Not Doing SAFe, No Problem
Not Doing These, Big Problem
What You Must do at Scale

Al Shalloway, CEO
Net Objectives

Al Shalloway
CEO, Founder of Net Objectives
alshall@NetObjectives.com
@AlShalloway

Co-founder of Lean-Systems Society
Co-founder Lean-Kanban University (no longer affiliated)
Contributor to SAFe
Purpose of This Tutorial

*Be clear, this is not a promotion of SAFe. It is an exposition of what we’ve learned from SAFe.*

- Be aware of issues you must solve at scale
- Understand how SAFe manages these issues
- Understand alternatives to SAFe’s solutions
- Understand that all methods are “point solutions” to what’s possible

*Learn why you should first become clear on your mindset, then pick an approach that’s consistent with your mindset.*
What Tutorial Covers

- What we must do
  - ...
- Steps to Agile at Scale

What You Must Do

**Use Lean-Thinking**

- Have an owner for the development value stream
- Organize the portfolio, program, team eco-system
- Solve the big picture first
- Have an explicit workflow and an “all-defined, all-in”
- Use Acceptance Test Driven Development (ATDD)
- Build iteratively for feedback, deliver incrementally for value
- Allocate capacity for architecture and tech debt
- Have management provide leadership
Eliminate Waste

Communication overhead as more people are added is not as easy as one might think.

Common Organizational Structure

Inspired by Dan North, BSC/ADP 2012
Hierarchical

What they can manage

Their people
- If they are working on the right things and at the right level
- Their “productivity”

The quality of work of their people

The Nature of Our Work

Marketing  Product Management  Development  Support
We Manage This Way

even though our value flows this way

Hierarchical vs. Lean Management

<table>
<thead>
<tr>
<th>What they can manage</th>
<th>What they need to manage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Their people</td>
<td>Time-to-market</td>
</tr>
<tr>
<td>• If they are working</td>
<td>How upstream and</td>
</tr>
<tr>
<td>on the right things</td>
<td>downstream groups affect</td>
</tr>
<tr>
<td>• Their “productivity”</td>
<td>them</td>
</tr>
<tr>
<td>The quality of work</td>
<td>How they affect upstream</td>
</tr>
<tr>
<td>of their people</td>
<td>and downstream groups</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
here’s a spot!
...and another!

Who is managing the value?
how often does work wait?

What percent of the time is our work moving forward?
How much of the time is it waiting for something else to be done?
How would you know?
No one is managing this in most companies.
GROUP DISCUSSION

What happens when adding value is delayed?

- Between getting requirements and using them?
- Between writing a bug and it being detected?
- Between two groups getting out of sync?
- Just waiting for someone?

The Whole Picture
What You Must Do

Use Lean-Thinking

- **Have an owner for the development value stream**
- Organize the portfolio, program, team eco-system
- Solve the big picture first
- Have an explicit workflow and an “all-defined, all-in”
- Use Acceptance Test Driven Development (ATDD)
- Build iteratively for feedback, deliver incrementally for value
- Allocate capacity for architecture and tech debt
- Have management provide leadership
First Order Solution: Pilot Scrum On a Few Teams

Leaves Several Problems

- How do we manage competing priorities?
- How do we feed the teams?
- How do we handle dependencies?
- How do we handle product portfolio management?
- What do we do with those that don’t quite fit the model?
- How do we best integrate?

Scrum solves intra-team dynamics, not inter-team dynamics

If we have quality issues or lack of automated testing, things will be harder to solve
Lean-Agile / SAFe Can Help Solve These

- How do we handle dependencies?
- How do we feed the teams?
- How do we handle product portfolio management?
- What do we do with those that don't quite fit the model?

Educate business stakeholders in Lean Product management

协调计划中的依赖关系

如何整合?

改善质量通过ATDD、模式、自动化测试和持续集成
What Many Development Processes Look Like

Component Teams Are Better
Feature Teams Are Best (align strategy & delivery)

Why don’t we just start with cross-functional teams?
Why don’t we just start with cross-functional teams?

Do it if you can. Often have no real idea how.
Lean-Agile / SAFe

**SAFe**
Portfolio to guide what to do
Program to structure teams
Team level for efficient work

All in, all-defined, all the way

Organized for business value delivery

Pragmatic
Inflection Points

Decisions to be made – but many of the answers are really multiple-choice, i.e.:

How should we organize development groups for a focus on finishing?

- SAFe Trains
- Teams with shared backlogs
- Dynamic Feature Teams

*While it is true that there is no best practice that fits all situations, there are typically 2-4 that will cover 80-90% of the situations.*
“Inflection Points” are decision points about defining your practices, eco-system and workflow. SAFe can be thought of as a particular answer to the 40 inflection points we’ve found in large scale Agile. Understanding this enables SAFe to be adopted as is, tailored, and/or extended.
**Team Level Process Inflection Points**

**Input → Priority → Planning**

**Epics → MBIs**

**BUSINESS STAKEHOLDERS**
IDENTIFY, SIZE AND SEQUENCE WORK

**Enough business value?**

**Enough ROI?**

**Technically feasible?**

**Quarterly**

**Staging → Ready to Pull → Prioritize Across the Teams**

**MBIs & Features**

**Enough capacity?**

**Biweekly to Quarterly**

**Align Teams with Shared Backlogs**

**Stories & Tasks**

**Prepare**

**Scrum Kanban**

**XP Leanban**

**Kanban (flow)**

**Should we estimate our work?**

**What estimation method should we use?**

**What You Must Do**

**Use Lean-Thinking**

Have an owner for the development value stream

Organize the portfolio, program, team eco-system

**Solve the big picture first**

Have an explicit workflow and an “all-defined, all-in”

Use Acceptance Test Driven Development (ATDD)

Build iteratively for feedback, deliver incrementally for value

Allocate capacity for architecture and tech debt

Have management provide leadership
Core Inflection Points – Initial Focus

**INPUT** → **PRIORITY** → **PLANNING** → **BUSINESS STAKEHOLDERS**

**Epics** → **MBIs** → **IDENTIFY, SIZE AND SEQUENCE WORK**

- How will our intake process be controlled?
- How does our planning cycle look?
- How will tactical work be handled?
- What should our planning cycle be?
- How do we align our people/teams to the work to be done?

**STAGING → READY TO PULL → PRIORITIZE ACROSS THE TEAMS**

**MBIs & Features** → **enough capacity?**

**Q UARTERLY**

**STORIES & TASKS**

**AGILE RELEASE** → **deliver on demand**

- How will we express, decompose, and validate requirements?
- What should our team process be?
- Do we have so much technical debt that we need to pay some down before proceeding?
- To what extent will we automate our tests?

Core Inflection Points - Secondary
What You Must Do

Use Lean-Thinking
Have an owner for the development value stream
Organize the portfolio, program, team eco-system
Solve the big picture first

**Have an explicit workflow and an “all-defined, all-in”**

Use Acceptance Test Driven Development (ATDD)
Build iteratively for feedback, deliver incrementally for value
Allocate capacity for architecture and tech debt
Have management provide leadership
Miami Lean-Kanban 2009

Very clear teams with explicit workflow learned how to work together better significantly faster than when they didn’t.

Teams adopting Scrumban and stopping there (i.e., retaining iterations) perform better than before.

*Agile is about collaboration, why would we not have an explicit workflow?*

*All-in, all defined is an extension of this.*

*But doesn’t mean all the way is correct.*

---

What You Must Do

Use Lean-Thinking
Have an owner for the development value stream
Organize the portfolio, program, team eco-system
Solve the big picture first
Have an explicit workflow and an “all-defined, all-in”

**Use Acceptance Test Driven Development (ATDD)**
Build iteratively for feedback, deliver incrementally for value
Allocate capacity for architecture and tech debt
Have management provide leadership
Customer: I need new features *yesterday*

Devs hear: Get it done; *Fast, at all costs!*

code base: Sloppy changes

code base: Increased defects

code base: Increased complexity

Testers are overloaded

Testers w/ more work
Systems w/ more defects

Devs don’t get immediate feedback

Devs create more defects

Further delays in feedback

Testing occurs long after coding
Finding defects is waste.

Preventing defects is essential
TABLE WORK

About Poor Quality

What causes poor quality?
What is the cost of poor quality?
Example: Acceptance TDD

Basic Employee Compensation
Each week, hourly employees are paid
- A standard wage per hour for the first 40 hours worked
- 1.5 times their wage for each hour after the first 40 hours
- 2 times their wage for each hour worked on Sundays and holidays

Payroll.Fixtures.WeeklyCompensation

<table>
<thead>
<tr>
<th>Standard Hours</th>
<th>Hol/Sun Hours</th>
<th>Wage</th>
<th>Pay()</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>8</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

These two are clear

Basic Employee Compensation
Each week, hourly employees are paid
- A standard wage per hour for the first 40 hours worked
- 1.5 times their wage for each hour after the first 40 hours
- 2 times their wage for each hour worked on Sundays and holidays

Payroll.Fixtures.WeeklyCompensation

<table>
<thead>
<tr>
<th>Standard Hours</th>
<th>Hol/Sun Hours</th>
<th>Wage</th>
<th>Pay()</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0</td>
<td>20</td>
<td>$800</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>20</td>
<td>$950</td>
</tr>
<tr>
<td>48</td>
<td>8</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
But what about this?

<table>
<thead>
<tr>
<th>Standard Hours</th>
<th>Hol/Sun Hours</th>
<th>Wage</th>
<th>Pay()</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0</td>
<td>20</td>
<td>$800</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>20</td>
<td>$950</td>
</tr>
<tr>
<td>48</td>
<td>8</td>
<td>20</td>
<td>$1520</td>
</tr>
</tbody>
</table>

Example: Acceptance TDD

<table>
<thead>
<tr>
<th>Standard Hours</th>
<th>Hol/Sun Hours</th>
<th>Wage</th>
<th>Pay()</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0</td>
<td>20</td>
<td>$800</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>20</td>
<td>$950</td>
</tr>
<tr>
<td>48</td>
<td>8</td>
<td>20</td>
<td>$1520</td>
</tr>
</tbody>
</table>
What You Must Do

Use Lean-Thinking
Have an owner for the development value stream
Organize the portfolio, program, team eco-system
Solve the big picture first
Have an explicit workflow and an “all-defined, all-in”
Use Acceptance Test Driven Development (ATDD)
**Build iteratively for feedback, deliver incrementally for value**
Allocate capacity for architecture and tech debt
Have management provide leadership

Iterative Business Slices

**Forecast-driven vs. Feature-driven**

Iterative = Start with what we know, implement some, show the customer, adjust
Incremental = Feature Driven ... Value in short cycles ... Slices vs. Layers

**BIG Batch**

UX
Client
Server
DB schema

**Incremental**

UX
Client
Server
DB schema

MORE Business VALUE ... OR DONE
What You Must Do

Use Lean-Thinking
Have an owner for the development value stream
Organize the portfolio, program, team eco-system
Solve the big picture first
Have an explicit workflow and an “all-defined, all-in”
Use Acceptance Test Driven Development (ATDD)
Build iteratively for feedback, deliver incrementally for value

Allocate capacity for architecture and tech debt
Have management provide leadership

Common Debate

Product Manager
If we let the devs have their way, they’ll design this thing until hell freezes over, and then we’ll all be out of a job

Architect
If we let the business folks have their way they’ll never let us fix this thing until hell freezes over, and then we’ll all be out of a job

Dean Leffingwell
What You Must Do

Use Lean-Thinking
Have an owner for the development value stream
Organize the portfolio, program, team eco-system
Solve the big picture first
Have an explicit workflow and an “all-defined, all-in”
Use Acceptance Test Driven Development (ATDD)
Build iteratively for feedback, deliver incrementally for value
Allocate capacity for architecture and tech debt

**Have management provide leadership**

---

Should a company target its culture in its efforts to transform its production processes and all the [roles] associated with it? It is tempting to answer: Yes! But that would be a mistake.

Culture is no more likely a target than the air we breathe. It is not something to target for change. Culture is an idea arising from experience.

Our idea of the culture of a place or organization is a result of what we experience there. In this way, a company’s culture is a *result* of its management system. ...

Culture is critical, and to change it, you have to change your management system.

Focus on the management system, on targets you can see: leaders’ behavior, specific expectations, tools, routine practices.

Lean production systems make this easier, because they emphasize explicitly defined processes and use *visual controls*.

---

**Should a company target its culture in its efforts to transform its production processes and all the [roles] associated with it? It is tempting to answer: Yes! But that would be a mistake.**

**Culture is no more likely a target than the air we breathe. It is not something to target for change. Culture is an idea arising from experience.**

**Our idea of the culture of a place or organization is a result of what we experience there. In this way, a company’s culture is a *result* of its management system. ...**

Culture is critical, and to change it, you have to change your management system.

Focus on the management system, on targets you can see: leaders’ behavior, specific expectations, tools, routine practices.

Lean production systems make this easier, because they emphasize explicitly defined processes and use *visual controls*.

---

**Should a company target its culture in its efforts to transform its production processes and all the [roles] associated with it? It is tempting to answer: Yes! But that would be a mistake.**

**Culture is no more likely a target than the air we breathe. It is not something to target for change. Culture is an idea arising from experience.**

**Our idea of the culture of a place or organization is a result of what we experience there. In this way, a company’s culture is a *result* of its management system. ...**

Culture is critical, and to change it, you have to change your management system.

Focus on the management system, on targets you can see: leaders’ behavior, specific expectations, tools, routine practices.

Lean production systems make this easier, because they emphasize explicitly defined processes and use *visual controls*.

---
Resistance is not to change.

“In practice, all systems do insist on exercising their own creativity. They never accept imposed solutions, pre-determined designs, or well-articulated plans that have been generated somewhere else.

Too often, we interpret their refusal as resistance. We say that people innately resist change.

But the resistance we experience from others is not to change itself. It is to the particular process of change that believes in imposition rather than creation. It is the resistance of a living system to being treated as a non-living thing.

It is an assertion of the system’s right to create. It is life insisting on its primary responsibility to create itself.”

People do not resist change, per se. People resist loss.

You appear dangerous to people when you question their values, beliefs, or habits of a lifetime.”
What You Must Do

Use Lean-Thinking
Have an owner for the development value stream
Organize the portfolio, program, team eco-system
Solve the big picture first
Have an explicit workflow and an “all-defined, all-in”
Use Acceptance Test Driven Development (ATDD)
Build iteratively for feedback, deliver incrementally for value
Allocate capacity for architecture and tech debt
Have management provide leadership

Everything must be made as simple as possible but no simpler. Einstein

While prescribing things that aren’t necessary is bad, not looking at things that are is simplistic.

Step one: Pick your mindset

Absorb what is useful, reject what is useless, add what is specifically your own. Bruce Lee
People Do Want an Answer, But Only for So Long

Novice
Advanced
Beginner
Competent
Proficient
Expert

Show me what to look at
Show me what to do
Provide me choices for where I am
Provide me a roadmap
Let me create my own roadmap

Telling people what to do without telling them how usually leads to frustration and a doubting of the approach.

I would suggest we have no idea how complex our problem is if we make people re-invent the wheel while we are having them learn how to learn.

Lean-Thinking Mindset

Attend to the System
Eliminate delays to eliminate waste
Use explicit workflows
Attend to quality – do not pass on errors
Management is important
Other Mindsets

It’s all Scrum
Grow from the team level up
Use evolutionary change, consider your eco-system orthogonal to your transition
Just jump to where you need to be

Step two: Decide on your approach

Absorb what is useful, reject what is useless, add what is specifically your own. Bruce Lee
Net Objectives Pattern Repository
http://www.netobjectivestest.com/PatternRepository
Register for newsletter and access to resources:
http://www.netobjectives.com/register

Net Objectives SPC Training in Seattle, Oct 27-30
www.netobjectives.com/events