Agile in the Large

Nathaniel T. Schutta
@ntschutta

We’ve come a long way...

Remember when
agile was extreme?

Even RUP was agile...

Agile has evolved.

Not surprising really.

We've gone from iterative to XP to Scrum to Kanban.

Many variations and combinations.

Even “agilefall”.

End of the day, agile works.

It matches the real world.

Is there anything you do that isn't agile in nature?

Fundamentally, agile understands one key:

Can't stop change,
adapt to it.

Most agile examples deal with small teams.

Small teams can accomplish a lot!

But certain projects require bigger teams.

We won’t fit in one room...

What do we do about that?

Scaling up.

A scrum of scrums isn't a panacea!

Harder to get everyone on the same page.

But it can be done.

Scaled Agile Framework.

http://scaledagileframework.com

At the highest level...

Investment themes.

How are we spending the money?

Another way to think of it:
what do we value?

Changes how we think about budgets...

Which can be a major stumbling block.

Many entrenched interests.

It *should* be an improvement!

“Don’t ask for $20 million, ask for $2.”

Themes shape epics.

Business and architectural.

Some large scale initiative.

Customer facing, usually cross cutting.

“An account executive should be able to…”

Span iterations,
maybe releases.

Architectural epics.

Make sure we can support the business epics.

Simplification, cost reduction, tech modernization.

End of the day, epics drive the teams.

Epics are organized into releases.

Releases span iterations.

Individual teams work at the iteration level.

Their work is informed by the epics for the release.

Goal is simple:

Teams should be working on highest priority stories.

Teams work just like a normal agile project.

Each team has a backlog.

They groom, prioritize, task out stories, execute.

But their work rolls up to the epic(s) for the release.

Should be able to trace a given team's tasks back.

And drill down from the epics to the teams.

Speaking of teams...

They should be as independent as possible.

Teams may divide work at abstraction layers.

Break up by architectural layer: UI, services, etc.

Define the contract, teams work on their side.

May want teams to own front to back.

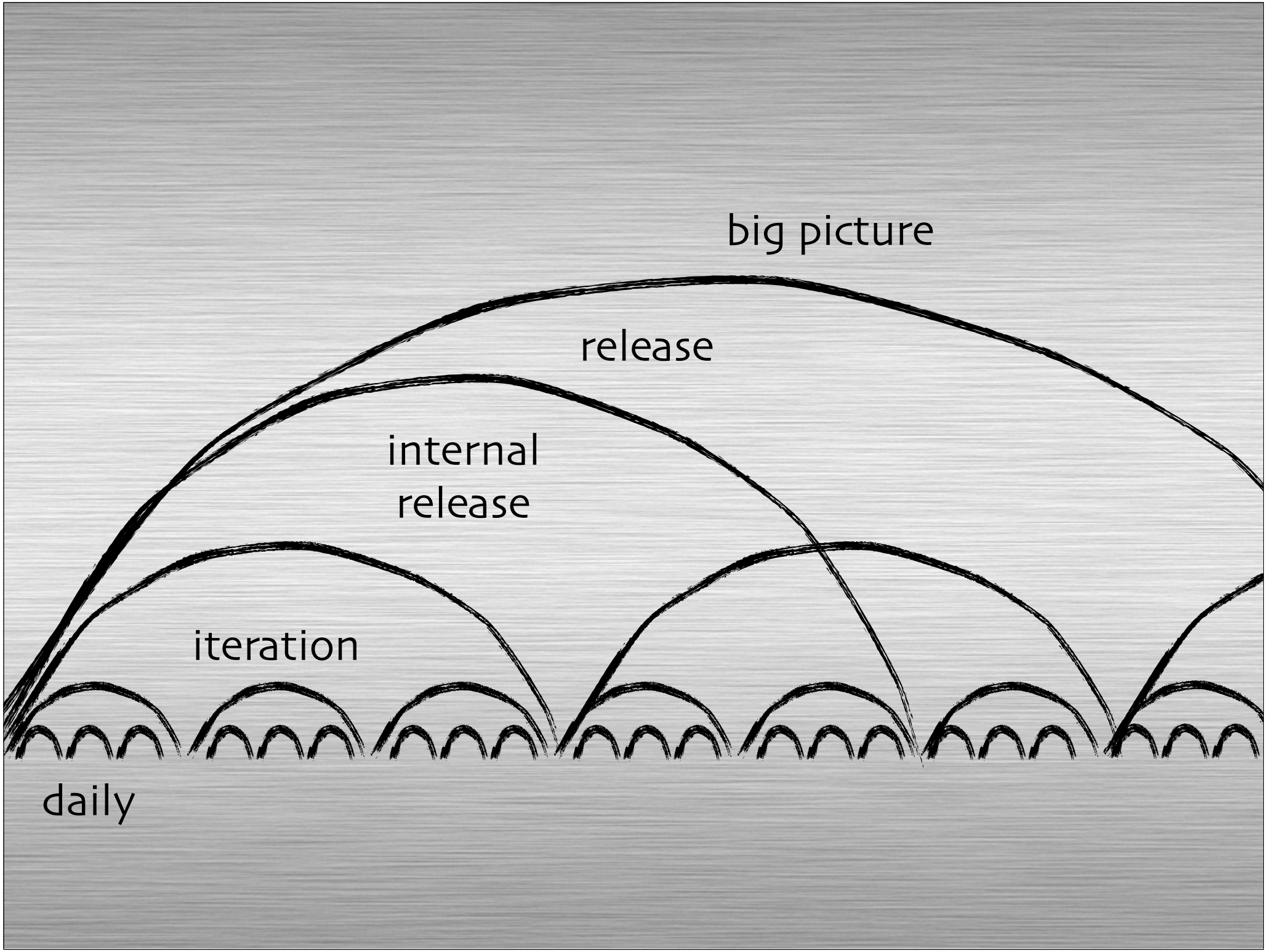
Requires proper skill sets.

Still want smallish teams - some use the ~7 metric.

Teams should work at the same cadence.

Iteration start/end, same demo day, planning etc.

To the teams, should seem like a “typical” agile project.


big picture
release
internal
release
iteration
daily

Bottom up...top down.

Architectural runway.

Architecture evolves.

Want to be just out ahead of what teams need.

Blow up technical roadblocks.

“We’ve never used X before.”

What do you need?

Licenses? Dev environment?

Teams shouldn't be sitting around waiting...

Tech foundation.

Prove it out.

Enough infrastructure to support what's next.

You will refactor.

May want to include a technical debt iteration...

System should be runnable!

Architectural spikes aren't endless!

Time box them.

Be sure you need it.

Focus on working code, not diagrams.

Will drive work down to teams.

Be careful...easy to get speculative.

Release Planning.

VPs don’t want to work at the iteration level.

They want the big picture.

Let them work at the release level.

A set of iterations (~5) equals a release.

Teams continue to work at their preferred scale...

But the higher ups can focus on the forest.

There should be a product owner at that level.

Defining the priorities for the release.

They work at the
“epic” or “saga” level.

Those priorities drive what gets done in the iterations.

Managing this is more involved!

Product council.

Program PO, team POs, other stakeholders.

Meet on a regular basis.

Groom the program backlog.

May even be one level up from that: portfolio.

Rinse repeat.

May also need regular contact for the PMs.

POs keep the teams informed.

Non-agile groups.

Nearly 100% chance you'll work with non agile groups.

You *might* be able to change them.

Don’t count on it.

They may talk the talk...

But walking the
walk is much harder.

They may have incentives for status quo.

Or they just flat can't change.

Shared resource pool.

Budget model.

Fear.

You'll have to work through it.

Identify it as a risk.

Keep leadership in the loop.

Find abstraction points.

Agree on the contract.

Or at least a first pass.

Stub it out...

Focus on your side
of the contract.

They can work at their preferred cadence.

Don’t let them go dark…

Include outside areas
as best you can.

Can a representative come to the standup?

What about planning, retrospectives, demos?

Be flexible.

Be patient.

Friction.

Adding people adds friction.

Usually means multiple rooms across sites.

Not a “bad” thing, just need to be aware.

Remote workers
normal these days.

http://blog.stackoverflow.com/2013/02/
why-we-still-believe-in-working-remotely/

Paper hanging on a wall might not work.

May need more digital radiators.

Wikis, SharePoint, etc.

Communication is harder.

Webcams, VC...
awfully cheap.

Seeing someone can make a huge difference.

Respect the people on the phone...

Circle back frequently.

Speaking of phones...

Make sure every team has a conference line.

Don’t just use Bob’s.

You'll probably need it on and off the entire day.

Be sure to have a live meeting setup too.

Publicize them!

Do you need
better phones?

More formal communication channels.

Meeting notes, iteration themes, etc.

Hate to say it, but:

May need more meetings...

Regular cross team collaboration.

At the PM and tech level.

Start with fewer...
add as needed.

Leverage the wiki.

Or whatever works!

Be sure people
know about them!

Pitfalls.

“We don’t have any project rooms.”

Space is often at a premium.

Even if it isn’t.

OK.

Build some.

You'll be amazed at what happens when you ask!

Book a conference room.

For a year.

See what happens.

Separation between meeting/work space.

What's the project budget?

How much does
a room cost?

Be patient...

“I’m new to agile.”

People fear change.

Help them see what's in it for them.

Be flexible.

How about a half day in the project room?

“Our group can’t work that way.”

Why not?

What can we do to work together?

Figure out what they need,
how can you provide it.

“We need a status report.”

PUSH data to the stakeholders.

Don’t wait for them to ask.

Fill the vacuum.

Show, don’t tell.

Demo days.

Working software buys a lot of credibility.

Show them what you did,
talk about what is next.

You *may* need to fill in some spreadsheets.

But show them
your radiators.

Them might like your approach better.

They can get *more* data,
that resonates.

“We’re the account team.”

Must break down silos.

Teams need to be flexible.

And working on the highest priority stories.

Even if it has nothing to do with accounts...

Teams needs
flexible skill sets.

May need to
rebalance, train, hire.

Be open and transparent.

Tell people what you're doing and why.

“I need two FTEs.”

People are people,
not resources.

Treat them like people.

Teams are more than people in the same room.

Managers want interchangeable parts.

That doesn’t motivate.

Things can actually be fun...

Are there ways you can make it more attractive?

Better chairs? Better equipment? Food?

Why should people join *your* project?

Developers will vote with their feet...

If people are leaving or won't join, why?

Communicate!

Retrospectives are huge.

Do them.

Adapt!

Same thing on the retro this iteration?

Fix it already!

Consider a survey for the demo day participants.

Should be *short*.

But it can generate useful intel.

Resistance.

You will meet resistance.

People hate change.

"[T]he innovator has for enemies all those who have done well under the old conditions, and lukewarm defenders in those who may do well under the new."

— Niccolò Machiavelli

Reputations may be at stake.

There will be FUD.

Be prepared for it.

What will they attack?

What is your counter?

Keep a pain register.

How much time/effort/etc?

A few weeks, a few months...what is the cost?

Harder to argue data.

May have to do it “the wrong way” first.

What are the opportunity costs?

What are your competitors doing?

They won't tell you why they're beating you.

Listen.

What are the concerns?

Some legitimate issues.

“Shared resource pool”.

Don’t just dismiss them!

Address them.

Have a conversation!

This won’t work...

How are you communicating?

With respect?

With derision?

What works better?

Change is tricky.

Habits are hard to break.

New Years
resolutions anyone?

Two approaches...

We can’t force change.

But we can influence.

Keep in mind:

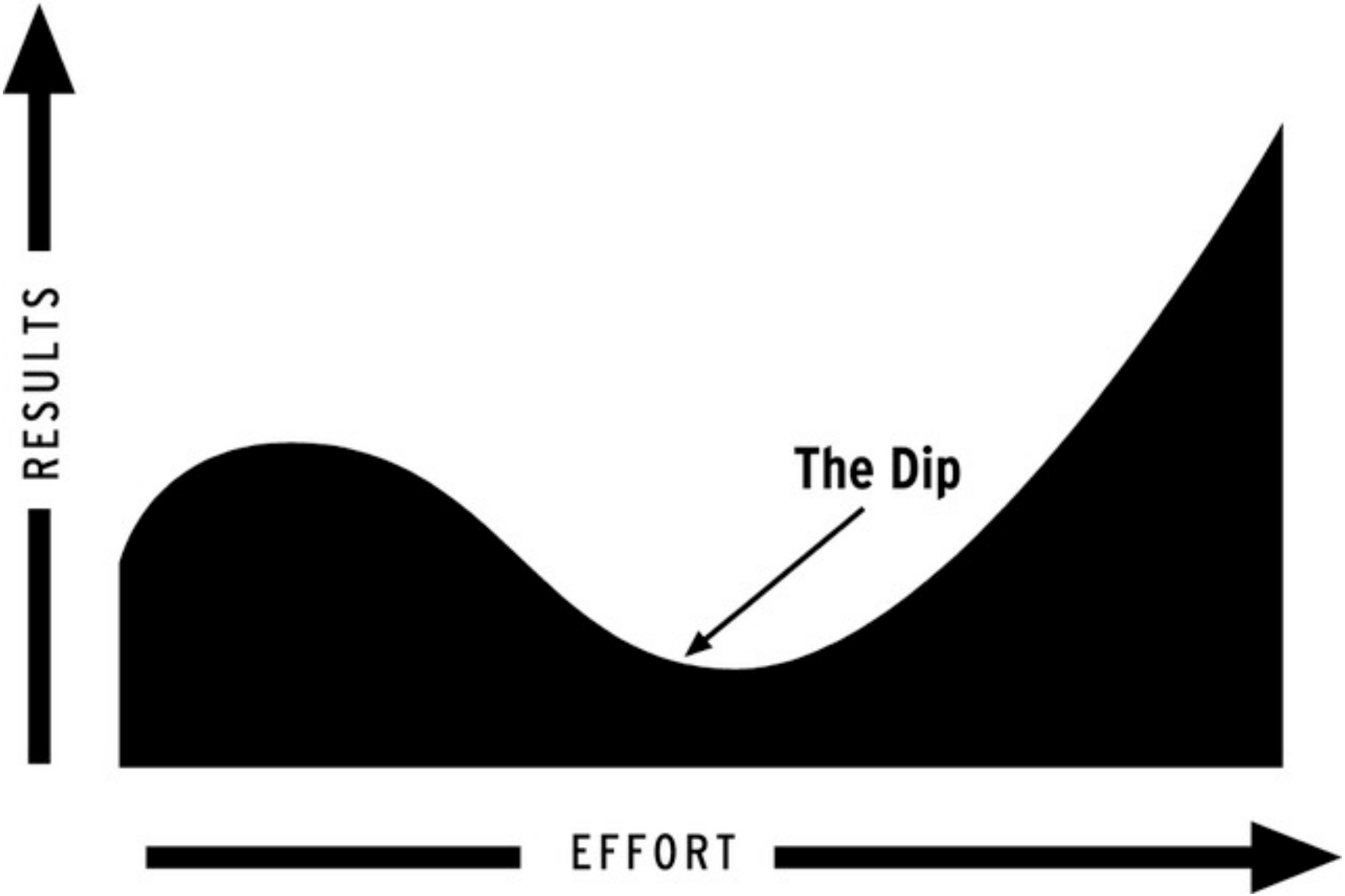
"Change must happen with people not to them."

— David Hussman
http://devjam.com

Change is hard to maintain.

We get worse before
we get better.

It's easy to back slide.



http://sethgodin.typepad.com/the_dip/2007/05/images_from_the.html

The Dip.

http://sethgodin.typepad.com/the_dip/

We have to push through the dip.

Thus why force doesn't work well...

Change takes time.

Be patient.

Change can overwhelm.

Don’t try to change too much at once.

Be aware of the politics of your organization.

http://www.history.navy.mil/photos/images/h96000/h96919k.jpg

“It's easier to ask forgiveness than it is to get permission.”

— Grace Hopper

Do it, prove it.

People rarely argue with success.

It can be done!

Good luck!

Thanks!

Nathaniel T. Schutta
@ntschutta