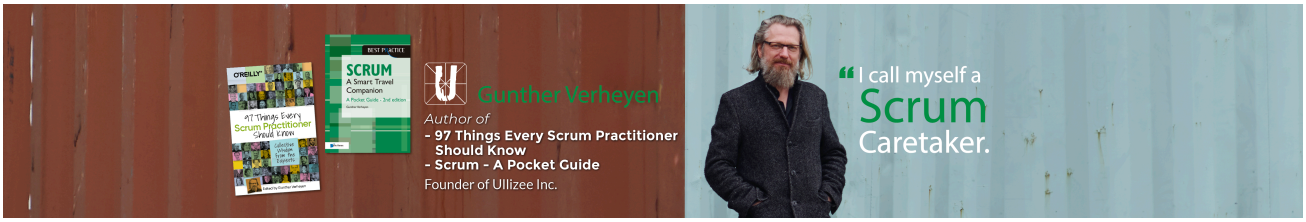
White Paper

Moving Your Scrum Downfield

The Six Essential Traits of the Game

by Gunther Verheyen
independent Scrum Caretaker



An ever-increasing variety of work in modern society is inherently complex. Scrum is being used in and beyond software and (new) product development for that reason. Apparently, it seems easy to get stuck at interpreting the rules of Scrum. This publication describes the six essential traits of the game that can help you understand better what it is that makes Scrum work, get unstuck and up your game. As they express rather intrinsic and implicit principles, they are too often disregarded. Yet, they are needed to grow to a more unconsidered performance of Scrum, which allows focusing on the goal of the game instead of the rules. The six essential traits are indicative of Scrum coming to life, no matter the nature of your problem.

Scrum Is Simple, Yet Sufficient.

Scrum supports people in addressing *complex* challenges and derive value from them, with *value* being a very different purpose than *volume* is.

Complex challenges are highly unpredictable and cannot be tackled with predefined or copy-paste solutions. Scrum is simple in the sense that it defines no more than a limited set of rules. That set is sufficient to devise a way of working specific and fitting to time and context, and continually optimize toward creating the most valuable outcomes. This does imply revising, adding, and improving work, management, people, and organizational practices.

In a nutshell, Scrum requires a Scrum Master to foster an environment where (repeatedly):

1. A Product Owner orders functions and solutions for value against an overarching product vision.
2. A Scrum Team creates valuable Increments of work against an overarching Sprint Goal.
3. All players figure out what to work on next and how to best organize for that.

Through its minimalistic design, every element in Scrum serves a purpose. Leaving out parts or not playing by the rules covers up problems, and limits –and eventually eliminates–substantial benefits.

Many struggle with this deliberate incompleteness of Scrum and demand exact instructions that universally apply–regardless of people involved, environment, tools, business, and markets. This desire for precision contradicts the complexity of reality and the reality of complexity.

Many tactics to apply the rules exist. Scrum is a skeleton process that can wrap new practices and render existing practices superfluous. Devise, apply, and tune patterns, practices, and techniques to better fit your Scrum, not to twist or overburden the framework.

The potential of Scrum unfolds by playing by the rules that apply, plus exploring how tactics, interactions, behaviors, and the six essential traits–of which *simplicity* is the first–make it work.

Scrum's DNA

Scrum is grounded in the management principles of *Self-organization* and *Empiricism*. They are entwined and form Scrum's DNA.

Self-organization asserts that the people undertaking complex work know best how to organize for that work. No external forces can do that better for them. Scrum sets the boundaries within which people are invited to use their intelligence and creativity to act with agility and collaboratively optimize for valuable outcomes.

Empiricism (or *empirical process control*) asserts that forward-looking decisions in complex work are best based on experience, observed results and outcomes of experimentation. Scrum implements empiricism via its methodical approach of inspection and adaptation upon transparency of the work being undertaken and results being produced.

Scrum combines self-organization and empiricism by engaging people in sharing or acquiring insights and skills to collectively perform complex work, while employing an iterative-incremental approach to make the best possible progress. It requires players to regularly stop, reflect, gather feedback, and learn from observation and inspection in order to continue or change course as needed, re-organize, improve, adapt.

Self-organization

Self-organization is the process of people forming organized groups around problems or challenges without external work plans or instructions being imposed on them. No single person can know better how to organize for complex work than a

Regarding the use of "Scrum Team"

In this publication "Scrum Team" is used where Scrum uses "Development Team."

Although 'development' should encapsulate all activities for creating and sustaining a product in an end-to-end way, it is often still understood as coding work only in the software domain.

Given the ever-expanding use of Scrum, there is no single term that resonates in all types of complex work for which Scrum is employed.

Rather than the neutral, but trivial "Team," "Scrum Team" is used here for the players moving as a unit up the field. Like "football team," "basketball team," or "rugby team."

A term for the combination of Scrum roles, for which Scrum uses "Scrum Team," is not used.

group of skilled people accountable for that work. Self-organization is employed when a group of people collectively organizes, manages, and performs its work.

Self-organization *is*, it happens. As any individual has the inherent quality to self-organize, self-organization needs not be instructed nor empowered. Rather, it requires removal of barriers that prevent people to apply their natural ability. Impediments to self-organization are not in people, but in processes, procedures, and organizational constructs. This is how external authorities are most effective—by removing organizational barriers to self-organization.

A shared (visual) workspace is an important enabler for self-organization. It forms a bounded environment that allows people to optimize focus, collaboration, and shared-ness, and benefit from the fastest information exchange possible. Self-organization is most effective in such workspace.

Empiricism

Empirical process control implies closed-loop feedback so that actual outcomes are regularly inspected and validated against desired outcomes. This is a self-correcting process as unwanted variances or results are eliminated or corrected through adaptation in the next or in future system runs.

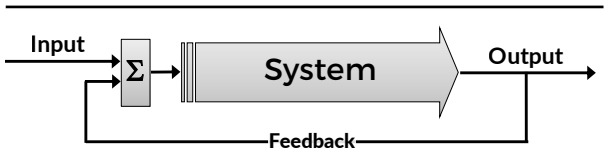


Exhibit 1: A closed-loop feedback system

Empiricism requires and creates transparency. Reality is exposed for inspection in order to allow for sensible adaptations. The player-inspectors take forward-looking decisions based on information that reflects their actual situation. Inspections of incomplete or twisted versions of reality lead to pointless adaptations and even detrimental deviations. Transparency serves the process of inspection and adaptation, and holds that all required information is available for the player-inspectors. It does not hold that any piece of information should be available for anyone.

From the need for transparency follows the need for standards and agreements to work and inspect against. They are transparent in the sense that they are agreed, followed, visible, accessible, and comprehensible. In Scrum, the definition of Done is particularly important in providing transparency over work to be done and work actually done.

The frequency at which inspections and adaptations are performed should be such that sufficient work can be performed to allow meaningful inspections while not impeding the opportunity to adapt to important new insights or evolutions.

Players Demonstrate Accountability

A sustainable complex adaptive system does not spring from individual heroism or hierarchical power. It requires people from different backgrounds and domains combining their skills, talents, experiences, insights, and personalities in working, learning, and improving together.

The increased emphasis on peer collaboration makes leadership more subtle, somewhat dispersed, and transient in time and place. Leadership becomes a distributed quality instead of an expression of individual assertiveness and dominance.

The complementarity of the accountabilities of *Product Owner*, *Scrum Team*, and *Scrum Master* instantiates such collaborative spirit. As they collaborate, they also develop new relationships with consumers, stakeholders, and others. Tensions are a natural part of this process.

Accountability is not in titles or functions. It cannot be demanded or instructed. Rather than installing measures and means to 'hold people accountable' foster an environment where self-organizing people can *demonstrate accountability*.

Product Owner

'Product Owner' is a one-person player role to connect consumers, stakeholders, and Scrum Teams. This is very different from dominating all communication or preventing direct interactions. The core accountability is to optimize value—for (but not limited to) the people receiving the work, the organization funding it, and the people performing it.

'Product' is *the* vehicle for a Product Owner to deliver value. A product is a tangible or non-tangible good or service, or is more abstract like the outcome of specific processes or actions. Without a clearly identified 'product'—including its consumers and stakeholders—a Product Owner can hardly be effective in optimizing for value delivered. As a result, Scrum is hardly used effectively.

A Product Owner, self-evidently minding the long-term viability of the product, leads through a product vision. A product vision captures *why* the product exists. It encapsulates what makes the product worthwhile buying, consuming, and investing in. A product vision helps consider specific goals, uncover product functions and solutions, and validate whether value is actually being created via product Increments.

A Product Owner orders envisioned functions and solutions in a Product Backlog and assures that they are known and understood for how they potentially deliver value. A Product Owner represents the needs of many and is the sole person deciding over ordering the Product Backlog and spending the game budget.

The Product Owner is accountable, whether doing the above or having others do it.

Scrum Team

'Scrum Team' is a multi-person player role consisting of a group of people self-organizing around the challenge of turning functions and solutions from the Product Backlog into observable, Done output. The core accountability is to create usable and valuable Increments of work no later than by the end of a Sprint, and sustaining the resultant product.

Scrum Teams work with the Product Owner in identifying the most important functions and solutions for a Sprint. Scrum Teams perform and manage all activities involved in delivering such forecast of Product Backlog in a Sprint. They work with the Product Owner as needed to optimize the Sprint's outcome captured in a Sprint Goal.

Beyond managing their own work, Scrum Teams also self-organize for that work in terms of size, skills, expertise, and availabilities, using the process of inspection and adaptation.

Size and work organization are such that Scrum Teams work at a sustainable pace, a pace that can be maintained indefinitely. Working in Sprints serves rhythm and cadence, and improves focus and consistency, but is not for burning out people.

To assure alignment and consistency of the collaborative work, a Scrum Team has an agreed set of work practices and standards in place to be applied to the collective work. In particular, the qualities and criteria that must be met for an Increment to be declared "Done" are captured in a *definition of Done*. This ensures a shared understanding of the state of an Increment when inspected. What is needed in Scrum Teams, in terms of skills, tools, and practices, is a function of what is defined as Done—not the other way round.

A Scrum Team is always accountable as a whole, regardless of the type of work needed or performed, or specialized skills and focus areas of individual players. No sub-teams, titles, or hierarchy exist within a Scrum Team.

Scrum Master

'Scrum Master' is a one-person player role acting like a game master in assuring that the rules of the game are known and understood and, additionally, supporting players in uncovering better ways to play. The core accountability is to guide self-organization toward the creation of valuable outcomes.

This requires certain management skills, traits, and insights, but it is very different from being a traditional manager. A Scrum Master has no formal power over people, careers, or incentives. A Scrum Master does not control progress, budget, quality, or tasks. A Scrum Master does support the right players figure out how to manage these in Scrum.

A Scrum Master leads through a vision of what can be achieved with Scrum in terms of engagement, creativity, and a humanized workplace. A Scrum Master induces the continual desire to become better players. Scrum Masters understand that embracing Scrum doesn't occur overnight, but is a journey. They are patiently impatient.

A Scrum Master facilitates players by making sure that *impediments* are removed, elements that hinder or block them in their work and are beyond their control. Think organizational expectations, directives, processes, procedures, or structures that are at odds with the rules, values, principles, or goal of Scrum. This may require coaching for behavioral change. Forming alliances with fellow Scrum Masters comes naturally when having to challenge some bigger status quo, organizational or otherwise.

Removing impediments, facilitating events, teaching techniques, supporting teams, educating the organization, keeping the road open to perform, to work, to innovate, to be creative are some of the services that a Scrum Master provides. How interventionist the services are is a mirror of the state of Scrum within an organization.

Scrum Master accountability cannot be eliminated as complex work, turbulent circumstances and changing environments inevitably give rise to situations and challenges for which players need support, guidance, and coaching. Scrum Masters can only strive to become invisibly present.

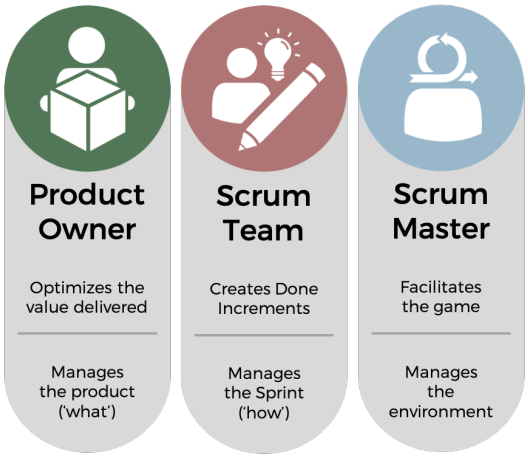


Exhibit 2: Accountability in Scrum

Transparency for a Flow of Value

Complex challenges are highly unpredictable. Deriving value from them requires, more than harnessing or 'managing' change, capitalizing on new insights, accumulated experience, and unforeseen opportunities. This is why Scrum implements empiricism.

Transparency, as the foundation of empiricism, includes that work done and work to be done can be fully understood at any point in time—regardless past hopes, dreams, and plans. The extent to which the Scrum artifacts of *Product Backlog*, *Sprint Backlog*, and *Increment* reflect reality impacts future outcomes, down to the risk of becoming useless and even harmful when

including unaccountable deviations. The definition of Done is particularly important in making the reality of observed work fully transparent.

The Scrum artifacts support maintaining a *flow of value* at a macro level by uncovering, ordering, and delivering functions and solutions. A clearly identified 'product,' as the vehicle to deliver value, provides focus and purpose to the use of the Scrum artifacts. Without such clear identification, optimizing for value is hardly possible and Scrum is hardly used effectively.

Product Backlog

Product Backlog is an emergent, ordered list of functions and solutions that the Product Owner deems as potentially valuable, while exposing other factors that enhance or obstruct the flow of value—like dependencies, compliances, and constraints.

Product Backlog is the primary source of work and progress in Scrum. While the Product Owner is accountable for its ordering, Scrum Teams are responsible for its sizing. Product Backlog is the single source of work for Scrum Teams. A Product Owner keeps everyone with a vetted interest updated on Product Backlog progress.

A Product Backlog reminds all players that right-time conversations are required to elaborate on what the work entangles. This is very different from exhaustive lists holding exhaustively detailed requirements. The primary value of Product Backlog is in providing focus on important opportunities to create value—not in completeness, precision, or detail. A Product Backlog is not a tool for trying to predict the inherently unpredictable.

The Product Owner is accountable that the Product Backlog exists and is ordered. To maximize transparency and assure fast and consistent decisions, one product has one Product Backlog ordered by one Product Owner, regardless the number of Scrum Teams involved.

Sprint Backlog

Sprint Backlog is the emergent plan of a Scrum Team for a Sprint. A Scrum Team uses Sprint Backlog to manage the improvements and the work anticipated to most effectively turn selected functions and solutions from the Product Backlog into Done Increments, therein guided by the Sprint Goal. Only the Scrum Team decides what is in their Sprint Backlog, and how to manage that.

Sprint Backlog is a living artifact that is kept accurate and realistic. Throughout a Sprint new insights for achieving the Sprint Goal surface. Work that becomes obsolete or was unanticipated is removed from or added to the Sprint Backlog. Sprint Backlog is never updated later than at the Daily Scrum.

Increments of integrated, Done output emerge from the collaborative work of Scrum Teams. Sprint Backlog is used to keep track of progress of a Sprint in order to avoid missing out on adaptations. If the actual progress impacts the forecast of Product Backlog, the Product Owner is consulted. If no movement is radiated, the empirical process might be in danger and the Scrum Team in need of help.

Increment

Quality and progress are assured by repeatedly producing Increments upon defined, agreed work practices and standards. An Increment is a solid, meaningful body of work that is available for inspection no later than by the end of a Sprint. An Increment has no hidden or "undone" work, work that does not comply with the definition of Done. Product is the integrated resultant of all available Increments. Adaptations to the definition of Done may reveal work that must be done for previous Increments first.

Increments incorporate additions, expansions, improvements, eliminations, and modifications. Regardless the time of their availability, Increments can be released when they are Done *and* are deemed useful by the Product Owner. An Increment assures that one or more—in the latter case a cohesive combination of—Done functions and solutions become available for the product's consumers.

It is good—if not essential—practice for a Product Owner to know of usage, satisfaction, or other indicators of impact and value of Increments to validate this against the product vision or other ambitions. Otherwise decisions to optimize value are still doused with opaqueness.

The definition of Done is particularly important in —as a minimum—assuring that an Increment is usable, stable and of high quality. Quality best encompasses more, however. A Done product should exhibit the qualities an organization envisions and wants to be known for with its consumers. A Done product should exhibit the qualities needed to deliver or result in value. Ideally, the definition of Done would echo *valuable* and exceed releasable.

The state of an Increment described in the definition of Done is not a function of the skills available in Scrum Teams, or the tools and practices applied, but the other way round.

Closing the Loops

Effective use of Scrum entails closing loops, likely more loops than meet the eye.

In Scrum all work is encapsulated in Sprints. Sprint length allows weighing progress in terms of tangible output against the ability to adapt upon feedback gathered at the macro level. Sprint length is a factor of minimal stability that reflects the contextual need for the frequency at which to inspect and adapt at a tactic and strategic level. Sprints are never more than four weeks.

For the external world, Sprint is the only unit of work and time—not days, hours, or work estimates. A Sprint is a shielded playfield for creating Done

Increments of work. What happens in a Sprint, stays in the Sprint. This is how external authorities are most effective—by preventing distractions or interventions during a Sprint, and not being one.

Complex work innately incorporates many divergent options. Players are required to converge regularly *within a Sprint*, not just toward its end. Full closure by the end of a Sprint is needed to preserve unburdened adaptability at the macro level. Open work is not a valuable outcome, blocks creativity and openness, and is a liability for adaptability and future value creation.

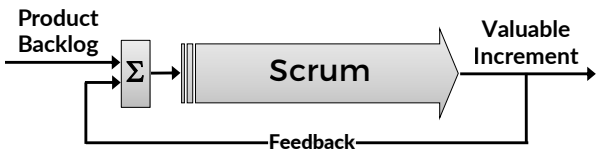


Exhibit 3: Closed-loop feedback with Scrum

A dramatic way of preventing proper closure is the termination of an on-going Sprint. It occurs only when the work of a Sprint is fundamentally and totally invalidated and cannot be replaced.

Planning for a Sprint

Planning for a Sprint is the opening act of a Sprint. The players consider and collaborate on *what* the most valuable functions and solutions are, *why* they are worthwhile and *how* to turn them into cohesive and observable output.

Scrum defines the *Sprint Planning* event for this purpose. The event takes as long as needed to meet its goal, but never more than eight hours. The goal is to choose direction and allow embarkment—rather than predicting exact output.

The players choose how to organize Sprint Planning. Experience shows how some select multiple functions and solutions first, identify the work needed, and update the selection as they find they have more or less availability. Others iterate dynamically between the *what* and the *how* of individual functions or solutions. Others do something in between. The players decide but assure alignment with a Sprint Goal.

The Product Owner shares and clarifies Product Backlog so the Scrum Team can anticipate and map out work to be performed. Only the Scrum Team determines how much it reasonably can achieve within the Sprint. A forecast of work for a Sprint is tuned to the Sprint length—not the other way round.

The events meets its minimal goal if Sprint Backlog holds enough work to embark and the Scrum Team can start turning selected functions and solutions into observable output. Additional work and new insights can be assessed and managed via Sprint Backlog in due time. The Sprint Goal expresses what makes the Sprint worth the energy and the investment—an envisioned state of the product or some other meaningful outcome.

Managing for Progress

Protected from external distractions, the players continue engaging in collaborative work to move as a unit up the field toward achieving the next game level, the Sprint Goal.

The Scrum Team applies its agreed work practices to incrementally create Done output, satisfy the Sprint Goal, and generate valuable outcomes. Feedback loops within the Sprint are closed regularly and repeatedly to assure alignment and consistency as well as to catch problems and errors early. Consider the propagation of errors in complex work that, if not detected early, potentially endanger proper closure of the Sprint.

Sprint Backlog is kept actual in terms of reflecting what is needed to achieve the Sprint Goal. No later than on a daily basis progress toward the Sprint Goal is updated and discussed to identify and agree over the upcoming Sprint work, in particular for the next day. Scrum defines the *Daily Scrum* event for this purpose, time-boxed to 15 minutes.

If during the Sprint the Scrum Team discovers that substantially different, more or less is needed or possible than planned for, the forecast can be renegotiated with the Product Owner. If an Increment complies with the definition of Done, it can be shared with the product's consumers upon the Product Owner's consent.

Sprint Reflections

Sprint reflections serve closing a Sprint. The players and invited participants collaborate on *why* the Sprint was undertaken, *what* functions and solutions are delivered in Increment(s) against this Sprint Goal, and *how* the Sprint went. They reflect on and identify adaptations for the next game round. Inspection without adaptation is pointless.

The definition of Done is particularly important in assuring that all have a shared understanding of the qualities and state of the work being observed. It is imperative that an Increment has all characteristics of the product. In general, a paper report or a presentation do not meet that demand.

Scrum defines two events for these closing activities, *Sprint Review* and *Sprint Retrospective*.

Sprint Review focuses on *why* a Sprint was undertaken and *what* was achieved. The Product Owner connects the Scrum Team with invited stakeholders and consumers for this purpose. Product Backlog is used to assess progress along with major changes that impacted it. All attendees collaborate and share ideas how to further improve the value of the product in the next Sprint(s). This is captured in an updated Product Backlog. The event takes as long as needed to meet this goal, but never more than four hours.

Sprint Retrospective serves a deep dive on *how* the Sprint went. Many aspects of the work are covered, including (but not limited to): team engagement, Done-ness of the Increment(s), the use of Scrum, practices and techniques, social

aspects, collaboration, team values, and team agreements. Although improvements may be implemented at any time, a Sprint Retrospective provides a formal opportunity to identify and plan for improvements of the actual work process. The event takes as long as needed to meet this goal, but never more than three hours.

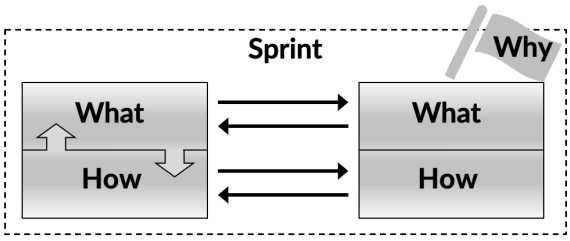


Exhibit 4: Information circulation at the macro level

The Scrum Values

Scrum is more effective through spirited collaboration, for which it provides a frame. Scrum, actually, is more about behavior than it is about process. Values drive behavior. The Scrum Values of *Commitment*, *Focus*, *Openness*, *Respect*, and *Courage* give direction to working in Scrum. All decisions, the steps taken, the tactics chosen to apply the rules of Scrum should re-enforce these values, not diminish or undermine them.

- *Commitment* shows in the working spirit of the players, in terms of motivation, dedication, and engagement. It is about the actions and the intensity of efforts, not about exact and predicted volume in terms of output.
- *Focus* is increased through the balanced but distinct accountabilities of Scrum. Time-boxing all work encourages players to focus on what is imminent now as the future in complex challenges is highly uncertain.
- *Openness* is in the attitude of all players. It is reflected in their collaboration, interactions and relationships, in how they deal with differences in skills, personalities, and opinions.
- *Respect* is essential for self-organizing groups to navigate complexity. It requires respecting, while at times respectfully challenging, rules, agreements, skills, practices, ideas, and viewpoints.
- *Courage* is much needed in the face of uncertainty, in upholding quality, in embracing imperfection, in applying empiricism to turn change into a source of inspiration and innovation, and in living… the Scrum Values.

How the six essential traits of the game are indicative of Scrum coming to life:

1. Scrum Is Simple, Yet Sufficient. The players unfold the potential of Scrum by using the *simple* rules that apply and explore how tactics, interactions, behaviors, and the six essential traits make Scrum work.

2. Scrum's DNA. The players form a *self-organizing* unit around the challenge of collectively creating observable, Done Increments of work, while employing *empiricism* to manage all work and progress.

3. Players Demonstrate Accountability. The players contribute to valuable system outcomes through spirited collaboration, and sharing and challenging rules, agreements, skills, practices, ideas, and viewpoints.

4. Transparency for a Flow of Value. The players use the Scrum artifacts to uphold transparency over all work done and work to be done, manage for a flow of value and preserve the ability to capitalize on unforeseen opportunities.

5. Closing the Loops. The players regularly and repeatedly close the many intertwined loops within a Sprint toward full closure by the end of a Sprint and preserving unburdened adaptability at the macro level.

6. The Scrum Values. The Scrum Values of *Commitment*, *Focus*, *Openness*, *Respect*, and *Courage* take prominence in the behaviors, relationships, actions, and decisions of the players and their ecosystem.

About the author

Gunther Verheyen ventured into IT and software development after graduating in 1992. His Agile journey started with eXtreme Programming and Scrum in 2003. Years of dedication and employing Scrum in diverse circumstances followed. In 2010, Gunther became the inspiring force behind some large-scale enterprise transformations. In 2011, he became a Professional Scrum Trainer.

Gunther founded Ullizee-Inc in 2013 to partner exclusively with Ken Schwaber. He represented Ken and his organization Scrum.org in Europe, while shepherding its Professional Scrum series and guiding its global network of trainer/coaches. Gunther is co-creator of Agility Path, EBM (Evidence-Based Management), and the Nexus framework for Scaled Professional Scrum.

Since 2016, Gunther continues his journey of humanizing the workplace as an independent Scrum Caretaker—a connector, writer, trainer, and speaker. His services build on more than 15 years' worth of experience, ideas, beliefs, and observations of Scrum.

Gunther authored the acclaimed books *Scrum - A Pocket Guide* (Van Haren Publishing, 2013 & 2019) and *97 Things Every Scrum Practitioner Should Know* (O'Reilly Media, 2020). Ken Schwaber recommends his pocket guide to Scrum as "…the best description of Scrum currently available" and "extraordinarily competent." Several translations of Gunther's work are available.

When not traveling for Scrum and humanizing the workplace, Gunther lives and works in Antwerp, Belgium.