Six Steps Towards Self-Learning Teams and Organizations

Andy Cleff, CSP, ICP-ACC, ICP-ATF, CSM, CSPO
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
We learn rapidly by experimenting frequently. We make our experiments “safe to fail” so we are not afraid to conduct more experiments.
What is the scrum master’s or agile coach’s role in all of this?
Overview: Six Steps

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible
Conditions for growth

In too many organizations too many people are doing a second job no one is paying them for...
Conditions for growth

Personal evolution, human flourishing, becoming a better person — these are at the center of the culture of a deliberately developmental organization (DDO)

agileuprising.librsyn.com
Step One: Build an Inventory
Skill inventory

Prompt:
• What do you already know?
• What do you want to learn?
• What skills/competencies do you think the team needs to be most successful, now and in the foreseeable future…
Skill inventory

Example Categories:
• Languages & Technologies
• Processes & Practices
• Soft Skills
### Skills Inventory

<table>
<thead>
<tr>
<th>List of Tools / Languages / Technologies / Platform Specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android AutoValue</td>
</tr>
<tr>
<td>Android &amp; Claims</td>
</tr>
<tr>
<td>Android In-app billing</td>
</tr>
<tr>
<td>Android CI/CD - Circle CI</td>
</tr>
<tr>
<td>Android Preguard</td>
</tr>
<tr>
<td>Android lint</td>
</tr>
<tr>
<td>Android Chromecast Receiver</td>
</tr>
<tr>
<td>Android Chromecast Sender</td>
</tr>
<tr>
<td>Android multi-windows</td>
</tr>
<tr>
<td>Android Facebook integration</td>
</tr>
<tr>
<td>Android Banner Ads - CCP</td>
</tr>
<tr>
<td>Android Video Ads - Freewheel</td>
</tr>
<tr>
<td>Android Widevine - Streaming &amp; Offline Viewing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Processes / Practices / Platform Agnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV testing</td>
</tr>
<tr>
<td>long-term strategy</td>
</tr>
<tr>
<td>understand growth KPIs</td>
</tr>
<tr>
<td>accounting</td>
</tr>
<tr>
<td>code review</td>
</tr>
<tr>
<td>debugging</td>
</tr>
<tr>
<td>learning to learn</td>
</tr>
<tr>
<td>abstraction</td>
</tr>
<tr>
<td>identity and access management</td>
</tr>
<tr>
<td>caps</td>
</tr>
<tr>
<td>refactoring</td>
</tr>
<tr>
<td>continuous integration</td>
</tr>
<tr>
<td>continuous deployment</td>
</tr>
<tr>
<td>code profiling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of Soft Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>self-management</td>
</tr>
<tr>
<td>speaking</td>
</tr>
<tr>
<td>adaptability</td>
</tr>
<tr>
<td>researching</td>
</tr>
<tr>
<td>communication</td>
</tr>
<tr>
<td>networking (people)</td>
</tr>
<tr>
<td>interviewing</td>
</tr>
<tr>
<td>facilitation</td>
</tr>
<tr>
<td>coaching</td>
</tr>
<tr>
<td>improve</td>
</tr>
<tr>
<td>giving and receiving feedback</td>
</tr>
<tr>
<td>writing for publication</td>
</tr>
<tr>
<td>conflict resolution</td>
</tr>
<tr>
<td>collaboration</td>
</tr>
</tbody>
</table>
Checkpoint

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible

Is anything confusing so far?

What are you doing already that's working?

What's not working for you?
Step Two: Build a Team Level Matrix

<table>
<thead>
<tr>
<th>Technical/Platform Specific</th>
<th>Apprentice</th>
<th>Jour</th>
<th>（Use... dabbled there once or twice）</th>
</tr>
</thead>
<tbody>
<tr>
<td>qsql</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alacrite</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>android (Java)</td>
<td>N</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>angular.js</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autolayout (iOS)</td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>AWS</td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Backbone.js</td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>Brightscript</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cassandra</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>charles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chef</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>projcore</td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>consul</td>
<td></td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>cucumber</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: “N” = new, “F” = future
Team matrix of competencies

Simple Model of Competency

• Apprentice
• Journeyperson
• Master
Team matrix of competencies

Dreyfus Model of Skills Development

- Novice
- Advanced beginner
- Competent
- Proficient
- Expert
Team matrix of competencies
<table>
<thead>
<tr>
<th>Technical/Platform Specific</th>
<th>Apprentice (Um... dabbled there once or twice)</th>
<th>Journeyperson (Yeah, I can do it.)</th>
<th>Semi-Master (Heck, I can teach it)</th>
</tr>
</thead>
<tbody>
<tr>
<td>.net</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ajax</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algolfire</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>android (Java)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>angularJS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Autolayout (iOS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackboneJS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brightscript</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c#</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cassandra</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>charles</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>chef</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>composer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cucumber</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>consul</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cucumber</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: "N" = now, "F" = future, leave blank if does not apply
Dealing with fear
Building safety

• What I learned over the last ___ weeks, and how it relates to the team’s mission

• What I want to learn over the next __ to __ weeks, related to the team’s mission

• What I need to start learning that thing
Checkpoint

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible

Is anything confusing so far?
What are you doing already that's working?
What's not working for you?
Step Three: Learning Backlog + Slack & Flow
"Fool, don’t you see I’m busy? I don’t have time to sharpen the saw"
Checkpoint

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible

Is anything confusing so far?
What are you doing already that's working?
What's not working for you?
Step Four: Get Your Guilds Going
Share knowledge, skills, and capabilities

- Special Interest Groups (SIGs)
- Centers of Excellence (CoE)
- Communities of Practice (CoP)
- Birds of a Feather (BoF)
- Tech Clubs
- Guilds
- Tribes
- ! = Committee
Communities of practice

• **A domain of knowledge**, which defines a set of issues; creates common ground and a sense of common identity

• **A community of people** who care about this domain; creates the social fabric of learning; fosters interactions, relationships, hopefully mutual respect and trust; willingness to share ideas, expose one’s ignorance, ask difficult questions, and listen carefully.

• **A shared practice** that they are developing to be effective in their domain; a set of frameworks, ideas, tools, information, systems, language, stories, and documents/knowledge bases.
Community design possibilities

- **Bootlegged**: Only visible to a circle of people “in the know”

- **Legitimized**: Officially sanctioned as a valuable entity

- **Supported**: Provided with direct resources from the organization

- **Institutionalized**: Given an official status and function in the organization
“When people work this way, barriers and boundaries between people and what they do are often insubstantial or irrelevant, since a collective endeavor holds people together.”

Seely Brown, *The Interaction of Complexity and Management*
Outcomes of CoP’s

• Sharing of Information, insights, problem solving
• Pondering of common issues, exploration of ideas, acting as sounding boards
• Creation of tools, standards, design patterns
• Hosting knowledge-based activities, like lessons learned, Q&A sessions, and on-boarding newcomers
Checkpoint

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible

Is anything confusing so far?
What are you doing already that's working?
What's not working for you?
Step Five: Measure
“…Analysis without numbers is only an opinion.”

Atkins Law #1

“…If all we have are opinions, let’s go with mine.”

Jim Barksdale
Measure many things

Process Health

Release

Product Development

Technical / Code

People / Teams
Process health metrics

Assess day-to-day delivery team activities and evaluate process changes:

- Cycle time
- Cumulative flow diagrams
- Control charts
- Number of experiments performed
- Number of improvements you’ve made to your process over time.
- Flow efficiency
- Story/epic lead time
- Successful iteration completion
- Escaped defect resolution time
- Time thievery
The five thieves of time

- Too much WIP
- Unknown dependencies
- Unplanned work
- Conflicted priorities
- Neglected work
Release metrics

Focus on identifying impediments to continuous delivery:

- Time since last release
- Release success rate
- Release time
- Cost per release
- Escaped defects
- Release net promoter score
- Release adoption / install rate
Product development metrics

Help measure alignment of product features to user needs:

- Customer value delivered
- Risk burn down
- Value stream mapping
- Sales velocity
- Product net promoter score (NPS)
- User analytics / DAU / MAU
- Backlog health index

- Number of validated business-level hypotheses developed
- Number of times a week you talk to an actual customer
- Ratio of implemented to non-implemented customer-driven changes
Technical & code metrics

Help determine quality of implementation and architecture:

- Automated test coverage
- Number of tests written before coding
- Unit / regression test coverage
- Build time
- Defect density
- Code churn
- Code ownership

- Code complexity
- Coding standards adherence
- Crash rate
- Build breaks
- Technical drag
- Ratio of fixing work vs feature work
People & teams’ health and wellbeing

- Gallup Q12
- Mood App
- TeamMood
- eNPS
- Comparative Agility Survey
- Crisp Happiness Index
- Team Barometer

- Spotify Health Check
- Lean Agile Intelligence
- Team Morale Metric
- H.I.P. Survey
- OfficeVibe
- 15Five
- Happiness Wall
Checkpoint

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible

Is anything confusing so far?
What are you doing already that's working?
What's not working for you?
Step Six: Make Everything Visible
Visibility and transparency

• Build momentum
• Drive interest
• Remove constraints
• Enable collaboration across boundaries
• Share Knowledge
• Lean coffee
• Celebration grid
Success and Failure

It’s not about the first-mover advantage; It’s about the fast-learner advantage.

Don’t change things; run experiments.

Create a safe-to-fail environment.

CELEBRATION GRID

MISTAKES
- You lucky bastard!
- WTF, dude! You screwed up! Where’s your brain?

EXPERIMENTS
- Yay! You exceeded AND you learned!
- Ok, you failed BUT you learned!
- Argh, bad luck!

PRACTICES
- Yay! You exceeded by doing the right things!

LEARNING
No learning

SUCCESS

Don’t change things; run experiments.
Run more experiments, faster, and cheaper.
Recognize and celebrate Etsy
Recognize and celebrate
Recognize and celebrate
Recognize and celebrate
“Simple, public recognition is one of the most effective and most underutilized management tools.”

Laszlo Bock
Checkpoint & recap

1. Build an Inventory
2. Construct a Team Level Matrix
3. Populate a Learning Backlog
4. Get Your Guilds Going
5. Measure Stuff
6. Make Everything Visible

Is anything confusing?
What are you doing already that's working?
What's not working?
Better Me + Better You = Better Us

• Creativity and learning takes a mix of disciplines

• Free flowing collision of ideas through conversations and connections makes a difference

• Exchange of ideas and technologies speeds innovation
Let’s keep the conversation going…

Andy Cleff

andycleff@icloud.com
andycleff.com
linkedin.com/in/andycleff
@JustSitThere
coalition.agileuprising.com