How are great products created?
[pause about 2 seconds]

To develop a great *small* product, we'd sit together with our team and the customer, give them a new version every few minutes, and work together on the technology to solve their problems. You already know this is the best way to develop small products.

But what usually happens with *large* product development?

Typically we get boring specifications from an internal department, then hand over our work to another internal department.

We interact with managers instead of real customers, don't have much say in the process, don't have much effect on the whole product, and lack a sense of purpose.

We might even believe we're doing Scrum or some other Agile methods, but our organization's inability to learn and adapt to reality will bite us sooner or later.
[pause about 2 seconds]

What was better about small scale development?

Our approach was customer centric. We saw the whole product at once. We had minimal process, and we had effective reality checks about the product and our way of working. It was natural.
[pause about 2 seconds]

Is it possible to take what works so naturally in small scale development and apply it to large scale development?

This question has driven Craig Larman and Bas Vodde [pronounced Bus] for over a decade. [Bas wrote "Better not try to pronounce my name correctly as it is even more wrong than the typical English pronunciation. (last name, that is. First name is ok)"] It's led to hundreds of experiments and the formulation of LeSS.

LeSS is a simple framework for scaling agile development using these principles and some others.

LeSS has been used with groups of 12 people, hundreds of people, and thousands of people on products from banking to telecom, games to radar systems.

How does LeSS work? LeSS is multi-team Scrum.

There's one Product Owner providing vision and one adjustable prioritized list of customer centric items, the Product Backlog.

We want one integrated, shippable product increment every Sprint – every 1-4 weeks.

Multiple teams develop this product one shared Sprint at a time.

Development is iterative and incremental.

Each Sprint starts with Sprint Planning 1, a short shared event where each team selects features from the top of the Product Backlog that they will implement during the Sprint.

That's followed by Sprint Planning 2, where the teams discuss their strategies for developing their features.

During the Sprint, each self-managing team develops the features they selected while collaborating and continuously integrating with the other teams on the Potentially Shippable Product Increment.

Coordination outside the team is now a team responsibility, so there are no assigned coordinators.

Halfway through the Sprint, teams briefly pause the current Sprint work for Product Backlog Refinement -- collaboration with customers and end users to clarify work for future Sprints. By connecting teams to customers we free up the Product Owner for vision and prioritization.

We have one Sprint Review, a shared session where teams and customers explore what was done and determine the best next increment to develop.

Each team retrospects to inspect and adapt its own way of working.

We want teams to *own* their methods and processes, not *rent* them. Without ownership, there can be no continuous improvement. And in LeSS we don't stop with the team retrospectives.

Teams, Product Owner, Scrum Masters, and management use the *Overall Retrospective* to explore the systemic and organizational obstacles that prevent higher value delivery. Use LeSS to inspect and adapt the entire organization.

Then they repeat the whole cycle, experimenting and making wiser mistakes each time.

When we have more than 8 teams, we do something similar called LeSS Huge, still targeting one shippable product every Sprint.

So that's the LeSS Framework, just a few rules you can learn in a few minutes.

[pause about 2 seconds]

But it won't work in your organization!
[record scratch]

Not the way your organization is *now*.

Your organization would have to change into one that allows it to work. This isn't quick or easy.

You and your organization are currently designed to resist, undermine, and neutralize any steps toward real agility.

[pause about 2 seconds]

The few rules that LeSS has are impactful, with far reaching implications. Adopting LeSS can take years of stripping away the *existing* organizational rules, processes, structure, and habits.

LeSS adoptions aren't just process changes or roles added to your existing organizational structure. You've already seen Agile labels applied to existing habits with no real change. LeSS adoptions go deeper, challenging conventional wisdom about projects, products, roles, technical practices, and management practices.

We allow agility to work by changing structure and policy. For example, the teams must be *cross functional*, which means they don't only code and test. They also include software design/architecture skills, business domain knowledge, and UX/UI design skills.

The teams are responsible for requirements clarification with the customers and end users.

The teams also must be *feature teams*, which means they can develop end-customer centric features, not just internal components. LeSS teams span components and work in a shared code environment.

[pause about 2 seconds]

Organizations often try to solve immediate problems by adding complexity. For example, let's say our product crashed in production because a team neglected to run a test. My first reflexes as a manager would be to impose prescriptive process steps on the teams, assign someone to a new specialized role, or form a new department to prevent this error in the future. But all three of these quick fixes can exacerbate the underlying problems by reducing team responsibility.

Giving responsibilities to processes, specialized roles, or other departments takes responsibilities away from the team. Companies wind up with employees who are just mindless zombies.

We believe in removing that complexity instead: *more with less*.

More with less also discourages things like handoffs, preparation Sprints, stabilization Sprints, so-called "dependencies", separate analysis groups, separate architects, and separate queues for each team.

In LeSS, Scrum Masters and management help the teams learn. Management shifts focus from direct command to improving the capability of the development system.

In our experience, the LeSS framework is the minimum, barely sufficient structure that product groups need to take ownership, gain a whole product view, and optimize organizations for value delivery and flexibility.

You can learn about the experimental mindset (and also over 500 experiments) that help inform the framework, guides with strong recommendations about how to apply it, and more about the 10 principles we touched on. There's also *us*, a growing community of practitioners and coaches.

We invite you to move away from life-sucking industrial-age machine organizations toward human, purposeful organizations where work can be fun.

It won't be easy, quick, or painless. But if you work at it, you can simplify your organization, increase your adaptability to changing business conditions, remove unnecessary drudgery, and unleash the potential your organization had all along.