**Nexus Framework *(FILE CHANGE)***

1. **Brief description of what it is and aims to achieve**

Nexus is a framework that focuses on complex product development in which multiple teams work on the same product or product family.

1. **How it works.**

Nexus focuses on 3-9 Scrum teams, which as a whole are referred to as the Nexus.

Fewer teams don’t really need the overhead of a scaling framework. More teams will be enabled to work together effectively.

A Nexus consists of teams who have the skills required to deliver an increment of working software.

Nexus allows for the existence of component or layer teams, but all work done in the Sprint across all teams must be integrated and deliver value together.

A Nexus works on a single product or product family.

The main artifacts of Nexus are the same as those in Scrum, with the addition of the Nexus Sprint Backlog, which shows the dependencies between the Sprint backlogs of the individual teams.

Nexus includes a Nexus Integration Team (or NIT) which consists of the Product owner and those members of other teams required to facilitate integration.

1. **What are strengths of the framework? Not a solution.**

Improved management of teams if up-scaling.

1. **What are the weaknesses?**

It’s not a solution in its own right, it’s a framework to assist in coordination of multiple development teams.

The Nexus Framework presumes Scrum experience.

1. **Is it a good framework for scaling up DevOps? Why? If not, why not?**

Nexus is an effective way to scale Scrum; it was developed by one of the original founders of Scrum. Unlike SAFe which enforces a lot of structure and process on organizations, it takes into account that not all organisations are equal and that they don’t have to be in order to scale effectively. It introduces a number of new concepts to help teams collaborate effectively when they grow past a certain size (generally when there are three or more teams), but it tries to limit its prescriptiveness to a minimum.

**Spotify Framework**

1. **Brief description of what it is and aims to achieve, and**

Autonomous ‘squads’, with up to 8 members all working in the same area, decides everything except

(a) Squad Mission,   
(b) Product Strategy, and   
(c) renegotiated 3-month short-term goal.

Optimised office area with easy access to each other’s screens, and nearby lounge for planning sessions and retrospectives.

1. **Brief description of what it is and aims to achieve**

Motivation makes it efficient.

Locally made decisions so no hand-off delays with other decision makers, so it’s fast.

Its easier to scale without dependencies, coordination and approvals by committees, etc.

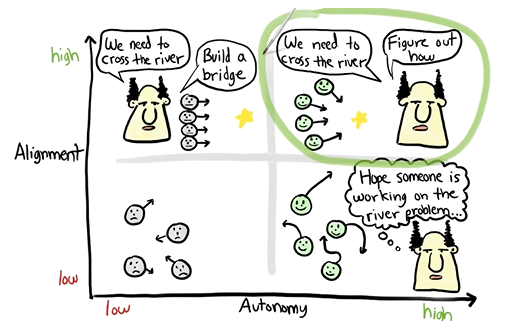
Each squad has its own mission but their Mission Strategies need to be aligned with company strategies.

Spotify’s overall mission is more important than the mission of any single squad.

Squads listen to each other wrt the Spotify mission.

Key: Loosely coupled but tightly aligned squads.

Experiments used to investigate ways of squads getting closer.



1. **How it works.**

Each ‘Squad’ is a member of a ‘Tribe’ with cross-squad chapters with similar Competencies.

The most valuable communication occurs in unpredictable and informal ways.

‘Informal Guild’ relationships exist between staff with ‘Communities of Interest’ mailing lists and conferences.

Small, frequent releases and use of test automation and continuously releasable infrastructure.

Squad decoupled releases with less interdependencies.

3 types of squads: Infrastructure, Client Apps and Feature Apps.

Self-service squads also developing support for other squads to self-serve.

Regular release trains so NO politics and NO failure.

Squads release into Prod without control.

Make mistakes faster and learn faster,

Fail-friendliness.

Failure wall. Failure not closed until leaning complete. Lots of retrospectives.

Gradual roll-out to limit risks, increase courage and innovation.

If everything is under control, we’re going too slow.

Build-Release – each week.

1. **What are strengths of the framework?**

Very strong culture of general respect and helpfulness. And very high job satisfaction.   
This means NO politics and NO failure.  
Very fast.

1. **What are the weaknesses?**

Lack of standardisation between squads.

Architecture includes over 100 different systems.

Risks of building the wrong thing.

Can’t plan. No predictability.

Not good at big projects.

Growth pain and chaos and increasing structure causes bureaucracy hold-ups.

No release control outside of each squad – a difficult culture to introduce into a traditional organisation.

No formal company organisational structure so it will be difficult to change an organisation to fit this model.

1. **Is it a good framework for scaling up DevOps? Why? If not, why not?**

It is a good framework albeit with perseverance of scaling up pains and the necessary integration of teams and the coordination complexity that brings with it.

Perhaps Nexus could be used as a framework for integrating larger projects.