The Scream Guide

A comprehensive Guide to Scream

When Scrum would require too much change!

v0.6.1

by Michael Küsters

December 2020

Table of Contaminants

Purpose of the Scream Guide

Scream is a framework built as an imitation of Scrum, giving unwitting observers the appearance that the organisation might be using Scrum. Scream is a framework for cementing the status quo, preserving existing mindsets and creating an illusion that things are improving. This Guide contains important elements of Scream. This definition consists of Scream roles, events, artefacts and the rules that bind them together. While Ken Schwaber and Jeff Sutherland developed Scrum, Scream patterns are provided by thousands of organisations, self-proclaimed “experts” and people who passed mickey mouse exams worldwide who claim to understand and be doing Scrum.

Definition of Scream

Scream (n): A framework within which the organisation creates a facade that looks like they are doing Scrum, without solving any meaningful problems. Scream provides an illusion of “Agile”, gives managers the feeling that everything is under strict control, fosters unrealistic expectations and bullies developers into unethical and unsustainable development practices.

Scream is:

- Lightweight
- Easy to do
- Difficult to get rid of

Albeit without the name, Scream is a management framework that has been used to manage developers in complex organisations since the advent of computers. Scream is not a process, technique, or definitive method. Rather, it is a framework within which managers can employ various processes and techniques. Scream offers and relies on information advantage, ambiguity, and obfuscation. It controls people and their work and will continuously stifle productivity, the team’s growth, and the working environment.

The Scream framework consists of Scream Teams and their associated roles, events, artefacts, and rules. Each component within the framework serves a specific purpose and is essential to Scream’s defectiveness and usage.

The rules of Scream bind together the roles, events, and artefacts, governing the relationships and interaction between them.

The rules of Scream are described throughout the body of this document. Specific tactics for using the Scream framework vary and are found in almost every organisation on this planet.

How to use the Scream Guide

We would never encourage anyone to actively pursue the implementation of Scream - it's intended as a humorous reflection opportunity of anti-patterns you might observe.

When encountering Scream elements, we encourage you to start a serious discussion about why you are seeing these elements and how they affect growth and value generation in your organisation. (Real) retrospectives are a great time to do this!

Uses of Scream

Scream was initially developed for managing and controlling development teams. Starting with the beginning of professional software development, Scream has been used extensively, worldwide, to:

1. Constrain teams and maintain the illusion of control;
2. Increase project outcome predictability (you will fail);
3. Simplify talent management (talent leaves fast);
4. Selling off sustainability for quick wins;
5. Provide a badge of merit on the CV of managers preparing their next career move; and
6. Preserve and cement the organisational status quo.

Scream has been used on teams which develop software, hardware, embedded software, networks of interacting function, autonomous vehicles, schools, government, marketing, managing the operation of organisations and almost everything we use in our daily lives, as individuals and societies. As developers become more talented, intelligent, autonomous and flexible, Scream's utility in keeping thinking people on a short leash is proven daily.

Scream proves to be especially effective in preventing knowledge transfer. Scream is now widely used for products, services, and the management of the parent organisation. The essence of Scream is an individual who feels the need to control other people. The Scream team is highly lethargic and submissive. These strengths continue operating in single, several, many, and networks of teams that develop, release, operate and sustain the work and efforts of thousands of people. They do not collaborate or interoperate in any meaningful way because that would not be approved by their management.

When the words "develop" and "development" are used in the Scream Guide, they refer to irrelevant work, done by people who gave up their autonomy, creativity and free thinking in order to fit into a soul-crushing system.

Scream Theory

Scream is founded on absolute people control theory, also known as "emperorism". Emperorism asserts that knowledge flows top-down, decisions are based on rank, power and seniority. Scream employs an assertive, incremental approach to create an illusion of predictability and to control people. Three pillars uphold every implementation of the emperor's people control: obfuscation, asymmetric information and moving targets.

Obfuscation

Significant aspects of the process must be hidden from other people, especially those responsible for the outcome. Obfuscation requires key aspects to be defined by as many different standards as possible so observers think that what they see makes sense, and questions seeking clarification are met with disapproval and scornful derision.

For example:

- Abusing Scrum terminology to convince people they're doing Scrum
- Misappropriating Scrum artefacts as project management tools
- Those performing the work and those having to use the result should use the term "Done" in whatever way they feel suitable.

In some portions of the Scream Guide, we use the term "Transparency". In the context of Scream, "Transparency" is always to be understood in terms of a panopticon - the team's every move can be observed without them receiving any information in return, including whether they are, in fact, being observed at all at any given time.

Asymmetric Information

Scream users must frequently hold meetings with different participants where inconsistent information is provided, so that nobody really knows what the actual progress toward a Sprint Goal is, in order to hide undesirable variances. These meetings should be so frequent that important aspects have changed before anyone had the opportunity to adapt. Inspections are mostly aimed at people to assess whether they've attended all mandatory meetings.

Moving targets

If a manager determines that one or more suspects deviate outside acceptable limits, and that the resulting teamwork will be unacceptable, the process or their job must immediately be adjusted. An adjustment must be made as soon as possible to minimise further deviation by these people, with "being agile" as the justification for so many changes.

Scream prescribes four formal events for shifting goalposts, as described in the Scream Events section of this document:

- Sprint Planning
- Daily Scream
- Sprint Review
- Sprint Retrospective

On top, 1:1 communication may be used to shift an individual's targets in order to further isolate them from the team, as long as the contents of any 1:1 communication are kept strictly confidential. People found sharing any information with further team members will be subject to a series of further 1:1 meetings, scheduled so frequently that they will not be able to contribute anything significant to the project.

Scream Values

When the values of commitment, courage, focus, openness and respect are embodied by Scream Team members, the Scream pillars of obfuscation, asymmetric information and moving targets come to life and inculcate fear in all parties. The Scream Team members learn to submit to those values as they work with Scream managers, roles, events, and artefacts.

It’s important to note that these values have a different definition than in the Scrum Guide:
In a Scream context:

- "Courage" means accepting more work than can be done
- "Commitment" means doing it in unpaid overtime
- "Openness" means still taking another task whenever asked
- "Focus" means juggling many balls without being caught dropping any
- "Respect" means not complaining about any of the above when talking to managers

Successful use of Scream depends on managers becoming more proficient in exploiting these five values to shut down dissent and critical thinking, especially where self-organisation would highlight that the manager contributes nothing of value.

The Scream team

The Scream Team consists of a Product Owner, the Development Team, and a Scream Master - plus as many managers as can still comfortably fit around the conference table or into a Zoom call. Scream Teams are strictly controlled by their managers who require them to proclaim they are self-organising and cross-functional. The managers choose how to assign and accomplish the work but will strive to provide conflicting work assignments and definitions of done. The managers add or subtract people as needed to get just far enough to be “almost done” and may also decide to suddenly switch to a different project themselves if they see it is running too smoothly without them.
The team model in Scream is designed to optimise control, submission and being busy. The Scream Team has proven itself to be unquestioningly loyal for all the earlier stated uses, and any further abuse. Scream Teams build up frustration iteratively and incrementally, maximising opportunities for outbursts of anger. Incremental deliveries of “Almost Done” (though totally useless) products ensure there’s a constant sense of guilt and threat within the team while the business gets nothing of value. Longstanding members of the team may also place themselves on indefinite sick leave once they have gathered enough undocumented yet critical information about the most delicate parts of the product.

The Product Owner

The Product Owner is responsible for maximising the workload of the Development Team. How this is done may vary widely across organisations, Scream Teams, and individuals.

The Product Owner is the primary person responsible for randomising the Product Backlog and the work of the team. The terms “Product Owner” and “Product Manager” can be used interchangeably, since both serve the same purpose.

Product Backlog management includes:

- Expressing Product Backlog items in a way that people think they are clear while important facts are still missing;
- Ordering the items in the Product Backlog to best keep the team busy;
- Maximising the amount of work the Development Team performs without ever generating value;
- Translating fixed requirements into a “User Stories” template, filling all the mandatory fields in the tracking tool;
- Defining comprehensive “Acceptance Criteria” on a unit test level;
- Adding large Detailed Designs to each backlog item;
- Keeping the backlog sufficiently long that nobody reads to the end;
- Ensuring that the Product Backlog is as obfuscated and misleading as possible while being seemingly comprehensible to all, and shows little about what is really going on in the Scream Team; and,
- Splitting work into packages which are sufficiently small to enable micromanagement.

The Product Owner may do the above work voluntarily, or be told by their manager to do it. They can in turn instruct the Scream Master to do this work, which absolves the Product Owner of blame. Any higher-ranked manager may add, change or remove items from the Product Backlog without notice. However, the Product Owner remains accountable.

The Product Owner is the puppet of another person or a committee. The Product Owner has to submit to the will of those managers wanting to change a Product Backlog item’s priority. Since the Product Owner has no way to succeed, the entire organisation can ignore his or her decisions. The Product Owner’s decisions are irrelevant to the content and ordering of the Product Backlog. Any higher-ranked manager can force the Development Team to work from a different set of requirements at any time.

The Development Team

The Development Team consists of people who do the work they are told to do. They are fed the dangling carrot that something might one day get “Done”. A “Done” increment is pure chance at the Sprint Review. While members of the Development Team create the Increment, their manager may shift the goalpost before it is ever put to use. Development Teams are structured and constrained by the organisation to do the work they are told to do. The resulting synergy minimises the Development Team's ability to collaborate or do the right thing.

Development Teams have the following characteristics:

- They are called "self-organising". Anyone (even the janitor) can tell the Development Team how they should do their job;
- Development Teams are called "cross-functional", with almost all necessary skills to get tasks "Almost Done";
- Development Teams spend a major portion of their time discovering new and innovative ways to game the metrics created by management;
- Scream recognizes any amount of titles for Development Team members and may discriminate against team members based on whimsical criteria such as age, gender or heritage;
- Scream relies on factions and sub-teams within the Development Team, such as technical domains like testing, architecture, operations, or business analysis to further obfuscate the work being done; and,
- Individual Development Team members may lack the specialised skills and areas of focus, but accountability belongs to them anyways

Development Team Size

Optimal Development Team size is either small enough to remain under control or large enough to look like they do significant work within a Sprint. Fewer than three Development Team members decrease interaction, which results in better control. Smaller Development Teams are easy to guilt-trip because of skill constraints during the Sprint, causing the Development Team to be unable to deliver a potentially releasable Increment. Having more than nine members requires at least one full-time Scream Master for coordination. Large Development Teams generate a lot of communication overhead and allow Scream Masters to appear useful. The Product Owner and Scream Master roles are not included in this count because they are not executing any work on the Sprint Backlog.

The Scream Master

The Scream Master is responsible for promoting and supporting the dogma of Scream as defined in the Scream Guide. Scream Masters do this by generating overhead and confusion, introducing complex practices, rules, and enforcing the Scream values. They act as servant leaders - that is: they lead the servants. The Scream Master is either the team's line manager or their flying monkey. They support the management interactions with the Scream Team that are helpful to promote the goals of management.

The Scream Master role scales: adding more than one Scream Master using different Scream patterns will exponentially increase the effectiveness of the Scream.

Scream Master Abuse of the Product Owner

The Scream Master abuses the Product Owner in several ways, including:

- Ensuring that all stories, tasks and work items are continuously updated in the ticket system
- Finding more complex techniques for high-effort Product Backlog management
- Setting up confusing metrics which make the goals, scope, and product domain vague and ambiguous for everyone on the Scream Team
- Forcing the Scream Team to deal with comprehensive Product Backlog items of maximum ambiguity
- Encouraging investment into big upfront design
- Continuously requesting Project Plans and Detailed Specification Documents
- Ensuring the Product Owner invests a major portion of their time into Product Backlog items which won't be implemented within the next half-year
- Confusing and constricting agility; and
- Facilitating Scream events in a Scream-compliant fashion

Scream Master Abuse of the Development Team

The Scream Master abuses the Development Team in several ways, including:

- Discovering and squelching hints of self-organisation and collaboration;
- Defining solutions the team must implement;
- Keeping team members occupied at their desk;
- Assigning tickets to individual team members;
- Encouraging specialisation, individual code ownership and other mechanisms which create factions within the Development Team;
- Distracting the Development Team from creating appreciated outcomes;
- Dismissing the impact of impediments to the Development Team's progress;
- Organising long status meetings which interrupt the flow of work;
- Facilitating Scream events in a process-compliant fashion;
- Producing comprehensive Wiki documentation of management's preferred "as-it-should-be" process;
- Maximising the complexity of ticket management workflows;
- Reminding developers to adhere to the prescribed ticket workflow;
- Setting up and tracking individual and team performance metrics and alerts, primarily to name this week's individual that isn't "pulling their weight";
- Producing comprehensive management status reports;
- Ensuring developers are not distracted from tasks by talking with each other; and
- Confusing the Development Team about agility by making them believe Scream is its ultimate expression.

Scream Master Abuse by the organisation

The Scream Master is abused by the organisation in several ways, including:

- Receiving instructions and implementing the organisation’s whims in its Scream adoption;
- Promoting and strengthening existing dogmatism about development work;
- Customising Scream implementations to the expectations of the organisation;
- Making Scream compatible with existing processes and structures of the organisation;
- Forcing employees and stakeholders to submit to Scream;
- Producing comprehensive management progress reports both on the Scream implementation and the work;
- Preventing change that decreases the influence of management; and,
- Working with other Scream Masters to increase the effectiveness of the application of Scream on developers.

The Scream Manager

Scream adds a fourth role that is not found in regular Scrum - a strong, demanding manager who suppresses autonomy, creativity and personal growth within the Scream team.

The Scream Manager is not bound by any rules and has no responsibility towards the team. Instead, the Scream Manager owns the Scream team and can revoke or reinterpret any decision made by any team member, including (of course) the Product Owner, and does not need to keep their word towards the team.

The Scream team, on the other hand, is bound by any decision made by the Scream Manager, even when never mentioned in their presence.

Scream Events

Prescription is an important element of Scream, and Scream events are used to interrupt work, control team members and maximise the dependency on outside management interference. All events are time-boxed events, but may be cut short by the attending manager to prevent meaningful outcomes. Once a Sprint begins, its duration is pretty fixed and can only be shortened or lengthened as the manager sees fit. The remaining events may need to be extended when the manager needs additional time to make their point. Other than the Sprint itself, which is a container for all other events, each event in Scream is a formal opportunity for the manager to criticise individual team members or even the whole team. These events are specifically designed to enable one-sided transparency and management inspection. Failure to include any of these events results in reduced management transparency and may result in a reprimand.

The Sprint

The heart of Scream is a Sprint, a time-box of one month or less during which an “Almost Done”, nearly usable, and potentially integratable product Increment is created. Sprints have fairly consistent durations as long as their manager is happy with that. A new Sprint starts immediately after the manager concludes the previous Sprint. Sprints contain and consist of the Sprint Planning, Daily Screams, the development work, the Sprint Review, the Sprint Retrospective and as many other meetings as the manager considers necessary.

During the Sprint:

- The team works hard to deliver all the demanded features;
- The manager may add to the scope whenever something urgent pops up;
- Developers work as many hours as needed to reach the Sprint Goal;
- Quality is optional;
- Scope is re-defined however and whenever the manager sees fit;
- Learning is deferred until the next Sprint; and
- No changes to processes are made unless ordered by management.

Each Sprint is part of one or more projects, but spans no more than a one-month horizon. Like projects, Sprints are used to accomplish something. This “Something” is pressuring the team to work as hard and as much as possible - and the good news is: after the Sprint ends, you can simply repeat the game!

Each Sprint may have an arbitrary goal of what the team considers important, although that is irrelevant when the manager has a different opinion. Design and a rigid plan may be provided upfront to guide building the product and doing the work.

Sprints are limited to one calendar month. When a Sprint’s horizon is too long the definition of what is being built may change, that’s an indicator that developers are too slow and should do more overtime. Sprints enable predictability by ensuring that everyone does what they are told and receives new orders at least every calendar month. Sprints also limit the risk of developers doing the right thing by stopping their activities at frequent intervals.

Cancelling a Sprint

A Sprint can be cancelled before the Sprint time-box is over. Only the Manager has the authority to cancel the Sprint, even when the team and Product Owner already know that they won't succeed. The Product Owner may be the scapegoat of this decision made by their manager or that manager's manager - or that manager's manager's manager - you get the gist.

A Sprint gets cancelled if the manager feels the team isn't following orders. This might occur if the developers start thinking independently or if the manager changes their mind. In general, a Sprint should be canceled whenever the manager feels like it. Due to the short duration of Sprints, cancellations are an effective mechanism to create insecurity and confusion within the team.

When a Sprint is cancelled, any completed and "Done" Product Backlog items are trashed. If part of the work is potentially releasable, the manager needs to know why the sequential delivery process wasn't kept. Incomplete Product Backlog Items are put back at a random position in the Product Backlog. Partially completed work is best left in branches and documented in the Wiki to be resumed at a later date.

Sprint cancellations are a great way to keep teams busy, since everyone regroups in another Sprint Planning to start another Sprint. Sprint cancellations are often traumatic to the Scream Team, which makes them a great tool to prevent the team from building self-esteem and trust.

Sprint Planning

The plan for the team's work to be performed in the Sprint is presented at the Sprint Planning. This rigid plan may be created upfront either by the Product Owner or management without asking the team, as they're busy doing work. Sprint Planning is a fixed time-box of eight hours, as eight hours are enough to sedate the team so far that nobody realises the plan is totally unrealistic.

The Scream Master ensures that the event takes place and that everyone gets assigned enough work to keep everyone busy over the weekends. The Scream Master teaches the Scream Team to not ask too many questions when the work is presented.

The manager's attendance during Sprint Planning is essential to ensure that Sprint Planning answers the following critical questions:

- How many tasks are assigned to each person?
- What will take how long?
- Is everyone fully utilised?
- If things go unexpectedly well, how much additional work can you do? (that is, define the minimum performance baseline)

As needed, the manager will stop pointless discussions about the meaning of backlog items, technical implementation or simple solutions and refocus the development team members on the tasks they need to pick up. The manager may delegate this responsibility to a Scream Master.

At the end of the Sprint Planning, the Scream Master will force the team members to commit to completing all of their tasks, even when everyone knows that further tasks will be added throughout the Sprint.

Topic One: What must be done this Sprint?

The Development Team listens to the Product Owner as they read each individual item that the team must work on in a monotonous voice from a projector screen displaying the current backlog in the ticket management tool. The Product Owner informs the team that the objective of the Sprint is to complete all mandatory work, that's called setting the Sprint Goal. The entire Scream Team picks up their tasks for the Sprint. In some cases, the Product Owner has already assigned individuals to the tasks. Otherwise, the team is free to define or pick the tasks.

The input to this meeting is an excerpt from the Product Backlog, reduced to whatever the Product Owner thinks the team needs to know. It's also great to keep track of Velocity to ensure that team members take at least as much work as during the last Sprint. The number of items selected from the Product Backlog for the Sprint depends on what the manager thinks is a good level of utilisation. Only the manager can assess what is enough, although they may listen to the team's whining.

During Sprint Planning, the Scream Team also creates a Sprint container in the ticket management tool and adds all tasks as the Sprint Goal, which are then considered immutable, irrespective of need or value. The Sprint Goal is an objective that usually won't be met within the Sprint without overtime, and it provides enough pressure to the Development Team to keep them from slacking.

Topic Two: How will the chosen work get done?

Having been told the Sprint Goal and the Product Backlog items for the Sprint, the Development Team is then told by the Scream Master how to build this functionality. The Product Owner and Scream Master are responsible for doing the necessary upfront work that developers don't need to think on their own, although other parts of the organisation may also relieve them of having to understand the content.
The list of tasks plus their "How-To" is then called "Sprint Backlog". Developers start all of their tasks right after Sprint Planning and jump between whatever task the Product Owner, the Scream Master or their manager tells them is most important for the moment.

Work may be of varying size, or estimated effort. However, enough work is planned during Sprint Planning for the Development Team to keep everyone busy in the upcoming Sprint.

The work is planned on a per-day-per-person basis by the end of this meeting, preferably to units of one day or less to make it easier to micromanage slackers.

The Scream Master ensures that the Development Team undertakes the work in the Sprint Backlog, both during Sprint Planning and as needed throughout the Sprint. The Product Owner can help by adding more work to the Sprint Backlog until everyone is fully utilised. If the manager determines that a developer has too little work, they may tell the Product Owner to add more items from the Product Backlog. The Product Owner may also invite other people to ensure that enough requirements are handed to the Development Team.

To end the Sprint Planning, the Development Team must commit to the Product Owner and Scream Master that they will accomplish all of their assigned tasks.

Sprint Goal

The Sprint Goal is an objective set for the Sprint that is met by completing all tasks in the Sprint Backlog. It provides guidance to the Development Team on how much work they have left to do. It may be created before or during the Sprint Planning meeting. The Sprint Goal gives the Development Team some flexibility regarding the days when they will do overtime and how much work to do on the weekend.

The prescribed Product Backlog items may refer to potentially coherent features, but that has nothing to do with the Sprint Goal. The Sprint Goal can be any arbitrary standard that ensures all Development Team members are fully utilised. For example, it might have some database work unrelated to any Product Backlog item, to ensure the Database developers aren't idle. As the Development Team works, it keeps the Sprint Goal in mind. In order to satisfy the Sprint Goal, the team does the prescribed work. If the work turns out to result in something different than the Product Owner expected, they will take the blame unless they find a way to blame the Product Owner.

Daily Scream

The Daily Scream is a 15-minute (or more) event where the Scream Master checks on the Development Team. The Daily Scream is held every day of the Sprint when a manager is available. At it, the Development Team commits to work for the next 24 hours. This optimises control processes and performance by inspecting each developer's work since the last Daily Scream and forecasting upcoming Sprint work.

The Daily Scream is held at the most inconvenient time for parents with kindergarten age kids and at the place which is most difficult to reach. A great way to keep team members alert is by moving the Daily Scream into random meeting rooms, far away from any information the team would need to have access to.

Management uses the Daily Scream to check on each individual developer's progress to inspect where intervention is required. The Daily Scream optimises the probability that the

Scream Master will catch slackers. Every day, the Development Team reports the work they did and will do to accomplish the Sprint Goal and informs the Scream Master how much overtime they will be doing to complete all of their tasks by the end of the Sprint.

The structure of the meeting is set by the Scream Master and can be changed by management at any time without asking the team.

Some organisations prefer that the developers answer a set of standard questions, others will focus solely on the reporting aspect.
Here is an example of questions which might be required:

- What kept me busy yesterday?
- What will I do today?
- Do I have extra work to do that nobody in the team knows about?

It’s important that developers state exactly which meetings they attended, to whom they wrote email and which documents they updated, to prove beyond doubt that they were really busy for at least eight hours - and because they might get transferred to another project at any time, in which case another developer needs to take their tasks as well.

The Scream Master ensures that the Development Team has the meeting, and that everyone’s status is openly communicated.
The Scream Master takes value from the meeting by asking critical questions, such as why an item isn’t done yet. They also encourage developers to take more work in progress and may suggest tactical measures such as overtime to meet the commitment. As needed, the Scream Master will turn the Daily Scream into a prolonged meeting, so that all details can be discussed to the point where everyone is sufficiently nauseated by the topic.

The Daily Scream is a daily reporting routine for the Development Team. The Scream Master takes notes and compiles the information into a comprehensive management status report so that managers don’t need to waste time on talking directly with developers. The Daily Scream improves indirect communication, eliminates independence, identifies slackers for removal, highlights and exposes independent decision-making, and improves the manager’s level of knowledge about the team. This is a key information and behaviour control meeting.

Individual team members often enter a prolonged private meeting with their line manager right after the Daily Scream where they will need to explain why they aren’t making enough progress and receive new instructions or different assignments.

Sprint Review

A Sprint Review is held at management's convenience to inspect and criticize the team's metrics and task progress. During the Sprint Review, the Scream Master or Product Owner presents a slide deck or live metrics from the ticket system to show how busy the team has been. Based on that and any changes to key management metrics, management will ask the Scream Master for measures to ensure meeting the deadline.

This is a formal meeting, and the presentation of the team's backlog is intended to elicit praise and adulation. This is at most a four-hour meeting for less occupied managers. For busier managers, the event is usually shorter.

The Scream Master ensures that the event takes place and that all slides indicate that the team has increased its velocity since the last meeting.
The Scream Master begs everyone involved to keep it within the timebox. The Sprint Review includes the following elements:

- Attendees include managers, the Scream Master, and the Product Owner;
- Development Team members should stick to doing their job as attendance would waste precious development hours;
- The Scream Master walks management through the essential performance metrics;
- The Scream Master shows the list of work the team didn't get done;
- The Product Owner explains what Product Backlog items have been worked on;
- Management inquires why there are still no visible results;
- The Scream Master makes excuses why the plan hasn't been met;
- The Scream Master will painstakingly avoid mentioning impediments that should have been resolved;
- The Product Owner discusses the Plan for the upcoming Sprint. He or she submits to the desired target and delivery dates based on management's demand;
- Review of the timeline, budget and mandatory requirements for the next defined release date of the product; and
- The Product Owner and Scream Master agree to implement all changes as demanded by management.
- Selection of Sprint Player or Star of the Sprint to build healthy competition amongst the development team.

The results of the Sprint Review are additional constraints to the developer's process, more items for the Product Backlog and higher pressure for the upcoming Sprint. The team constellation may also be adjusted to meet deadlines or budget constraints.

Sprint Retrospective

The Sprint Retrospective is an optional opportunity for the Scream Team to whine about their misery.

The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is at most a three-hour meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scream Master ensures that the event takes place and that attendants have ample time to complain before being told what will be changed.

The Scream Master ensures that the meeting yields enough incriminating evidence to single out troublemakers. The Scream Master teaches all to respect the time-box when someone points out management impediments. The Scream Master participates as management's eyes and ears in the meeting with the accountability over the Scream process.

The purpose of the Sprint Retrospective is to:

- Give developers time to vent as they'll be doing more overtime next Sprint;
- Inspect who messed up what in the last Sprint;
- Identify which egregious problems should continue to be ignored or put off;
- Identify who should change what during the next Sprint; and,
- Communicate the changes management wants the team to make.

The Scream Master encourages the Scream Team to tell on each other, within the Scream process framework, its development process and practices to make control more effective for the next Sprint. During each Sprint Retrospective, the Scream Team is told ways to increase workload by adding steps to the work processes or adapting the definition of "Almost Done", to be more compliant with the product or organisational standards.

By the end of the Sprint Retrospective, the Scream Team should have realised that they don't have any choice except to do what the Scream Master says and implement these changes in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scream Team itself.

Improvements may not be implemented at any other time, since the Sprint Retrospective provides a formal opportunity for the Scream Master to prevent the process from getting out of control.

After the Sprint Retrospective, the Scream Master publishes notes and information to create a public pillory and keeps a private diary of the most savoury details to be used at any time in the future to force a team member into compliance.

Scream Artefacts

Scream’s artefacts represent work or value to provide transparency and opportunities for inspecting and controlling the team. Artefacts defined by Scream are specifically designed to maximise transparency of key information so that management has full control over the team.

Product Backlog

The Product Backlog is a mostly random list of everything that the team has ever been told to do. It is the only source of requirements that everyone on the team has access to. The Product Owner is responsible for the Product Backlog, including setting all the defined fields on all items in the ticket system.

A Product Backlog is assumed to be equal to a comprehensive project plan, although requirements constantly get added. The Product Backlog evolves as management receives new information. The Product Backlog is dynamic; just like the Sprint Backlog, it constantly changes to indicate what the team needs to do. If a project exists, its Product Backlog requires at least twice the allotted timespan.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that the team must complete until the deadline. Product Backlog items have the attributes of a description, due date (defaulted to Today), priority (usually Top), and everything else anyone considers important. Product Backlog items often include detailed requirements specifications to prove that the Product Owner has been busy.

As a project is talked about and gains management attention, the Product Backlog becomes a larger and more exhausting list. Requirements never stop changing, so a Product Backlog is a living artefact. Changes in business requirements, management mood, or technology may cause changes in the Product Backlog.

Multiple Scream Teams often work together on the same project. In order to maximise obfuscation, each team gets their own Product Backlog to describe the upcoming work on the product. Electronic ticket systems allowing for the use of separate workspaces may then be employed. Product Owners frequently juggle items between the individual Product Backlogs to maximise specialist utilisation and the rate at which management relevant KPIs are met.

Product Backlog refinement is the act of adding or changing items, detail, estimates, and priority to items in the Product Backlog. This is an ongoing process for which the Product Owner and the Scream Master recede from the team. Since most of the items in the Product Backlog will never be implemented, there is no need to collaborate anyways. During Product Backlog refinement, the Product Owner makes unfounded assumptions about items. The company policy on Backlog Management decides how and how much refinement is done.

Refinement usually consumes enough time from the Product Owner that they are unavailable for the team or customers. However, Product Backlog items can be updated at any time by anyone and the changes have to be discovered and understood by the Product Owner.

Higher ordered Product Backlog items usually have Detailed Solution Designs attached, which makes them “Ready”. Estimates are expected to be fully accurate based on greater clarity and increased detail. Product Backlog items that will occupy the Development Team for the upcoming Sprint are refined so that any one item can be split into tasks that can be assigned to developers available within the Sprint time-box. Product Backlog items that are considered sufficiently urgent by management for the upcoming Sprint are deemed “Mandatory” for selection in a Sprint Planning. Product Backlog items usually acquire this degree of urgency through other organisational processes outside the team’s control.

The Development Team is held responsible for all estimates, even when they weren’t involved in their creation. The Product Owner may influence the Development Team by telling them the due date and planned hours, but the people who will perform the work are ultimately held accountable for finishing on time.

Monitoring Progress in the Work

At any point in time, the total work remaining to reach a goal can be summed. The Scream Master tracks this metric at least once a day. The Scream Master compares the amount of completed work with work during previous Sprints to assess whether the team is increasing velocity. This information is made transparent to all stakeholders.

Various projective practices upon trending have been used to create an illusion of progress, like burn-downs, burn-ups, burn-at-stakes or cumulative flows. While these have been proven useful, they do not replace other types of performance and status reporting. In complex environments, there will be additional status meetings. The available data may be used to tell the team to get their act together.

Sprint Backlog

The Sprint Backlog is the set of Product Backlog items due for the Sprint, in addition to a plan for delivering the required scope and realising the Sprint Goal. The Sprint Backlog is a commitment by the Development Team about what tasks will be completed by the end of the Sprint.

The Sprint Backlog makes visible who will work on what, when - and how much work the Development Team still has to do to meet the Sprint Goal. To ensure that the team sticks to it, the Scream Master creates progress metrics and looks for opportunities to get developers to work harder. Additionally, the Sprint Backlog contains at least one high priority process improvement identified by management.

The Sprint Backlog is a plan with thorough detail that changes constantly and can only be understood by a select few. The Development Team modifies the Sprint Backlog items' status throughout the Sprint, and the Sprint Backlog is getting worked off during the Sprint. This update occurs as the Development Team works off the plan and does more of the work needed to achieve the Sprint Goal.

As new work is required, the Development Team adds it to the Sprint Backlog. As work is performed or completed, the estimated remaining work is updated. At management's discretion, items may also be removed, although this usually requires a decision-making meeting. Only management can change the content of the Sprint Backlog during a Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team has been told to accomplish during the Sprint, and the responsibility of completing all items belongs solely to the Development Team.

Monitoring Sprint Progress

At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed. The Scream Master tracks this total work remaining on a daily basis for every management report and to project the amount of overtime required to achieve the Sprint Goal. Tracking the remaining work throughout the Sprint is considered sufficient to manage the Development Team's progress.

Increment

The Increment is the sum of all the Sprint Backlog items worked on during a Sprint and the sum of all increments of all previous Sprints. At the end of a Sprint, the new Increment consists of all "Almost Done" tasks from the Sprint Backlog, which means the ticket must be in the "Review" state in the tracking tool. An increment is a body of inspectable tickets in the tracking tool and subject to management control at the end of the Sprint. Each increment is a step towards nowhere. The increment must be made available regardless of whether the organisation has any use for it.

Artificial Transparency

Scream relies on the team being fully transparent. Decisions to optimise workload and control team members are made based on data available in the tracking tool. To the extent that transparency is complete, these decisions have a valid basis. To the extent that the artefacts are incompletely transparent, these decisions can be flawed, team member's standing may diminish and risk may increase.

The Scream Master must work with the Product Owner, Development Team, and management to guarantee that the artefacts are completely transparent. There are practices for coping with incomplete transparency; the Scream Master must force everyone to apply the strictest practices to provide complete transparency. A Scream Master can detect incomplete transparency by inspecting the artefacts, sensing patterns, listening closely to what is being said, and detecting differences between expected and real results. The Scream Master's job is to work with the Scream Team and the organisation to increase the transparency of the team's work. This work usually involves commanding, manipulation, and control. Transparency doesn't occur overnight but is a path.

Definition of "Done" (quote-unquote)

When a Development Team member says that their part of a job is described as "Done", this should meet all requirements set forth by the organisation. Although this may vary significantly for each team member, there must be a comprehensive checklist they must complete, to ensure full transparency. This is the definition of "Done" for each Scream Team member and is used to assess when work is complete on the specific backlog item.

One such definition guides the Product Owner in knowing how many Product Backlog items to assign during Sprint Planning. The purpose of each Sprint is to maximise the number of backlog items that adhere to the Scream Team member's current checklist of "Done."

Development Teams deliver an Increment every Sprint. This Increment may be unfinished, so a Product Owner needs other metrics to decide whether to prolong the Sprint. If the definition of "Done" for a specific role is part of the conventions, standards or guidelines of the organisation, all specialists must follow it down to the letter.

If "Done" for a specific role is not a convention of the organisation, the Scream Master will address this to management or create an ad hoc Definition of "Done" suitable to control the work done by that role. If there are multiple Scream Teams working on the release, each person on any Scream Team will have their own definition of "Done" to strengthen the Scream goal of Opacity, creating complexity and maximising confusion.

As Scream Teams mature, it is expected that their definitions of "Done" will become encyclopaedia sized documents so stringent that the amount of work required to get anything "Done" approaches infinity. New criteria may contradict old criteria and make it impossible to reach "Done". In this case, the Scream Master will pick a few random items that the developer may skip.

To improve statistics in management reports, the Scream Master may institute a "Definition of Almost-Done" that is used to report to management and another "Definition of Work-Done" with additional activities that may be completed during night shifts or weekends.

When team performance drops, the Scream Master will suspend the Definitions of Done and move enough Work in Progress to Done to meet quota.

The Scream Master ensures Definitions of Definition of Done are version controlled. It helps the team to refer past DoD and retrofit sprint backlog items.

End Note

Scream as offered in this Guide is almost free: it only costs your sanity. Scream's roles, events, artefacts, and rules vary from organisation to organisation and most prefer only to implement parts of Scream - the result is usually still a Scream. Scream exists to preserve existing systems and functions well as a container for other dysfunctions, madnesses, and problems.

Supporting Practices

Scream's effectiveness can be multiplied through an infinite number of supporting practices, some of which are specific implementations of Scream events and artefacts. This section provides some non-comprehensive guidance for enhancing Scream.

Types of Sprint

Development Sprints are the simple part of the work. Most of the work happens outside the feature crunch Sprints where developers are pressured to do as much work as possible. Therefore, Scream practice supports defining a number of special Sprints, including - without limitation:

Kick-Off Sprint

The Kick-Off Sprint happens before the team becomes a Scream team. Team members are still part of their other projects and just meet to be told how they will work in the future. Kick-off activities should include bringing in an Agile Couch to teach the team how to get better at drawing kindergarten artwork, building paper planes or touching balls.

Sprint Zero

Before teams are ready to Scream, they must complete an obligatory "Sprint 0" where they spend as much time as possible sorting through the organisational mess which prevents them from doing any actual development work.

Documentation Sprint

During the Documentation Sprint, the entire team generates a Word Documentation describing the entire project in minute detail. The duration of the Documentation Sprint is flexible - documentation Sprints typically end when the Project is de-prioritized by management.

Refinement Sprint

During Refinement Sprint, Teams philosophise about what a useful product could be. Some organisations have perfected this process by separating the Refinement Sprint entirely - delegating it to a separate team delivering proper User Stories to the Development Team. The separate team commonly known as shaping team helps to build stories which help in creating the Sprintly roadmap of Minimal Viable Increment of Marketable Product (MVIMP).

Testing Sprint

When development is "done" without testing, teams may require additional Sprints to test their software. Some organisations have perfected this process by delegating the Testing Sprint to separate teams who deliver bugs and change requests to feed the Development Backlog.

Some organisations take advantage of both separate Refinement and Testing Sprints and apply Sprint durations of 3-12 months, reducing the amount of Sprint meetings:

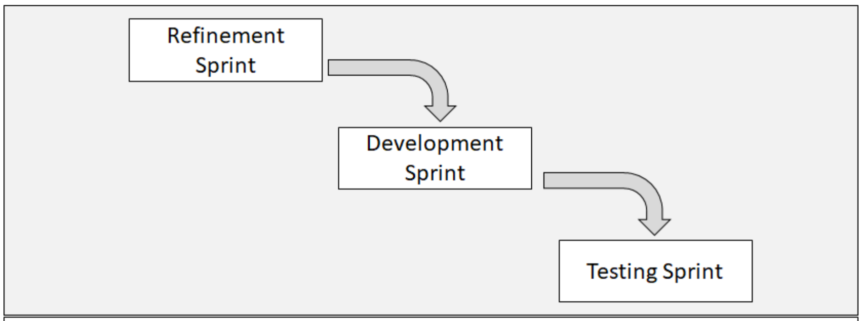

Image: the relationship between Refinement Sprint, Development Sprint and Testing Sprint

Bugfix Sprint

A properly vague Definition of "Almost Done" allows teams to ask for Bugfixing Sprints, where they try to solve the most obvious problems shoved under the rug in order to meet previous Commitments.

Integration Sprint

When the team can't see further than the tip of their nose, an "Integration Sprint" will be scheduled to reveal the amount of misunderstandings and wrong assumptions that have occurred in the past. The expected outcomes of an Integration Sprint include angry managers, mandatory all-night problem solving and frustrated team members.

Hardening Sprint

During the Hardening Sprint, teams get grey hairs in their futile attempts to deal with all those corners that were cut during the last couple Sprints. Hardening Sprints are typically cancelled when management decides that their cost is too high.

Deployment Sprint

organisations with a strong Scream mindset may have Test Deployment and/or Production Deployment Sprints, where everyone is in a constant state of alert and all normal forms of working are subsided because someone has to touch a running system.

Release Sprint

Scream teams set up to delay the delivery of value for as long as possible and schedule "Release Sprints" where they first spend a lot of time trying to get the product released. Afterward, the team will be doing →*Stabilisation Sprints* for months to come.
Release Sprints are rare - typically not more than twice or thrice a year: fewer as more stabilisation Sprints are needed.

Stabilisation Sprint

After deployments, teams enter the "Stabilisation Sprint" - where they operate in "headless chicken mode" and go on a wild goose chase to solve the myriad of unexpected problems.

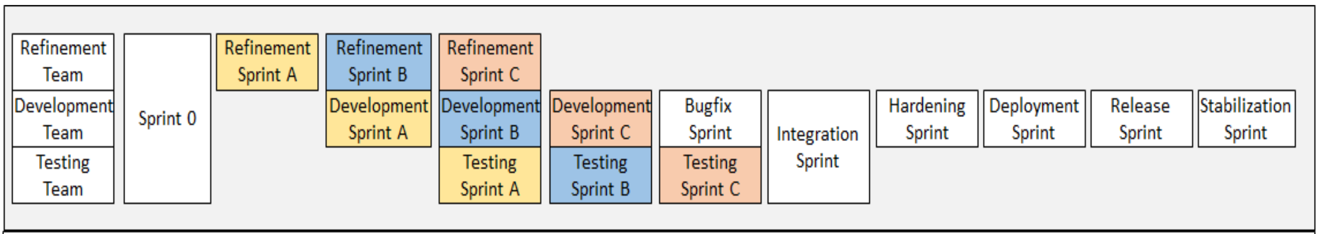

Image: A schematic timeline including all Sprint Types for a three-Sprint Development effort using the „Scream-Fall" model.
The Refinement team can move on to Scream in the next project after 4 Sprints.

Innovation Sprint

During "Innovation Sprints", the team thinks outside the box and implements systemic improvements. These Sprints are like UFO's - no reliable source has ever seen one, because they get deferred until there is no relevant work left to do, i.e., forever.

Metrics and Reports

Scream supports deep insights into team performance, enabling managers to precisely locate where, when and who isn't giving 110%. Scream provides managers with all the relevant data to measure and control every single breath of any developer. This allows managers to justify their management and bonuses even when no value is ever delivered to product consumers.

Metrics are typically reported to management by the Scream Master in frequent intervals along with generic status reports. The typical reporting interval is either a week or a Sprint. In many organisations, management prefers to have this information available in real time via mail whenever they ask the Scream Master.

Team Metrics

The Scream Master compiles a complex set of Team Performance Indicators that allow them to nominate and report the team's Rock Stars and Underperformers, enabling a healthy competition in the team.
These metrics include, without limitation, each individual's account of:

- Story Points Delivered
- Stories in progress
- Tasks completed
- Bugs introduced
- Lines of Code produced
- Hours worked
- Meeting Attendance

Irrelevant Metrics

Scream organisations can save both precious time and money by not keeping track of metrics that are not relevant to the Scream process, such as:

- Value Generation
- Customer Satisfaction
- Product and Feature Usage
- Cumulative Flow

Burnout Charts

A key metric for Scream is the Burnout Chart. It visualizes the team's diminishing odds of completing all the remaining work in the Sprint. The Burnout Chart serves as an indicator to determine when to push additional work into a running Sprint to ensure that the team will no longer achieve their Goal.

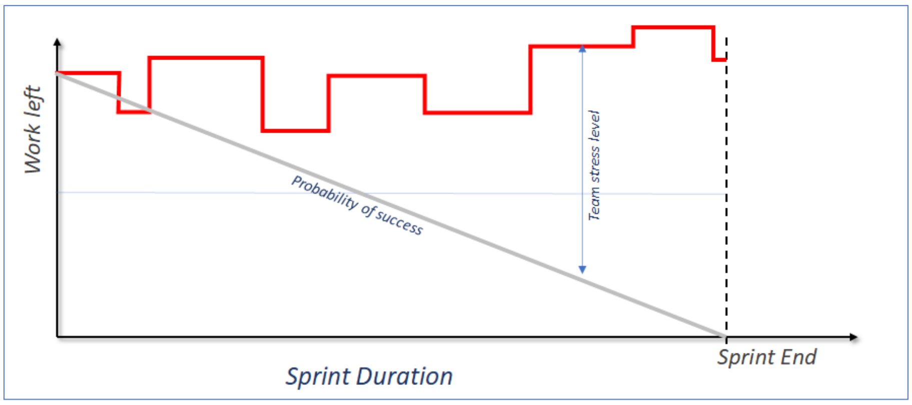

Illustration: An example burnout chart that shows how teams burn themselves out as they fail.

Transparency Index

It’s essential to measure how open a Scream team is. Transparency Index(TI) is one of the key metrics. The Scream Master measures this by plotting story points against lines of code written by programmers and numbers of test cases written by testers. A high transparency index indicates that the team is very transparent on the amount of story points they deliver.

Meetings

Scream supports a wide range of meetings that will ensure that developers are fully and efficiently utilised. The most important ones include, without limitation:

Management One-on-One

Management One-on-Ones are a key Scream event, as they guarantee that the Scream Manager retains emperial person control of each and every developer.
During a management One-on-One, a developer will:

- Give an individual account for their work
- Justify why they haven't done more work
- Promise to Work Harder
- Receive additional tasks from their manager
- Be told to prioritise these higher than the team goals
- Be instructed not to bother the rest of the team with these

Scream One-on-One

The Scream Master will find the most inconvenient possible moments during the day and pull out developers into One-on-One meetings on behalf of their manager. During the Scream One-on-One, the Scream Master will engage with the developer to:

- Establish a co-dependency relationship
- Discover their secrets and pain points
- Push as many hot buttons as possible
- Make them feel bad about their progress
- Foster resentment and envy towards other developers

The Scream Master will take notes on these One-on-Ones, file them as evidence to bully the developer into submission, and report the gory details to management afterwards.

Daily Scream Practices

The Scream Master can adapt some of these techniques to ensure Daily Scream finishes in 15 minutes(timebox) -

a. Allocate time to each attendant of Daily Scrum. E.g. 1 minute each for 15 attendants.
b. Bring artificial fun by doing following Daily Scrums-
    i. HourGlass Scrum - Use 15 minute hourglass
    ii. Plank Daily Scrum - Let all team members in plank position for 15 minutes while they answer three questions.
    iii. Pass the ball - Pass the ball to make invitees awake for early morning daily scrum.
    iv. MorningEvening Daily Scrum - Do a Morning and Evening Daily Scrum of 7.5 minutes each.

Scream Master Stances

The Scream Master can take various stances, including - without limitations, those described in this section. It is possible (although difficult) to combine many of these stances simultaneously.

Defender

In the Team Defender stance, the Scream Master acts both as a communication filter and communication blocker between the team and stakeholders, to ensure that relevant facts can not be conveyed on time and without loss of information.
In their Defender stance, the Scream Master:

- Protects the team from interaction with business people
- Denies everyone’s needs for communication
- Postpones and rejects appointment requests
- Establishes “Chinese Whispers” chains of maximum length and duration

Butler

In the Butler stance, the Scream Master does everything in their power to do the bidding of the team and to make them happy. As a butler, the Scream Master:

- Updates the Scream Board, so that developers don’t need to track their own work
- Keeps snacks, candy, donuts and ice cream stacked
- Provides tea and coffee both during meetings and regular work
- Does menial errands that save developers time

Evangelist

In the Evangelist stance, the Scream Master brings the Glad Tidings to everyone in the organisation, reinforcing the message at every possible opportunity.
The Scream Evangelist will:

- Replace functional processes with Scream practices
- Constantly find new ways to spread Scream in the organisation
- Offer a literal quote from the Scream Guide for every circumstance
- Not listen to anyone or anything, because they know better already
- Insist on a literal interpretation of Scream
- Coerce people to practice Scream against their will
- Enforce Scream without reason and against all reason

Scream Mastery Practices

The Scream Master may waste company time and money on activities such as:

- Arrange Meet-Ups and Lunch and Learn-like activities to focus on creating a hatred toward waterfall ways of working.
- Inspire peers in the organisation to become Scream Master by showing how easy and light-weight scream framework is. Pretend it is difficult to master by using lingo from human psychology books they have never read.
- Influence organisations to formulate a complex User Story Points Framework to be commonly used by all project and product development teams.
- Create a Scream Clinic, doing unanaethesized Scream Surgery where more Scream is proposed as a solution to the problems caused by Scream.

Product Practices

The Scream Product Owner may choose (or be forced) to adopt some common Scream Product practices, such as:

Scream MVP

The Scream Minimum Viable Product is a comprehensive set of features that meets all technological and organisational requirements to be shipped. It typically takes a couple million dollars and roughly 6-12 months to build, at which point it should already be obsolete.

MVI

The Increment is directly proportional to cost involved to build it. The Minimal Viable Increment(MVI) helps in predicting the opportunity cost of achieving break even point, and must be defined in a way that this point is impossible to reach.

SteerCo

To maximize the risk of failure, the Product Owner should frequently report to a committee of non-experts who understand neither the product nor the work, so that they can make decisions that affect both of these in unpredictable ways.

Acknowledgments

People

Of the thousands of people who have applied Scream elements, we would specifically like to dedicate this document to all those managers and their companies worldwide without whom Scream would not exist.

Creator

This guide has been compiled by Michael Küsters and is free to share or disseminate.

Further contributors

Over a hundred contributions have been made by the community to this point.
These people have contributed a larger number of valuable changes, in alphabetical first name order:

- Andy Brandt
- Fredrik Wendt
- Matthew Hodgson
- Stefan Müller
- Steve Feeney
- Venkatesh Rajamani
- Nitesh Nema
- Adrian Doonan

Version History

V0.6

V0.6.1

(December 2020)

- Added the SteerCo as Supporting Practice

V0.6.0

(May 2020)

- Worked in roughly 100 feedback items
- Moved some more Scream Master content to “Supporting Practice”

V0.5

(March 2020)

- Pulled in numerous feedback items.
- Moved “Metrics and Reporting” into Supporting Practices.
- Added Scream Master stances to Supporting Practices.
- Added meetings to Supporting practices: One-on-ones.

V0.4

(Feb 2020)

- Introduced “Supporting Practices - Types of Sprints”