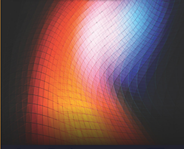
The TameFlow Hyper-Productivity Series

Hyper-Productive Knowledge Work Performance

The TameFlow Approach and Its Application to Scrum and Kanban



Steve Tendon
Wolfram Müller

4

MANAGEMENT'S PROFOUND UNDERSTANDING OF KNOWLEDGE WORK

Sample Chapter

The evolution of Scrum, arguably the most successful Agile method, has developed to the point where its proponents (and in particular Scrum's original founder, Jeff Sutherland) are actively trying to extend the use of Scrum to the organization as a whole, and not limit it to the development team alone. This is important, as it highlights a significant deficiency in Scrum: the lack of complete support for it by the entire organization and its leader.

A hyper-productive team can only exist within an organization that supports the team's effort *entirely*. The kind of organizational support that is required can only be found if the two noble patterns of *Unity of Purpose* and *Community of Trust* are pervasively present throughout the entire organization. For this to happen, it is necessary that the Chief Executive Officer (CEO) (or business owner) step up to a role of truly enlightened leadership, and that the organization as a whole stands behind the initiative.

We will say more about the importance of leadership later. We will also examine what is required for the organization and the team. For now, though, we want to focus on another dimension that is a precondition to enlightened leadership—that of *profound knowledge*.

PROFOUND UNDERSTANDING OF THE FUNDAMENTAL PROCESS

Deming, one of the most important thinkers about business and management, is known for sustaining the need for developing a profound understanding (Deming, 1993) about your business. It is of essence to possess a theory of knowledge about the *entire system* that constitutes your business.

It follows that anyone who owns, or is in charge of, a knowledge producing business, should have a *profound knowledge* about such knowledge development. In the case of software businesses, this is seldom the case. Except for those companies that were founded, grown, and led by individuals with a software development background, the majority of companies where software development is a critical part of business are led by people who do *not* have a software development background.

Unsurprisingly, in these cases, the obvious reaction to the claim that managers should have a profound knowledge about *software* is a total dismissal of the concept. The idea is deemed to be too technical, unrealistic, or unworkable and this is entirely justified because the impression is that managers need to know about all the technicalities and intricacies of that huge body of knowledge which is collected under the term of computer science and software engineering.

This impression, though, is a fallacy and in itself a misunderstanding. The assumption that a profound understanding about software development (or other knowledge-based work) requires deep technical knowledge is without any foundation.

The CEO and all of senior management can develop a profound knowledge about the nature of the knowledge-based process, without possessing any technical notion at all. In fact, those in charge of business management already possess the required skills and abilities to do so; but they lack the awareness about it.

The intent of what we are trying to achieve here is to make business managers aware of the fact that they are already *capable* of stepping up to the leadership role that is required of them to transform a knowledge-based organization into a hyper-productive one. People in upper management simply need to develop the awareness about themselves already possessing such capability and knowledge, and then consistently apply that insight to the management of the knowledge-producing organization. These are two distinct challenges, because creating awareness alone is not enough—managers must also develop the willingness to deliberately apply these capabilities to the daily management of the organization. Unfortunately, ingrained practices are working against this.

Sample Chapter

THE WICKED PROBLEM OF STRATEGY MAKING

The first step is to establish a frame of reference which can be recognized by management as something they already know and have deep familiarity with. We will look into *strategy making*. Surprisingly, strategy making has many things in common with knowledge-work processes.

In a Harvard Business Review article, Camillus (2008) presents a convincing interpretation of the nature of strategy making. Strategy making is seen as a *wicked problem*. The distinguishing trait of wicked problems is that they cannot be solved; yet it is possible to deal with them. Strategists handle such problems in practice.

Wicked problems were first described by Rittel (1973) and later by Conklin (2005). Wicked problems can be recognized because they are characterized by 10 properties which were described by Camillus (2008) as follows:

1. **There is no definitive formulation.** Unlike ordinary problems, wicked problems cannot be described by a precise *problem statement*.
2. **There is no stopping rule.** Unlike ordinary problems, which cease to exist once a solution has been found, wicked problems are ongoing, and the search for a solution never stops.
3. **There are no clear-cut solutions.** Unlike ordinary problems, wicked problems do not have solutions that can be impartially considered right or wrong. There might be multiple solutions, and the choice is ultimately a judgment call.

4. **Solutions cannot be tested.** Unlike ordinary problems, where the correctness of a solution can be proven immediately, the ways in which wicked problems are addressed have different consequences over time. It is difficult to evaluate their effectiveness.
5. **Solutions are "one-shot" operations.** Unlike ordinary problems, where solutions can be tried and dropped and it is possible to progress through trial and error, with wicked problems every attempted solution will bring about irreversible consequences.
6. **Solutions are not enumerable.** Unlike ordinary problems, which can be described through an exhaustive list of potential solutions, wicked problems can have an undefined number of potential solutions.
7. **A wicked problem is incomparable.** Unlike ordinary problems, which can be classified together with other similar ones for which there are similar solutions, a wicked problem is *one-of-a-kind*. Experience does not help resolving wicked problems, as there are no precedents.
8. **Wicked problems imply other problems.** Unlike ordinary problems, which are self-contained, wicked problems coexist with other problems; and there is no one single root cause that can be identified.
9. **Wicked problems have multiple representations.** Unlike ordinary problems, which can be described by a single uncontroversial statement, wicked problems can have multiple representations. Different stakeholders will have different perceptions and ideas of the problem and its causes.
10. **There is no right to be wrong.** Planners confronting wicked problems cannot afford failure because the impact of decisions will be so large and actions will be so costly that the problem solver will be held liable.

Camillus provides the caveat that these 10 properties should not be taken as criteria testing for wickedness; they should be considered as guidelines to allow one to judge if a problem is wicked. What is most striking is that there invariably is a social dimension that contributes to the wickedness. Camillus observes that wicked problems are more likely to appear when companies are exposed to major changes or extraordinary challenges. Notably, this all happens in a social context. The *degree of disagreement* among stakeholders is a sign of the wickedness of the problem. Hence, wicked problems are difficult to deal with not only because of the technical difficulties, but also because of all social complexity they cause. Camillus sees confusion, discord, and lack of progress as signs that an issue could be wicked.

Unsurprisingly, *social context*, *disagreement among stakeholders*, and *social complexity* are all traits that characterize all knowledge work and software development projects in particular as soon as we look beyond the technical field proper. In the field of software development, wicked problems have been recognized by Degrace (1990), and are ordinarily dealt with by practitioners in a number of ways. These are the key features that will allow executive management to recognize that they do indeed have not only familiarity, but also a deep understanding of the fundamental processes involved in knowledge work and software development because strategy making is typically a wicked problem.

Sample Chapter

Coping with Wicked Problems

With regards to strategy-related issues, Camillus identifies five characteristics that reveal the presence of a wicked problem:

1. There are many stakeholders with different values and priorities.
2. Causes are complex and tangled.
3. The problem is difficult to comprehend and changes with every attempt to address it.
4. The challenge has no precedent.
5. There's nothing to indicate the right answer to the problem.

Camillus advises coping with wicked problems in an illuminating and simple way. It is necessary to involve stakeholders, to make sure that all opinions are listened to, and to facilitate communications. Rather than becoming more systematic, companies can use social-planning processes and aim at generating a shared understanding of the issue and promoting a joint commitment about potential resolutions to try. Disagreement will be inevitable—the objective is to let stakeholders appreciate one another's positions. They should be able to reason about the different interpretations, and work collaboratively toward resolution. It is not only a matter of obtaining the ideas and opinions of all stakeholders; it is also a matter of involving them to find the solutions. Giving space to everybody to air their ideas helps to generate new perspectives, develops collective intelligence, and counters group-thinking and cognitive biases. The group as a whole will be better at confronting the problem than the individuals alone. Stakeholder involvement complicates coordination and planning, yet it increases the potential for creative solutions. The result is also buy-in and support from all involved parties.

Camillus further stresses the importance of actually documenting stakeholder's assumptions, ideas, and concerns as a means of communicating about the plans; and the planning process is seen as a vehicle for communication.

Empiricism at the Heart of Strategy Making

Since the world is complex, it is impossible to know which strategies will work and which won't, let alone what consequences might come out of them. Effective strategy making is less deterministic, and more empirical. Camillus explains that companies should experiment with a number of feasible strategies, even when the outcomes are very uncertain.

There is a risk that all strategists run, because the outcome might be counterproductive and lead to analysis-paralysis. This is exacerbated by the nature of a wicked problem, because every attempt at addressing it will alter it, and hence require yet another change in strategy. The risk is to keep analyzing forever, rather than doing something.

Therefore the best approach is empirical. It is better to test some strategy, and consider it as a starting point. The outcomes will reveal more information about the problem at hand. In other words, smart companies deal with wicked problems by running experiments and then learning from the mistakes they will inevitably make. Companies should encourage risk taking, even in the face of business failures. Unexpected, unsatisfactory outcomes contribute to the necessary organizational learning. As we will see, this kind of

Sample Chapter

advice is what is practiced when strategy making takes the concrete form of an emergent, iterative process.

Such a process has a profound social learning dimension at its heart. Organizational learning processes are at the heart of strategy making; likewise, they are at the heart of knowledge work. Recognizing this is the essence of executive management's gaining awareness of themselves already possessing the capabilities needed to manage a knowledge-based organization. What they need to understand and acknowledge is that knowledge work is really about organizational learning (and not only about technical and intricate topics). Let's see how we can gain that insight, with the help of two different approaches: one stems from economic theories and another from the parallel with collaborative performance arts.

CAPITAL GOODS AND SOCIAL LEARNING PROCESSES

The majority of business managers have a background in economics. Many will have earned MBAs, and have familiarity with economic theories. There are many economic theories, and most seem unrelated to knowledge work or software development. Luckily, there are some that have profound significance with respect to what happens (or should happen) in a healthy knowledge-work organization. Let's examine one economic theory's view of a software development organization, which we consider as representative of any knowledge-work organization.

In Baetjer (1998) we find the viewpoint of a capital theorist of the Austrian School—the viewpoint that software products are considered as capital goods. According to this school of thinking, any capital goods are an embodiment of knowledge. Specifically it is knowledge about how to execute some kind of production.

Because goods are seen as embodied knowledge, there is an important social dimension involved in their development; a social dimension that centers around organizational learning. The reason is that such knowledge is spread among many individuals, and mostly it is tacit, incomplete, and continuously in a state of change. New capital goods are necessarily developed through a process that is a social learning process, which brings the accumulated knowledge to fruition.

The need to capture this dispersed knowledge through a social learning process and embed that knowledge into a product or service is, in essence, the high-level process that governs the development of any knowledge-based product or service. Management executives need to internalize this insight, and thereby become aware of the fact that the process they use for strategy making (the social learning process), is really the same process they should use for managing any knowledge-based organization.

Knowledge about Product and about Process

Baetjer (1998) further extends the connection with capital-goods theory by examining the structure of capital goods and the relationships between capital goods. The relationships between capital goods can be considered as:

- A relationship of **complementarity**: when different capital goods are used together.
- A relationship of **dependency**: when one capital good is used to produce another.

Sample Chapter

The categories of capital goods mentioned are two: fixed capital and working capital. In the early days of computing, things were simple. The programming language (and its compiler) was the only fixed capital to care about. There were no intermediary artifacts to work with. Programmers literally started with a blank screen.

However, that has changed today with an increasing variety of programming tools and software technologies. Now even working capital can be recognized in the form of class libraries, design patterns, components, services, and many other artifacts which can be used directly or adapted for the programmer's intent.

Recognizing a distinction between fixed capital and working capital in the software development process suggests this important viewpoint: all software infrastructures (such as operating systems, database management systems, networking software, etc.) chosen to support your business can be classified as working capital that is used for producing your business software solution. A similar consideration can be made with respect to all digital tools and digital representations that are commonly used in knowledge-work.

In addition to complementarity and dependency, a third significant relationship is identified as the relationship between the items of capital goods and the knowledge about the process needed to produce them.

This last point is very significant as it highlights the importance of knowledge in the capital goods theory. The knowledge is not only that which is embedded into the products being made, but it is also that knowledge that is about the process used to make the products. The entire infrastructure (your technology stack) is ordinarily considered as an asset that you acquire; but in this economic theory it is considered as working capital.

It is like the raw material that you transform by developing your own work (code, in the case of software) on top of it, adding economic value. The technology stack can be considered as the input to your transformation process, while the product or service delivered is the result of the labor of your organization (whereby with organization we mean anybody involved in the transformation process, and not only the engineering or hands-on teams proper).

The labor that realizes this transformation process is effectively the transformation of dispersed knowledge represented by the collective mind of your organization into embodied knowledge that is encoded by the product or service being delivered. The transformation process itself results from the application of the collective knowledge about the production process.

Capturing all of this collective knowledge from the individual intellects involved implies a social learning process, whereby the knowledge is transferred, elaborated, and enhanced between individuals before it finally becomes a tangible product or a service that can be experienced by customers. At the same time, the collective mind of the organization will develop knowledge about the process it employs.

Sample Chapter

STRATEGY MAKING, ARTFUL MAKING, AND SOFTWARE DEVELOPMENT

The insight to be gained is that management needs to recognize that the fundamental process involved is the one of social learning. It is refreshing that a capital theorist states that producing software is essentially a social learning process. It is also notable that this happened three years before the Agile Manifesto (Beck, 2001) was published.

The social learning dimension in software development has been highlighted by many proponents of Agile methodologies. Most notably, Cockburn (2001) sustains:

> [Software development is] a cooperative game of invention and communication.

This aspect, the cooperative game of invention and communication, is what allows us to bridge software development not only to the theoretical economic model of capital goods theory which might be familiar to some managers, but also and more important for the practical consequences to the actual daily experience that managers have. In fact, senior management is very familiar with social learning processes and cooperative games of invention and communication because this is exactly what they engage in when formulating, executing, and validating their business strategies. Managers proceed through an emergent and iterative process, which can be thought of as shown in Figure 4.1.

An informal corporate model for strategy development often follows these four steps. The model is informal, because it is not explicitly exposed, manifest, or defined. The four steps often materialize with activities along these lines:

1. Mid-managers research the subject.
2. Mid-managers present their ideas to senior managers.
3. The proposals are discussed with senior managers; new ideas might emerge and get incorporated in the proposed strategy.
4. The process repeats.

Sample Chapter

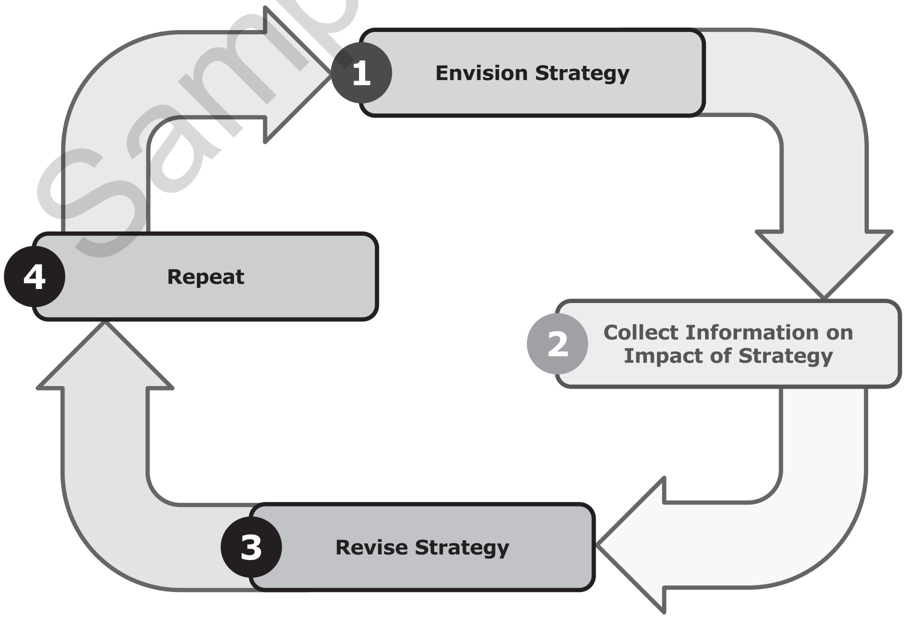


Figure 4.1 Strategy making

What is striking here, is that this management process is in essence very agile—though the resulting strategies might not always be as agile as the process itself. This strategy-making process can be compared to how the collaborative performance arts develop their plays:

1. Actors prepare for their roles.
2. Actors meet frequently to rehearse the emerging play.
3. The director and actors discuss each piece of the work as just experienced.
4. The process repeats.

While being developed through an apparently chaotic process, which involves frequent social interactions, all these collaborative performance arts are nonetheless extremely reliable, and deliver what is expected. The show that is not ready by the opening night will not be long-lived.

The connection with collaborative performance arts is significant. The processes employed are not master planned, but are iterative and emergent. Naturally the similarity connects to how agile software development processes typically unroll:

1. Developers discuss with customers concerning what the software should do, and then write some code.
2. Developers regularly build the system into an executable program that can be run, demoed, tested, and used.
3. Developers and customers discuss the episode of the last run, presenting any new ideas about change.
4. The process repeats.

The same kind of iterative and emergent process can be found in all knowledge work. The essence of all of these iterative and emergent processes is a loop that looks like the one shown in Figure 4.2.

It is in the third step, reconception and adaptation, that all plans and actions are *re-thought* by taking into consideration any new information, changes, or learning that has happened in reality. The process of rethinking implies exploring new directions, and trying again, basically repeating the process. It also implies learning, acceptance of criticism, and reflexivity.

In essence, this is a high-level feedback loop whereby the process adapts to reality, and a final working solution emerges, eventually. In the unrolling iterations of this loop, the social learning process that produces the capital goods (the knowledge-based product or service, in our case) takes place.

We see here that strategy making is very similar to collaborative performance arts, and to Agile software development. Therefore, it should be natural for company executives to embrace agility as an obvious mode of operations because they practice an agile process each and every time they develop, and then execute, a strategy.

Notwithstanding the preexisting profound understanding of the fundamental process, management is not able to put it into practice. It is unfortunate that despite appreciating the utility of feedback loops in strategy making, they get lost once the strategies are actually put into practice by line managers and staff. Often, management promotes the

Sample Chapter

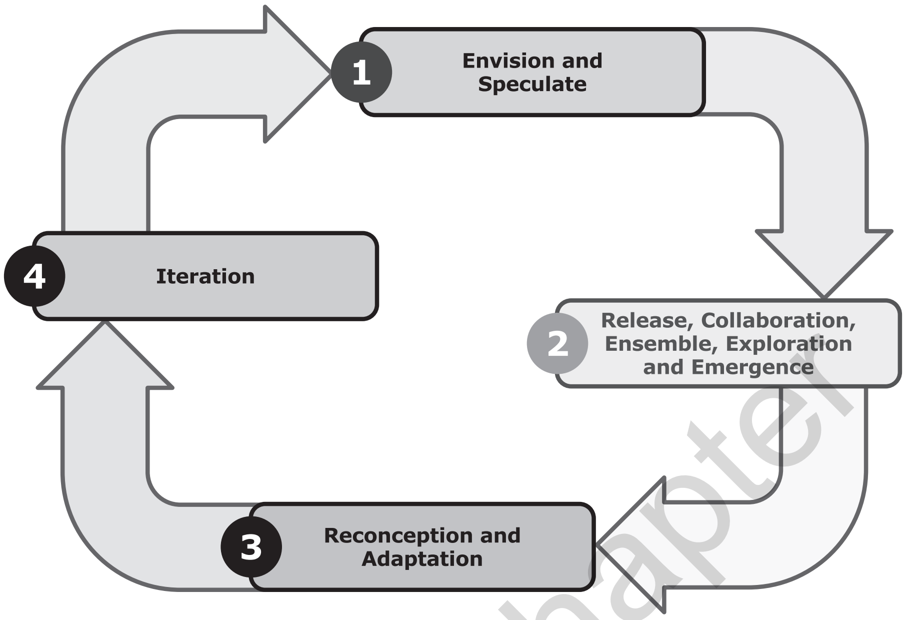

1
Envision and Speculate
2
Release, Collaboration, Ensemble, Exploration and Emergence
3
Reconception and Adaptation
4
Iteration

Figure 4.2 Iterative emergent process

Sample Chapter

operational use of more stringently defined processes which fundamentally hinder any learning to happen in the first place, even though learning is essential.

Senior executives who are successful at strategy making possess the essence of what is needed to successfully manage a knowledge-based organization. They know how to create strategies; they know that constant communication is imperative; they know that all of this is achieved through a social process of organizational learning. Now they need to acknowledge, internalize, and put into practice the concept that managing a knowledge-based organization is essentially the same as strategy making.

There are two powerful forces that prevent management from effectively leading a knowledge-based organization—their own (negative) attitude towards fostering a learning organization, and the conventional ways of exercising financial responsibility. In the next chapters we will see the implications of this and possible ways to overcome the impasse.

Web Added Value™

This book has free material available for download from the Web Added Value™ resource center at *www.jrosspub.com*