Ibidem). Indeed, these principles provide clear guidelines on how to reserve spaces dedicated to the exchange of knowledge;

the Scrum method entails ceremonies focused on knowledge, that's to say "Daily Scrum" and "Sprint Retrospective" that must be performed regularly;

the Lean-Agile Principle #8 by SAFe® "Unlock the intrinsic motivation of knowledge workers" (Leffingwell, 2018) recommends to invest in enhancing the personal knowledge owned by knowledge workers, through their motivation for autonomy, mastery, mission and minimum possible constraints;

"Choose your Wow!" by DA® entails the specific Decision Point "Improve skills and knowledge", according to an extended version of the hierarchy of competence by Burch, based on five levels of a learning journey for a given skill or knowledge area.

However, a knowledge framework is missing. In other words, we have so many pieces of the puzzle without a frame. Which are the relationships among knowledge transfer, knowledge sharing and knowledge creation in an agile environment? How to link and mix the explicit, implicit and tacit dimensions of knowledge? The answer is excessively discharged on the discretion, on the maturity, on the self-regulation of individuals and teams. On the other hand, a shared knowledge framework could be the basis on which knowledge-centered behaviours and dynamics can be supported.

2.3 The lack of Knowledge Roles/Responsibilities

The agile environment is crowded with many roles. Different agile methods and scaled approaches entail different terms for the same role, such as scrum master, team lead, agile coach. Indeed, this proliferation of terms is consistent with the agile movement that in a spontaneous and uncoordinated way has come to converge on shared cornerstones. Some roles are focused on business (product owner – do the right solution), other ones on process (agile coach – do the solution fast), other ones on design (architecture owner – do the solution right), other ones on contents (development team – capability and capacity), other ones on governance (project sponsor – strategy). In a knowledge-sensitive agile environment, the lack of knowledge-focused roles is deafening. A knowledge role does not mean that this role must perform knowledge activities; it means that this role should act as a knowledge hub. If we have an architecture owner, why can't we have a knowledge architect? Similarly, if we have an agile coach, why can't we have a knowledge catalyst? Of course, these questions are not aimed at creating additional organizational bureaucracy, within an environment based on lean principles. The point under discussion is "how to lead knowledge in an agile project?" A tight option is to formalize specific knowledge roles within the agile environment. Thus, an enlargement of the agile team with people that are in charge of the topic "knowledge". A lightweight option is to assign to existing roles specific responsibilities and tasks about knowledge. In any case, taking the ownership of knowledge creation is an open challenge that must be addressed.

3. The Agile Knowledge Projector (AKP®)

AKP® is a knowledge framework focused on agile projects. It creates a dynamic mix of knowledge combining in a continuous way explicit organizational artifacts, implicit team patterns and tacit individual expertise, for the entire duration of the project. AKP® should make up to knowledge lacks described above. A proposal of AKP® is portrayed in Figure 1. Why a projector? A projector is an optical device that projects images mixing the shades of three primary colors (blue, yellow, red). Similarly. AKP® projects within the agile environment

knowledge items combining "three primary colors of knowledge". This way, the colored and changing light of knowledge pervades the agile project and illuminates its execution.

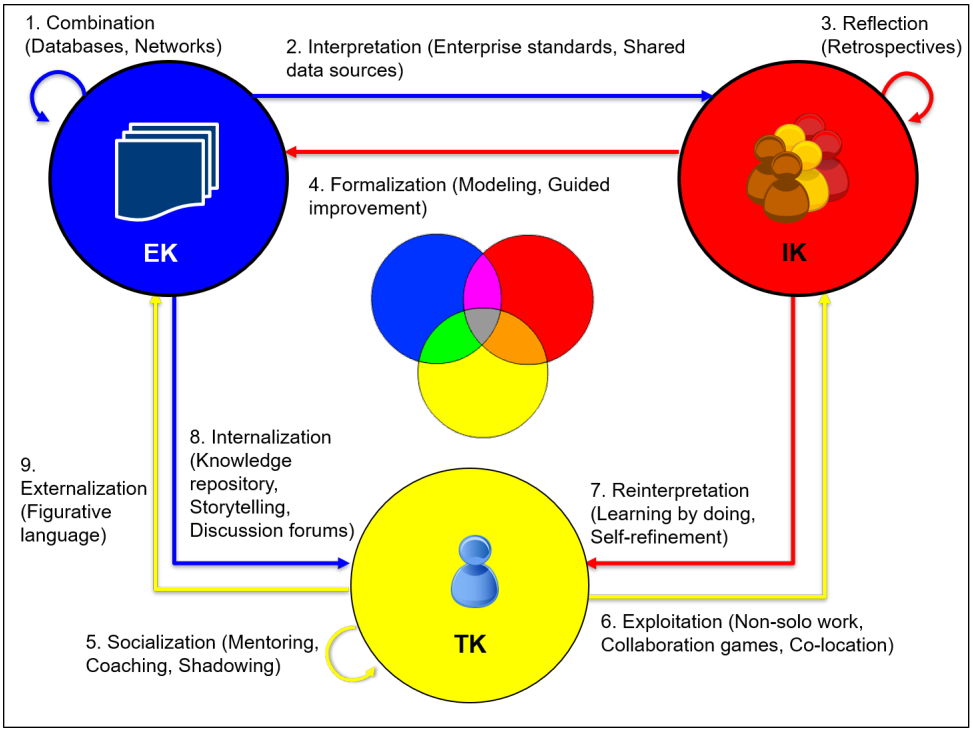

Figure 1: AKP® – Agile Knowledge Projector

AKP® entails three basic components. Each component is represented by a primary color: blue for EK (Explicit Knowledge), red for IK (Implicit Knowledge) and yellow for TK (Tacit Knowledge). The projector combines the basic colors to make a wide and useful spectrum of colors (knowledge). The primary colors are those which cannot be created by mixing other colors in a given color space. The purpose of an arrow is to create the knowledge on the arrow head using knowledge on the arrow base. The pre-existing knowledge on the arrow base is used and not consumed or modified. It must be pointed out that pre-existing chunks of knowledge, if well used, shared, combined, revised, can generate a new chunk of knowledge, that's to say something that is not the sum or the clone of previous ones. For example, the tacit knowledge personally owned by each member of the agile development team is used for creating a new implicit pattern, collectively owned by the team. An arrow may connect one type of knowledge with another type of knowledge, or it may be circular on the same type of knowledge. For example, pre-existing implicit knowledge collectively owned by the team may be used for creating additional team patterns. Each arrow refers to a specific knowledge process, such as "1-Combination of EK". The terms, one or more, written in brackets, such as (1- Databases, Networks) are the means by which knowledge is created. For example, a Mob Programming session, a type of non-solo work, during which "The team gathers around a single workstation, with one team member coding while others observe, discuss, and provide advice. The programmer is swapped our regularly and everyone codes at some point. The code is often projected onto a large screen" (Ambler Lines, 2020).

AKP® supports agile roles (i.e., Product Owner, Scrum Master, Development Team, etc.) to better set-up their agile environment and to better conduct agile ceremonies and agile work. For example, the Interpretation process may help the agile team to quickly agree its implicit WoW – Way of Working exploiting available explicit organizational artifacts. Similarly, the Reinterpretation process should allow the single agile team member to take the time to sort out the implicit experience gained working in team, and to refine his/her tacit agile way of thinking and acting. In summary, with AKP®, people called to work by projects on the basis of an agile framework have the opportunity to carry out their work more effectively, activating as needed knowledge management processes in a conscious and shared manner.

A brief comment for each of the proposed processes and means:

Combination (Databases, Networks). Regardless the specific agile project, a lot of explicit artifacts are available at the organizational level. They refer to databases that collect documentation from portfolios, programs and projects. For example, business cases, backlogs, lessons learned. Combination is the process that links and merges different bodies of explicit knowledge. Reconfiguration of existing artifacts can lead to create new explicit knowledge. For example, a central collection of risks encountered in agile projects for a specific industry or technology. Central functions, such as PMO, lead the combination, conversion, reconfiguration, creation of explicit knowledge.

Interpretation (Enterprise standards, Shared data sources). For managing a project with an agile approach, the team can count on a set of OPAs – Organizational Process Assets made available by the company. Particularly agile models (i.e., Gulf of evaluation), methods (i.e., Agile roles profiles, Value stream mapping, Story point estimating, Iteration planning, Iteration review), and artifacts (i.e., Information radiators, Team charter). Similarly, for Shared data sources, such as documentation from similar past projects. Using these explicit objects, the agile team can set-up its own implicit WoW - Way of Working.

Reflection (Retrospectives). Agile common experiences should be shared and improved. Retrospective is typically a two-hour meeting held at the end of each timeboxed iteration, during which the entire agile team (PO + SM + DT) reflects around processes (our way of working) and people (our behaviours and team dynamics). Retrospective is performed by the team for the team. Questions under discussion are: "What worked or went well? What caused problems, failed to work properly, or did not go well? What can be done differently in the next sprint?". Retrospectives improve implicit knowledge owned by agile team for the remaining part of the project. Some improvement work items might be added to the backlog.

Formalization (Modeling, Guided Improvement). Agile team implicit patterns are regularly evaluated in order to improve team performance, for example with sprint retrospectives. This local view might provide insights useful at a higher organizational level. Thus, Guided Improvement based on PDCA/PDSA/OODA loops, should be applied to create formal explicit knowledge reusable by other projects throughout the company. For example, more focused criteria for grooming the backlog or more effective ways for engaging key stakeholders. Similarly, for Modeling: agile team creates reusable representations of processes, ground rules, ceremonies, personas that feed the company knowledge base of explicit agile resources. This way, new agile OPA is created.

Socialization (Coaching, Mentoring, Shadowing). Tacit knowledge of an individual grows by sharing the tacit knowledge of another individual and vice versa. It happens through socialization, that's to say sharing mental models and technical skills. Not necessarily using language, but through observation, imitation, shadowing (forward and reverse). During "coach office hours" an experienced individual makes himself/herself available so that other individuals can drop in on him/her to get help on a specific topic. Similarly, the mentor supports the mentee, providing psychosocial support and role modeling, about how to develop oneself as a whole person, coping with personal challenges. Socialization entails spending time closely together, according to the famous quote by Polanyi "We know much more we can tell" …. "and we can tell more we can write".

Exploitation (Non-solo work, Collaboration games, Co-location). Agile is based on individuals with a T-shaped profile (generalizing specialist). Individuals as owners of a personal amount of tacit knowledge that can make the difference. A sum of excellent individuals does not necessarily make an excellent team. Implicit knowledge owned by the team is more than the sum of the individual expertise. Implicit knowledge is created through shared working spaces for emerging relationships. Non-solo work entails pairing and mob programming. Collaboration games are another shared space within which development team perform agile activities such as the construction of the product vision box, the DoD – Definition of Done, the planning poker session. Co-location, physical or virtual, fosters the creation of implicit knowledge through osmotic communication and learning by intrusion.

Reinterpretation (Learning by doing, Self-refinement). Working together, members of the agile team create and act implicit patterns. They live a bodily collective experience tailoring agile events (i.e., sprint planning) on project peculiarities. Individual should re-elaborate the implicit knowledge gained by the team to enrich his/her tacit knowledge. It could happen through private spaces of personal reflection during which making a summary of the time spent with others. Similarly, re-experiencing in a personal way what shared with the team. For example, writing user stories, addressing risks, acting as a generalizing specialist, taking ownership, managing conflicts. Internalization is the key that embodies implicit knowledge into tacit knowledge.

Internalization (Knowledge repository, Storytelling, Discussion forum). The bridge from explicit to tacit is asynchronous, in that explicit artifacts that are available in the company knowledge base can be brought into the individual space of knowledge at discretion of the individual. A collection of pre-registered project tales may be listened influencing the personal mental model. Similarly, an individual can attend a discussion forum where past experiences are shared and commented. In both cases codified past knowledge is reused. Moreover, individual can access other explicit artifacts such as case studies, articles, templates, knowledge checks, deciding when and how internalize them.

Externalization (Figurative language). Probably the arrow from tacit to explicit is the most difficult to put into practice. The point under discussion is externalization, that's to say the process of articulating tacit knowledge into explicit artifacts. Tacit knowledge, by its nature, it's hard to articulate through analytical methods of deduction and induction. Thus, nonanalytical methods should be used. Try to conceptualize an image, a chunk of tacit knowledge, through literal language leads to inconsistent results. Much better through figurative language. Metaphors and analogies are the pillars of figurative language. Hyperboles and other figures of speech might be very useful too. An agile individual should learn to use in figurative language better and better. "The product owner is Janus Bifrons" (metaphor); "Working in high-performing agile team it's like being in a beehive with so many other hardworking bees filling the cells with honey" (analogy); "Blood must flow during the sprint retrospective" (hyperbole). This way, personal experience gained in agile projects is conceptualized in a synthetic and incisive way. And, most importantly, in an immediately usable way by the organization.

To be truly effective, AKP® should be designed and administered properly. For a large project a knowledge architect might be appointed. Otherwise, the design of AKP®, customized on the peculiarities of the agile project taken into consideration, should be a responsibility of the agile coach (scrum master, team lead), in that the knowledge framework is a part of the overall agile framework that this role must however already set.

Once set up, AKP® must be carefully administered in order to ensure the continuous creation of the dynamic mix of knowledge which the project feeds on. Due to its conformation (width, interconnections, user's engagement), AKP® is unable to self-regulate by itself. Administering AKP® means to verify the correct application of knowledge tools, to evaluate the actual creation of new knowledge, to foster the connections between explicit, implicit and tacit dimensions of knowledge. Because of this, a clear responsibility must be assigned. Again, the role of knowledge catalyst might be appointed, or this responsibility might be assigned to pre-existing role, particularly to the agile coach.

4. Conclusion

Knowledge is a key driver for an added-value application of agile. Presently, the agile landscape shows a relevant awareness of the topic of knowledge. However, this awareness is fragmented, ambiguous in key concepts, poorly addressed, even misleading in some cases. Therefore, negative effects are determined at different ontological levels: project performance, team dynamics, personal development. The AKP® – Agile Knowledge Projector might be a practical solution for better addressing knowledge in an agile project environment, improving value. AKP® covers three main knowledge lacks: implicit knowledge, knowledge framework, knowledge roles/responsibilities. AKP® is not self-regulated framework, but instead needs to be carefully designed and administered.

References

Ambler, S. W. and Lines, M. (2020) *Choose your Wow!,* PMI® Disciplined Agile.
Augustine, S. (2005) *Managing agile projects,* Prentice Hall PTR.
Bohem, B. (2002) *Get ready for agile methods, with care,* IEEE.
De Carlo, D. (2004) *Extreme project management,* Jossey Bass.
Leffingwell, D. (2018) *SAFe® Reference Guide 4.5,* Pearson.
Polanyi, M. (1966*) The tacit dimension*, Anchor Books
Project Management Institute (2021) *A guide to the project management body of knowledge (PMBOK® Guide) Seventh Edition,* PMI®.
Project Management Institute (2017) *Agile Practice Guide,* PMI®.
Schwaber, K and Sutherland, J. (2020) The Scrum Guide, www.scrum.org.
Villa, T. (2015) *Knowledge: definition, types, application in the project environment - V1.3,* PMLAB®.