The Building Blocks of Scale

The Autonomy 🍊 Cooperation Paradox
The Problem of Scale

In the Beginning

Success Happens
Cylinders of Excellence

✓ Scale creates Cylinders [Silos].
  ✗ Starting with Competency Silos
  ✗ Often reorganizing to Module/Service Silos
✓ We are good at solving problems inside a silo.
✓ We are not good at dealing with problems that cross silos.*

The Basic Question
How can small teams be autonomous and work together for the good of the organization?

The Challenge is Balance
Using the material and context at hand

* Roger Martin: Dean of the Rotman School of Management at the University of Toronto
The Essence of Scale

**Autonomy**
Use Everyone’s Brainpower

**Cooperation**
For the Good of the Whole

**Building Blocks**
1. Stable Squads
2. Shared Responsibility
3. Rich Objectives
4. Managers as Integrators
5. Fast Feedback
6. Double Loop Learning
1. Stable Squads

“One of the most scalable organizations in human history was the Roman army. Its defining unit: The squad — eight guys. The number of guys that could fit in a tent.”

“As a manager, your focus should really be creating these high-performing squads of people who have good chemistry, who can get things done. Then you figure out how to apply them to problems.”

1. Squad members sit together.
2. Squads have an internal leader.
3. Squads are stable over time.
4. Problems are assigned to squads.
5. Squad dependencies are minimized.

Chris Fry  SVP of Engineering at Twitter
Previously led development at Salesforce
Case Study

Jeff Bezos: “We don’t need communication, we need decentralization! If a team can’t be fed with two pizzas, it’s too large.”

2001: Amazon’s monolithic architecture hit the wall – it would not scale. Moved to a service architecture that could be sold as a product. Each service is owned by a 2-Pizza team (8-10 people) with end-to-end responsibility for the service – from contract to operations.

2006: Amazon Web Services released.

2014: Current revenue ~ $1 billion per quarter.

Conway’s Law:
Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.
2. Shared Responsibility

No Monopolies!

- Cooperation requires accommodation.
- Accommodation has a cost.
- If the cost of accommodation is born by one party, resentment arises and cooperation fails.
- Monopolies (by definition) don’t have to accommodate.
- Thus monopolies destroy cooperation.

Examples of Monopolies:

- Staff Groups: HR, IT, Process
- XP: Bill of Rights
- Scrum: One Wringable Neck
Case Study

The Tale of a Large Hotel Chain.

Goals.
Fill the Rooms.
Charge Full Price.
Keep Customers Happy.
Minimize Back office Cost.

Not Profitable Enough.

KPI's.
Incentive Pay.

No Improvement.

Sensitivity Training.
Things got worse.

From
Six Simple Rules: How to Manage Complexity without Getting Complicated by Yves Morieux and Peter Tollman (Boston Consulting Group)
Failure to Cooperate

Go and See
Understand What People Do

Fill the Rooms.
Charge Full Price.
Keep Customers Happy.

Productivity!!!

Individual and department metrics trumped providing customers with functioning rooms.

Daytime Hours
Motivation to Cooperate

Receptionists given a voice in assessing housekeeping and maintenance performance.

Housekeeping and maintenance cooperate to make sure guests are happy when they arrive in their room, and are more responsive to requests from receptionists.

Increased Customer Satisfaction
Improved Average Room Rate
Higher Occupancy Rate
Lower maintenance costs due to preventative maintenance
Receptionist retention rate improved by a factor of 6

Tripled Share Price in 2 Years

Six Simple Rules: How to Manage Complexity without Getting Complicated by Yves Morieux and Peter Tollman (Boston Consulting Group)
3. Rich Objectives

1. Unifying Goal
   Scalable, Salable
   Infrastructure

2. Our Team’s Responsibility
   Useful Service

3. What We do to Make Sure Other Teams are Successful
   Interface Contracts
5. Managers as Integrators

Integrators create constructive cooperation throughout the company.

“Being an integrator is – or at least ought to be – at the very heart of the managerial role.”\(^1\)

- Deeply understand and actively shape work context
- Make context-based decisions to resolve trade-offs and contradictions
- Rely on judgment over metrics

“Managers need to be present to observe and gather through conversation and interactions the non-measurable data…”\(^1\)

“Metrics are a conversation starter”\(^2\)

---

1. Six Simple Rules: How to manage Complexity without Getting Complicated
   By Yves Morieux and Peter Tollman

2. A Practical Approach to Large Scale Agile Development
   By Gary Gruver, Mike Young, and Pat Fulghum
Reconfigured the Development Process

1. Manage features, not projects
2. Decouple releases from development
3. Architects and specialists become consultants
   a) Architectural runway work
4. Leaders decide which features to do next
5. Engineers reassigned from functions to teams
   a) Small, multi-function feature teams
   b) Product and technical leadership
   c) Access to customer engineer
   d) Determine their own process
6. Teams Assigned Features
   a) One feature (~3 weeks) at a time
   b) Date is fixed, team determines content
   c) Feature is expected to pass release verification
   d) Figure out how to get it done – get help if necessary

The past was not good enough for the future.

What Happened:

✓ Twice the speed
✓ Higher hit rate
✓ Significantly higher quality
✓ More engaged engineers
✓ Reciprocity between teams

Thanks to Mats Lindén, Ericsson
4. Fast Feedback

Next Generation Software Development Process

- Acceptance test driven development process
- Tight collaboration between business and delivery teams
- Cross-functional teams include QA and operations
- Automated build, testing, db migration, and deployment
- Incremental development on mainline with continuous integration
- Software always production ready
- Releases tied to business needs, not operational constraints

Next Generation Product Development Process

Build a Minimum Viable Product

Measure the Response

Learn

**Continuous Delivery**

- Constant flow of new features into production
- Customer
- Delivery team

Repeat

Development team
Customer
It’s ~ 2004 in Atlanta…

We have to BUILD!
Building is HARD!
How often do we Build?
How about every day?

Time passes….

✓ Build every hour

✘ Deploy Last Good Build at End of Day in Atlanta

✓ Deploy EOD Shanghai
6. Double Loop Learning

The CEO started a lean initiative. Numerous Kaizen events were held. Lots of problems were fixed. The team leaders said: “That was easy, we already knew about those problems – we’re thrilled the lean initiative fixed them!”

What’s wrong with this picture?

Question
Assumptions
What assumptions underlie these software development concepts?

- Requirements
- Estimating
- Backlogs
- Projects
- Defect Density
- Standard Process
- Resource Efficiency
- Self-Organizing Teams
Case Study: Anonymous Delivery Organization

Large, successful product
Arduous approval process
  ✓ Little involvement from delivery teams
Obscure link between work and value
  ✓ No one responsible for delivering value
Competency managers responsible for:
  ✓ % Utilization of people in their organization
  ✓ Finish their function’s part within estimated hours
Result of counterproductive metrics:
  ✓ Workers assigned to multiple projects
    ⇒ Teams delayed waiting for people
  ✓ Pressure to meet function-specific estimate
    ⇒ Passable work from each function
    ⇒ Huge integration problem at the end

Insanity: Repeating the same thing … and expecting different results.
The Building Blocks of Scale

Use Everyone’s Brainpower

For the Good of the Whole

Start with Immersion
Understand how the work works

1. Stable Squads
Strong teams ↔ Tough problems

2. Shared Responsibility
No ☞ Monopolies

3. Rich Objectives
Overall Goal + Our Part + Interaction

4. Managers as Integrators
Create Constructive ☝️ Cooperation

5. Fast Feedback
Rapid {end-to-end} learning

6. Double Loop Learning
Question ♦ Assumptions
Thank You!

More Information:  www.poppendieck.com
Why aren’t we predictable?

Understand how the work works.

Accept uncertainty and learn how to live with it.

No more point estimates!
All estimates must be best-case / worst-case.

Manage features, not projects.
Fix scope, let deadline vary.

Thanks to Hendrik Esser, Ericsson

On Time Deliveries
Superior Quality
Case Study

High End Infotainment Systems
2007: 70% Automotive Market Share
⇒ $2b of $3b in sales
BUT: How do we grow?

2008: Commissioned Saras
✓ Headed by Sachin Lawande
✓ Aimed at emerging markets
✓ ½ the price; ⅓ the cost
✓ 1 year to market

So the product had to be:
⇒ Simple, modular, inexpensive
⇒ Standard technologies
⇒ Third party solutions
⇒ No after-sale customization

Software group in India
✓ Led by Arvin Baalu*

Hardware group in China
✓ Led by Kelei Shen*

* leaders with established credibility at Harman

Reported to Sachin Lawande
✓ Rather than automotive head

Competency managers coordinated the contributions of their group.

Organized around major subsystems
✓ Eg. Navigation system
✓ Radical shift from company tradition

Rest of company skeptical
✓ Engineers expected inferior product
✓ Sales reps had to be ordered to sell it
✓ First sale to Toyota changed everything

2009 Launch. $3b in profitable sales in 18 months.

Vijay Govindarajan – Reverse Innovation