Scaling up

Kim Tielen
Mo Marghich

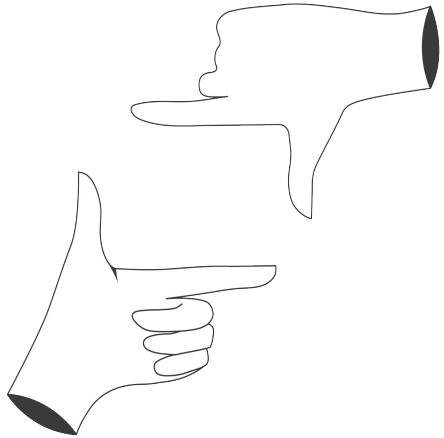
Emakina's guide for how to transitioning from one to multiple scrum teams to increase business value output.

How to start?

Before scaling up to multiple Scrum teams, there are a few fundamental decisions you have to make for a smooth transition. It is important that the new way of working is clear for everyone in order to make the transition a success. In this playbook we highlight what these decisions are and we show different approaches, with their pros and cons.

The topics we cover in this playbook are:

1. When and why to scale up?
2. Work division between teams
3. PO Role and Backlog Management
4. Sprint Rhythm and Releases
5. Roles & Responsibilities
6. Technical Dependencies
7. How to Measure Success
8. Main pitfalls / what to watch out for

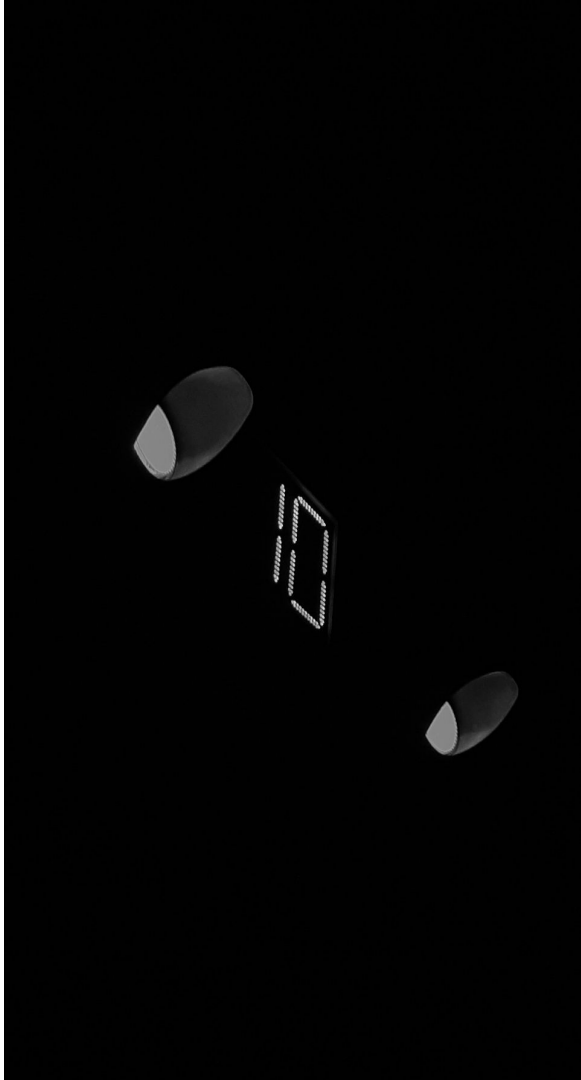


1.

When & Why to Scale up?

Digital transformation is a must

Digital transformation is a focus for more companies because of changing customer buying behaviour and regulations. This strategy means scaling digital teams to improve delivery output and capability is a must. However, increasing team size has its limitations, this is where splitting teams can offer solace.



Why consider scaling up?

You can have several reasons to decide to go from one to multiple Scrum teams. Scaling up can have a large impact on the current way of working and the team's output. Therefore, the right preparation is key.

It is important that you know your company's motivational reasons to make certain decisions within the setup of these Scrum teams. In chapter 7 we describe how you can set clear goals. Below are some of the more common goals for scaling up Scrum teams.

Main goals for scaling up Scrum teams:

- ➔ Increase velocity and fulfill business needs
- ➔ Managing Roadmap items and depencies simultaneously
- ➔ Reduce the number of communication lines within the team
- ➔ Get a better oversight of the workload
- ➔ Enhance overall Quality Assurance
- ➔ Be able to scale up and down based on actual business needs

What does the Scrum Guide say?

The Scrum guidelines were written to help Scrum teams make the best possible use of the Scrum framework to deliver maximum business impact. When you are in a transition phase we recommend keeping the most important Scrum guidelines at hand so you create the best possible setup.

Based on our experience of scaling up Scrum teams we've highlighted below what we think are the most important guidelines:

A team:

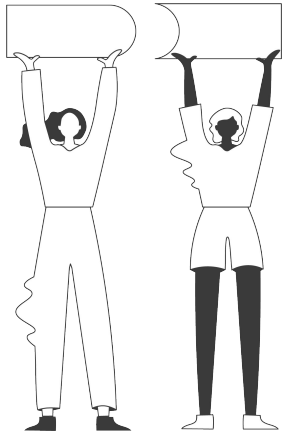
- ➔ ...consists of 3-9 people
- ➔ ...is self-managing and cross-functional
- ➔ ...determines its own velocity
- ➔ ... has all the Scrum roles covered
- ➔ ... collaborates directly with the client on a daily basis

2.

Work division between teams

How to split work between teams?

There are many ways to re-organize your Scrum team(s) when you scale up. In this Playbook we work with 3 alternatives that can be used as a foundation to differentiate in your desired way of working.



Components:

Expertise teams with a specific focus area (of a platform or product)

Workstreams:

Teams working on different workstreams: development, Support, Rollouts

Feature teams:

Fully equipped teams with equally divided knowledge and expertise that can develop E2E

Overview of the 3 different options

Type of split	Pros	Risks
Components	+ clear separation of concerns + feature experts in the team + efficient QA process	– expertise on feature level – inconsistent workload – keeping the work challenging enough
Workstreams	+ clear separation of concerns + focus on the operations + manageable outcome per stream	– inconsistent workload per stream – fragmented expertise – work not challenging for developers
Feature teams	+ consistent workload + spread knowledge + challenging work for team member + flexibility in team changes	– increasing overhead (PM, SM) – additional roles required (Dev, QA) – change on client organisation side (PO)

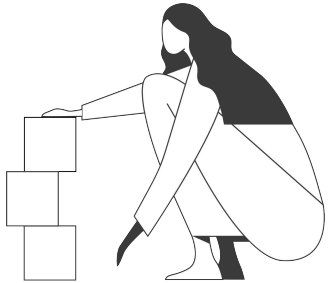
Main takeaways

- Don’t base the division on the organisation or departments. Enable teams to deliver the most value.
- Determine how the type of workload is generated and how it should flow through the teams.

- It’s hard to make the right decision at once. Experiment with a small team first and apply the learnings.
- Involve the customer and/or stakeholders in an early stage to make them part of the solution.

PO Role & Backlog Management

How to manage your backlog with one or multiple Product Owner's



There are many ways to re-organize your backlog and PO role when you scale up. Here are 4 alternatives that can be used as a foundation to differentiate in your desired way of working.

1 Backlog with 1 PO:

Multiple Scrum teams working with the same product owner and from the same backlog

1 Backlog with multiple PO's:

Multiple Scrum teams with each its own product owner, but working from the same backlog

2 Backlogs with 1 PO:

Multiple Scrum teams with each its own backlog, but working with the same product owner

2 Backlogs with multiple PO's:

Multiple Scrum teams working on separate backlogs, each one managed by different product owners

Overview of the 4 different options

Option	Pros	Cons
1 Backlog, 1 PO	+ Single point of contact team + stakeholders	- Workload PO - Scattered backlog
1 Backlog, multiple PO's	+ Scalable + Works in large setups	- Scattered backlog - More overhead - Decision making
2 Backlogs, 1 PO	+ Single point of contact team + decision making + Clear split in work	- Workload PO - Additional coordination
2 Backlogs, multiple PO's	+ Scalable + Clear split in work	- More overhead - Decision making

Main takeaways

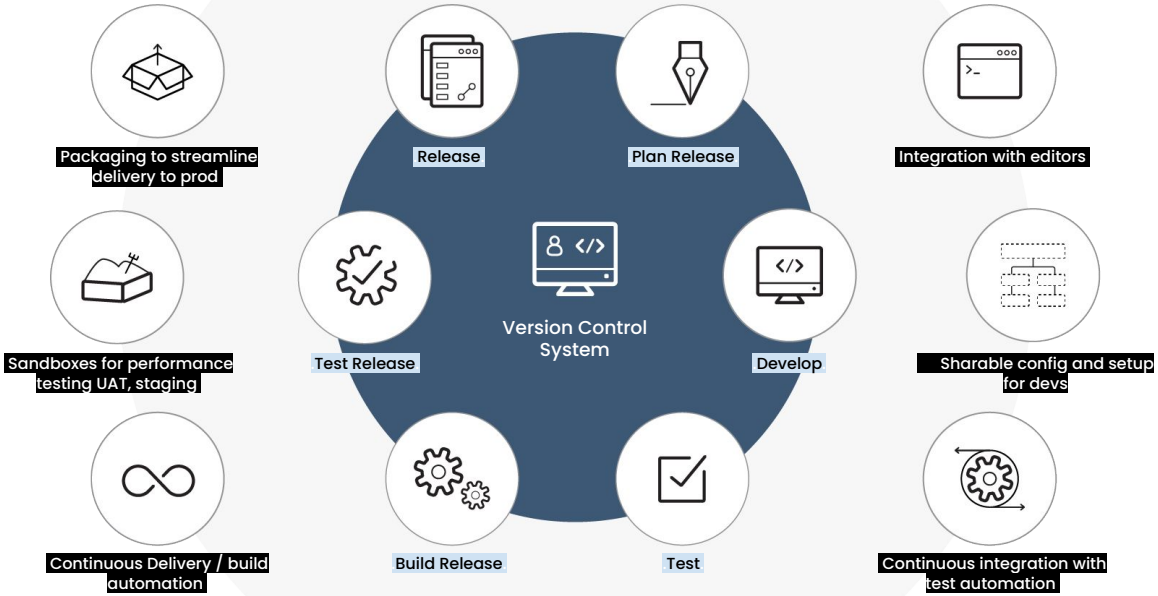
- A clear overview of the workload of each team is needed to be able to plan ahead
- Each team should have its own velocity
- Make a clear split in code and/or features to avoid dependencies between the multiple Scrum teams

- Keep the amount of communication lines as little as possible; less is more
- Make sure to make clear agreements about decision making
- Keep stakeholders up to date about the new team setup
- Make sure your setup is future proof and in line with the company's strategy

4.

Sprint Rhythm & Releases

Application Lifecycle Management

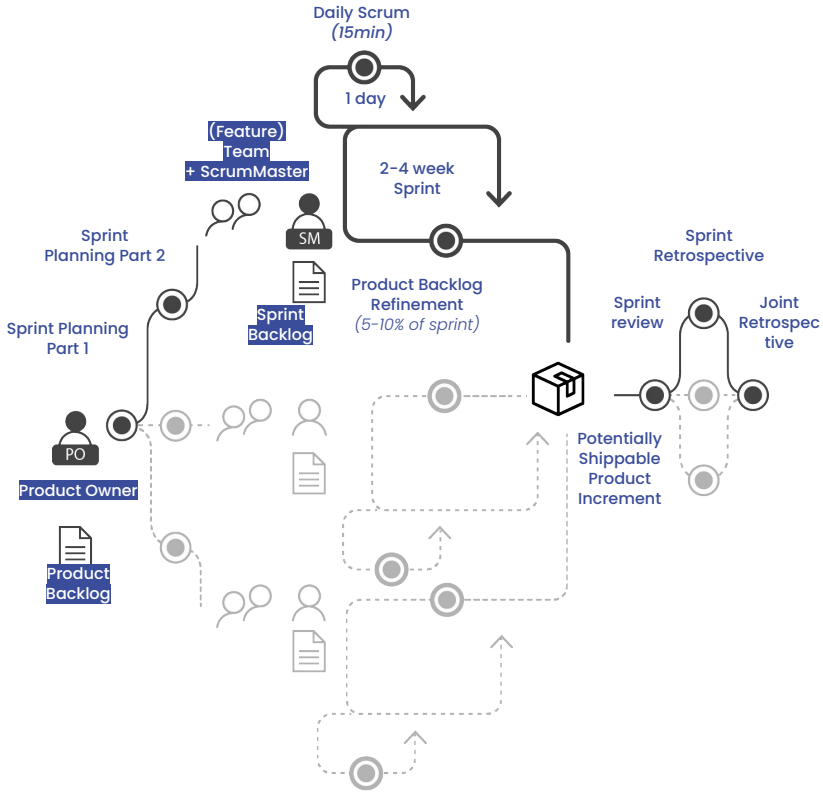


Sprints

How do sprints work with multiple teams? If you want to benefit from Scrum when scaling up, keep the same way of working and extend that 1-on-1. The organisational process around the sprints will change a little but will not affect the productivity in a negative way.

One of the proven ways is LeSS *(Large Scale Scrum)*:

- The sprint rhythm can vary *(2 to 4 weeks)*
- End result of the sprint is a *(potentially)* shippable Product Increment
- Most efficient if one single PO is assigned to multiple teams
- Joint overall Sprint Planning session and separate ones on team level
- Refinement session per team
- Joint Sprint Demos
- Retrospective on team level
- Retrospective *(PO, SM, project sponsor)*



Main takeaways

- Use a proven methodology to scale up
- Stay close to your current way of working
- Agree on how to embed QA in the process (in the sprints or after releasing to acceptance)
- Teams have conflicting priorities across their individual backlogs

- Scope is dynamic so dependency management is ongoing
- With short release cycles, there is less time to coordinate
- Team completion dates may not align

5.

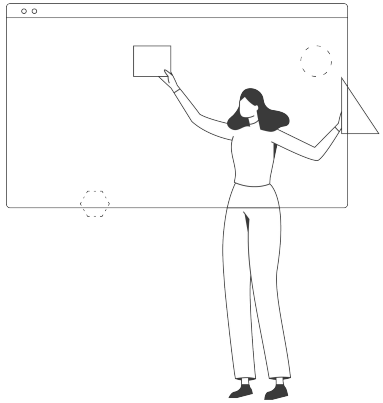
Roles & Responsibilities

How to define?

When the processes around the multiple Scrum teams are defined, it's time to review the roles & responsibilities from each team member. When you do this, it is important the whole team works together to clarify individual responsibilities and to find unowned gaps that need to be filled. Use a whiteboard with sticky notes to collect the observations from your team, or use an online tool like Trello.

To be able to define the roles & responsibilities of your Scrums team(s), you can follow the following steps during your session:

1. Identify roles
2. Identify responsibilities
3. Discuss role responsibilities
4. Review unowned responsibilities
5. Summarize and document
6. Review this exercise on a regular basis or when the team experiences change



Tips & tricks

1. Identify roles

Some people may have more than 1 role. Makes sure you collect all roles.

2. Identify responsibilities

Focus on responsibilities related to roles, not people.

3. Discuss role responsibilities

Collect the 3-5 most important responsibilities per role.

4. Review unowned responsibilities

When you have a lot of unowned responsibilities, this could mean a new role is needed or an existing role needs to be revised.

5. Summarize and document

Make sure that each individual agrees to what has been decided.

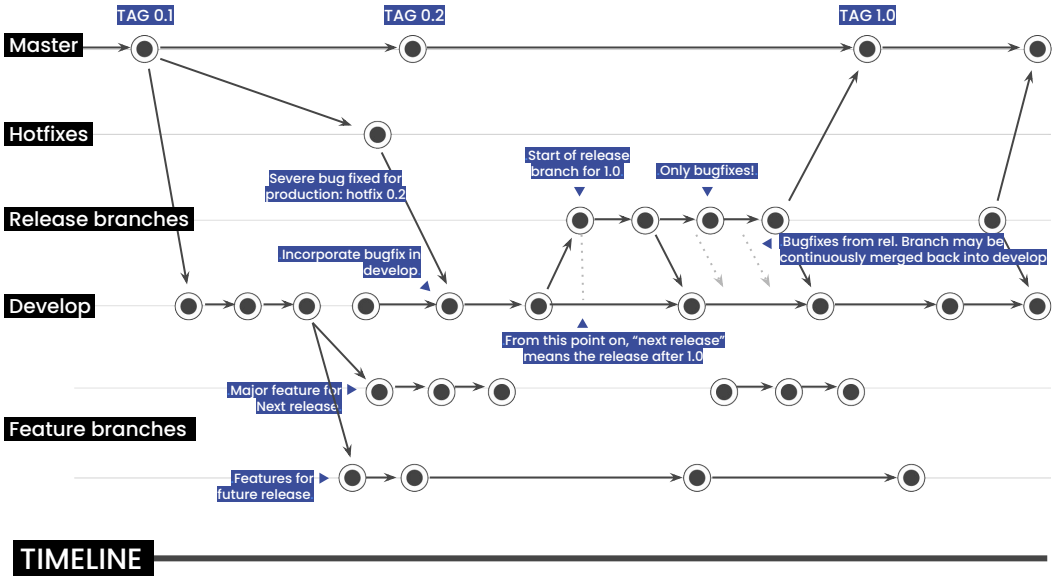
6. Review this exercices on a regular basis or when the team experiences change

3 to 6 months is usually a good timeframe to review this exercice.

Technical Dependencies

Branching strategy

Working with multiple teams on a single code base requires a clear approach for handling version control in the development team.



Consider the following:

- Document and communicate the branching strategy to your team about when/where a developer should branch and when/where to merge
- Use version numbers to distinguish hotfixes from major and minor releases.
- Agree on the code review process when merging (integration tests). Integrating code earlier, or integrating across teams one at a time, allows contributors to quickly isolate an issue.
- Do you support multiple versions of your software? Ensure bug fixes are synced across the release branches to avoid regression
- Automated tests can help validate (main) features, detect issues quickly and let developers focus on solving possible code conflicts

7.

How to Measure Success

Set goals

If you go from one to multiple Scrum teams you want to be able to determine if the split is a success. To be able to measure success it is important that you define the most important goals and KPIs for your organisation.

In chapter 1 we gave some possible goals for scaling up to multiple Scrum teams. These goals can be applicable for your organization as well, but other goals might be more relevant. It is important that goals have clear intentions, to help you stay on course. A framework you can use for setting goals is the SMART framework *(Specific, Measurable, Achievable, Relevant, Timely)*.

- **S**pecific:

  To make a goal specific you can answer questions like What? Who? Where? When? Which parts? and Why?

- **M**easurable:

  Ensure your goals have an observable result, with quantitative data like analytical data, performance measures, or direct revenue attached.

- **A**chievable:

  Identify the key benefit by answering the “Why” question, to make sure your team members understand the importance of the goal and how it contributes to the bigger picture.

- **R**elevant:

  It is important that your goals are feasible and meaningful to motivate people and release energy.

- **T**imely:

  Check if you have set a deadline to your goals to create a sense of urgency.

- **SMART**

Review goals

To be able to track your goals and to evaluate the success of the team split, you should set KPIs *(Key Performance Indicators)*. In other words, KPIs are the answer to the 'how' question. But how do you set actionable KPI targets? Below you can find the process we use for setting actionable KPI targets to be able to continuously measure success

➔ **Review your business goals**

➔ **Analyse your current performance**

➔ **Set short and long term KPI targets by using lagging and leading indicators**

- Lagging indicators: the outputs or results that you are getting, like revenue
- Leading indicators: the actions that lead to your results, like the unique website visitors you have per month

➔ **Process benchmark your KPI's by evaluating your KPI's in relation to best-practice companies' processes**

➔ **Review progress on team and/or management level and readjust**

8.

Main pitfalls/what to watch out for

Scrum is easy, why would teams fail?

- **Changing** people's (work) habits requires persuasion, training, patience and practice. More people means more communication and coordination.
- **Quality** Assurance is key: rule rule of thumb is to allocate 30% of the development work for testing. All the untested work will result in reported issues later.

- A common mistake is to scale up with a **weaker** team with less experienced team members. This has a negative impact on the ambitions and further development of those team members.
- Look at a way to **deliver** the most **value**. Aim for a team division with a constant workload. Too much work will stress out the team, too little work will cause boredom.

- People always like others to change. **Involve** your **customer** early about the required changes to make it work. There might be new roles needed.

We are here to support

Let’s get started! Contact us.

EMAKINA@HOME
Your direct contact is **Kim Tielen**
+31 (0)6 52 18 13 02
or send her an email at
k.tielen@emakina.nl

EMAKINA Office
Danzigerkade 4, 1013 AP Amsterdam,
Netherlands
www.emakina.nl