Supporting Teams When the Tech is Unknown

====

Presented at:

- Tech Nottingham - August 3, 2015

====

In this session you'll learn the steps needed to support a team through their first development project with a new technology. The lessons learned here can be applied to any team working outside of their comfort zone.

Specifically, we'll talk about how to pace your team to success by:

- Breaking down a project into achievable components and structuring an Agile schedule to deliver them
- Promoting radical transparency between stakeholders and developers
- Mitigating the learning curve of a new platform by building on known best practices
- Limiting the avalanche of new information through just-in-time learning
- Keeping each person on the team motivated and in the zone by customising how you engage, and by addressing -head-on- the anxiety which comes from building software when it feels like all your tools have changed

AGENDA

- Define the underlying purpose
- Scope out the challenge you're facing
- Plan (and pace) sprints
- Maintain focus and motivation

- Purpose: create a "why" statement for the change
- Scope: artefact is project approach doc
- Planning sprints: artefact is Epics
- Focus: formatting communications in digestable chunks

This is a talk I've been giving in various formats for a couple of years in the Drupal community.

Drupal 8 is just around the corner and there are a lot of devs who are terrified of the unknown changes that are about to hit them. Every project has unknowns though, and how we run the project is what allows teams to dive into new challenges with enthusiasm.

For those who are:
- **resistant to change**,
- **eager but need to remain motivated** for the entire change process.

your vision of the future must answer the question "why?"

Disclaimer:

“Change management” (or “organisational change”) is something I’ve been exploring at working, and talking about at conferences for a few years.

There are eight steps identified by John Kotter. As a project manager, I focus on three.

1. Create urgency.
2. Assemble a guiding coalition.
3. Form a strategic vision for change.
4. Enlist a volunteer army
5. Enable action by removing barriers.
6. Generate short-term wins.
7. Sustain acceleration.
8. **Institute** change by anchoring it in corporate culture.

There are eight steps identified by John Kotter. As a project manager, I focus on three.

1. Create urgency.
2. Assemble a guiding coalition.
3. Form a strategic vision for change.
4. Enlist a volunteer army
5. Enable action by removing barriers.
6. Generate short-term wins.
7. Sustain acceleration.
8. **Institute** change by anchoring it in corporate culture.

"Average companies give their people something to work on.

"In contrast, the most innovative organisations give their people something to work toward."

Simon Sinek

PLAN TO DEAL WITH EMOTIONS

Denial

Anger

Bargaining

Depression

Acceptance

Kubler-Ross's five stages of grief

"People who come to work with
a clear sense of WHY are
less prone to giving up after
a few failures because they
understand the higher cause."
Simon Sinek

Define Your Why.

REWARDS & RISKS

Expose fears and hesitations with the change.

Allows team to uncover motivators that will keep people engaged throughout the project.

Name, and explicitly plan for the biggest risk factors in the project.

If not managed, can surface differences between team members that are difficult to recover from.

ENSURING SUCCESS

1. Define scope.

2. Plan (and pace) sprints.

3. Maintain focus and motivation.

Successful change management is more likely to occur if the following are included:

- the change benefits the leaders
- effective communication informs stakeholders for the reasons **why** the change is necessary
- skills upgrading / training is included in the roll-out plan
- resistance is acknowledged and countered to align employees with the overall strategic direction of the organization
- personal counselling is provided to alleviate fears
- monitoring and fine tuning is applied

What are we going to do?

DEFINING THE SCOPE OF WORK

Define Scope

REWARDS & RISKS

Allows you to identify and name project gremlins.

Allows you to get a read on how / when stakeholders want to be involved in the project.

Allows you to start the idea of a “won’t build” list.

Can cause tensions if stakeholder thinks you’re trying to avoid work with the “won’t build” list.

RADICALLY TRANSPARENT PLANNING ARTEFACTS

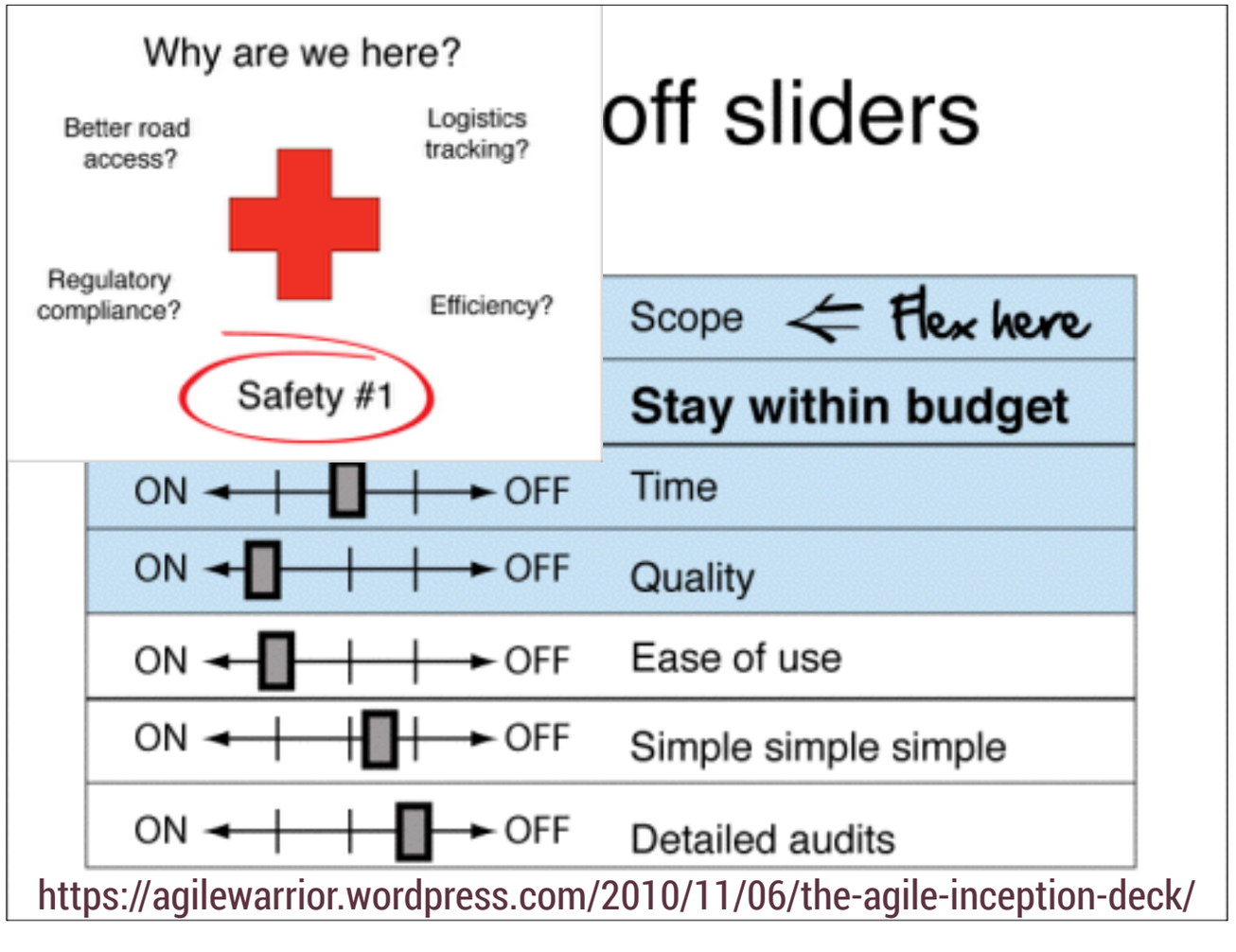
- Inception Deck.
- User Story Map.
- Project Approach Document.
- Epics / Backlog
- The “Won’t Build” List.

- Inception deck: look it up online. There’s a standard format for it. I’ve had mixed results with it -- and I don’t think it really works for distributed teams. Better as a F2F artefact that you co-build with stakeholders.
- User Story Map. Define with your stakeholder the entire journey for each user if they had unlimited budget and time. Then pick off the easiest wins from the most important tasks. If you’re doing agency work, you may not have the luxury of doing this. But if you don’t, you risk misunderstanding requirements and scaffolding out the wrong infrastructure.
- Project Approach Document. List of EPICS which includes the MVP for any given Epic as well as possible enhancements. This single document should get sign-off before moving it into your backlog.
- Backlog should start with Epic definitions from the project approach document, and the first 1-3 sprints written out in detailed User Stories.
- As soon as it is politically feasible, start a “won’t build” document. Any stories in your backlog that are not a priority for the current build should be marked as “won’t fix” and pulled into the “won’t build” list. This becomes the justification for a second round of funding and includes the rationale for why each item was not prioritised.

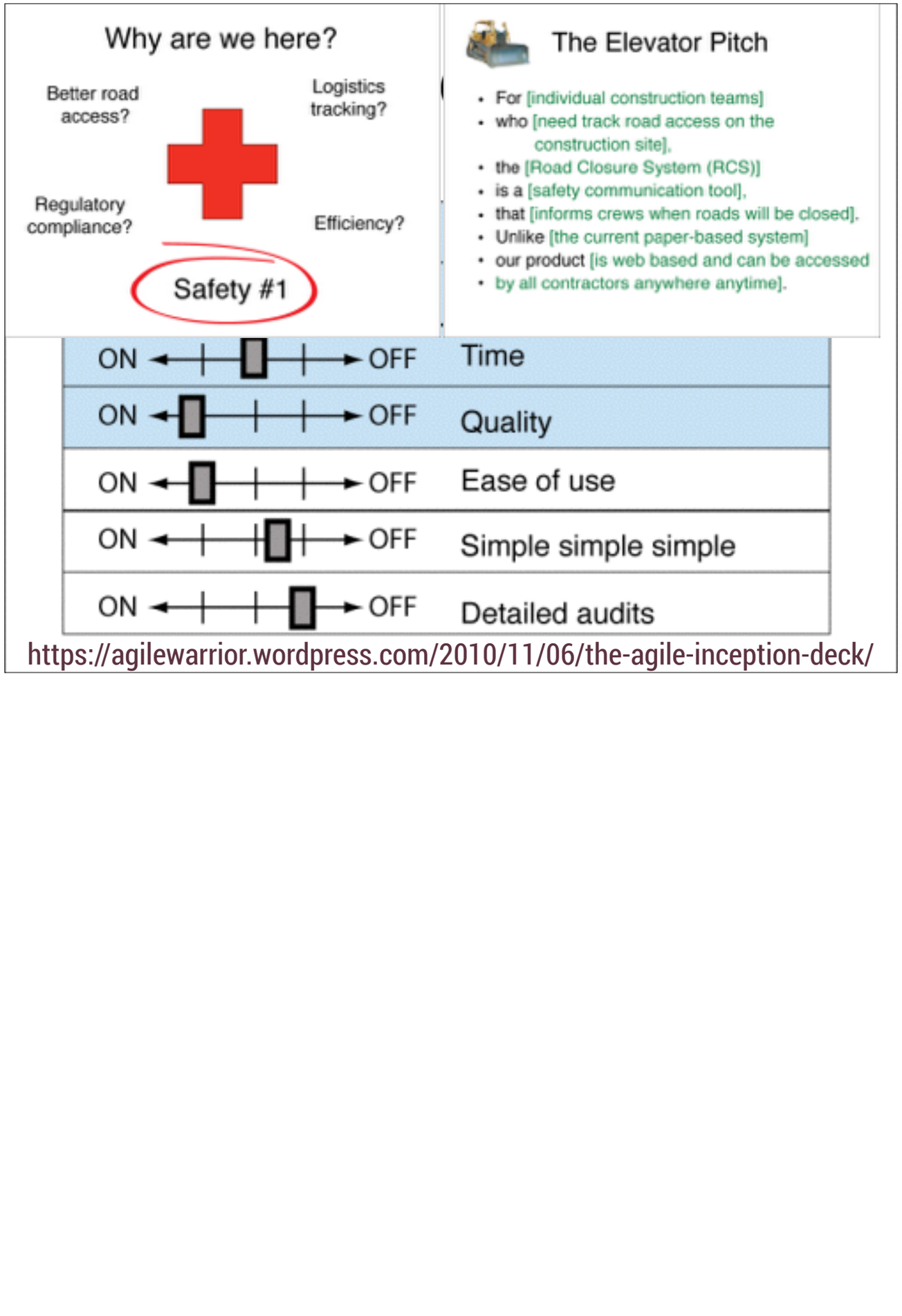
Trade-off sliders

ON ◄──┼──┼──┼──► OFF	Scope ⇐ Flex here
ON ◄──┼──┼──┼──► OFF	**Stay within budget**
ON ◄──┼──┼──┼──► OFF	Time
ON ◄──┼──┼──┼──► OFF	Quality
ON ◄──┼──┼──┼──► OFF	Ease of use
ON ◄──┼──┼──┼──► OFF	Simple simple simple
ON ◄──┼──┼──┼──► OFF	Detailed audits

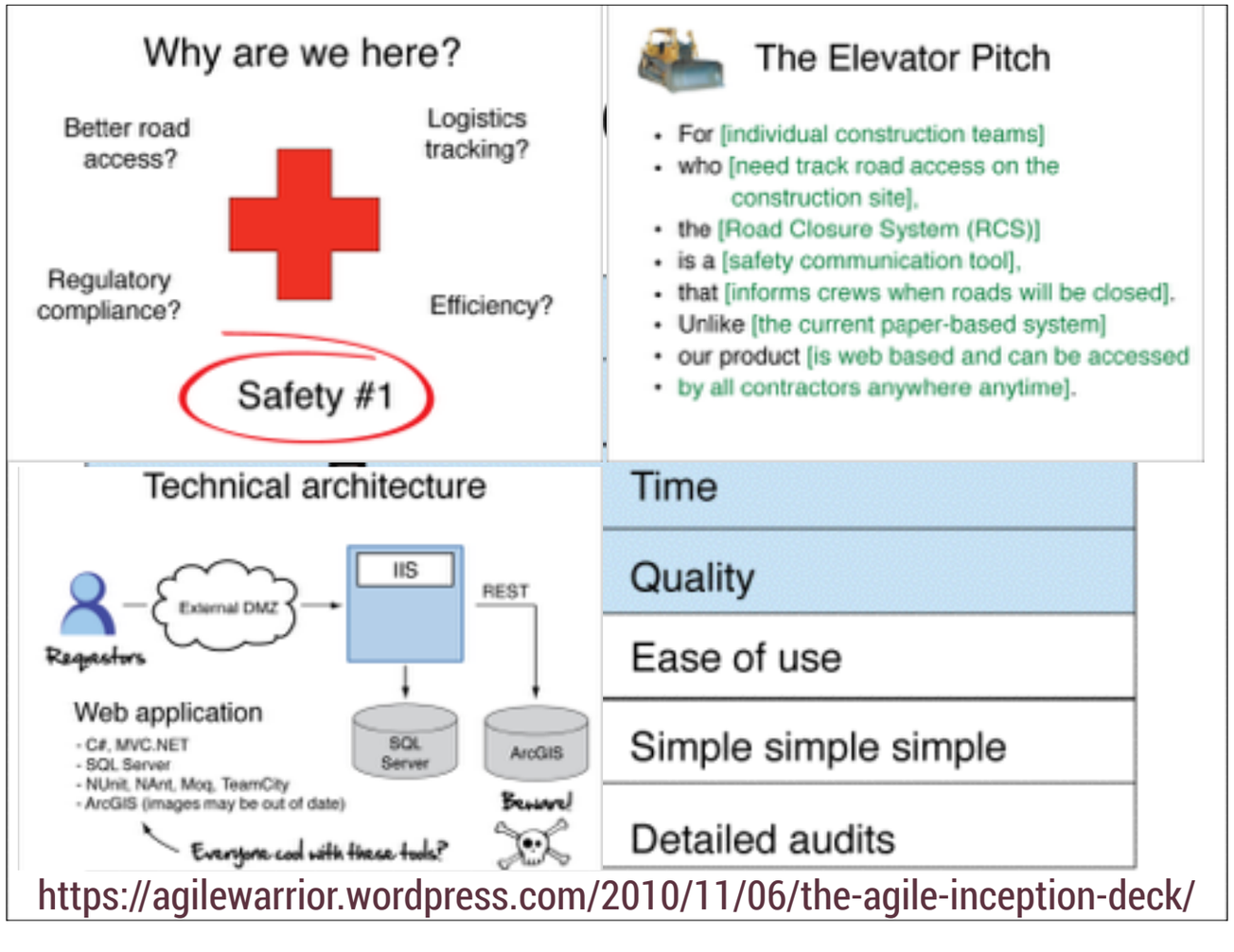
https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/



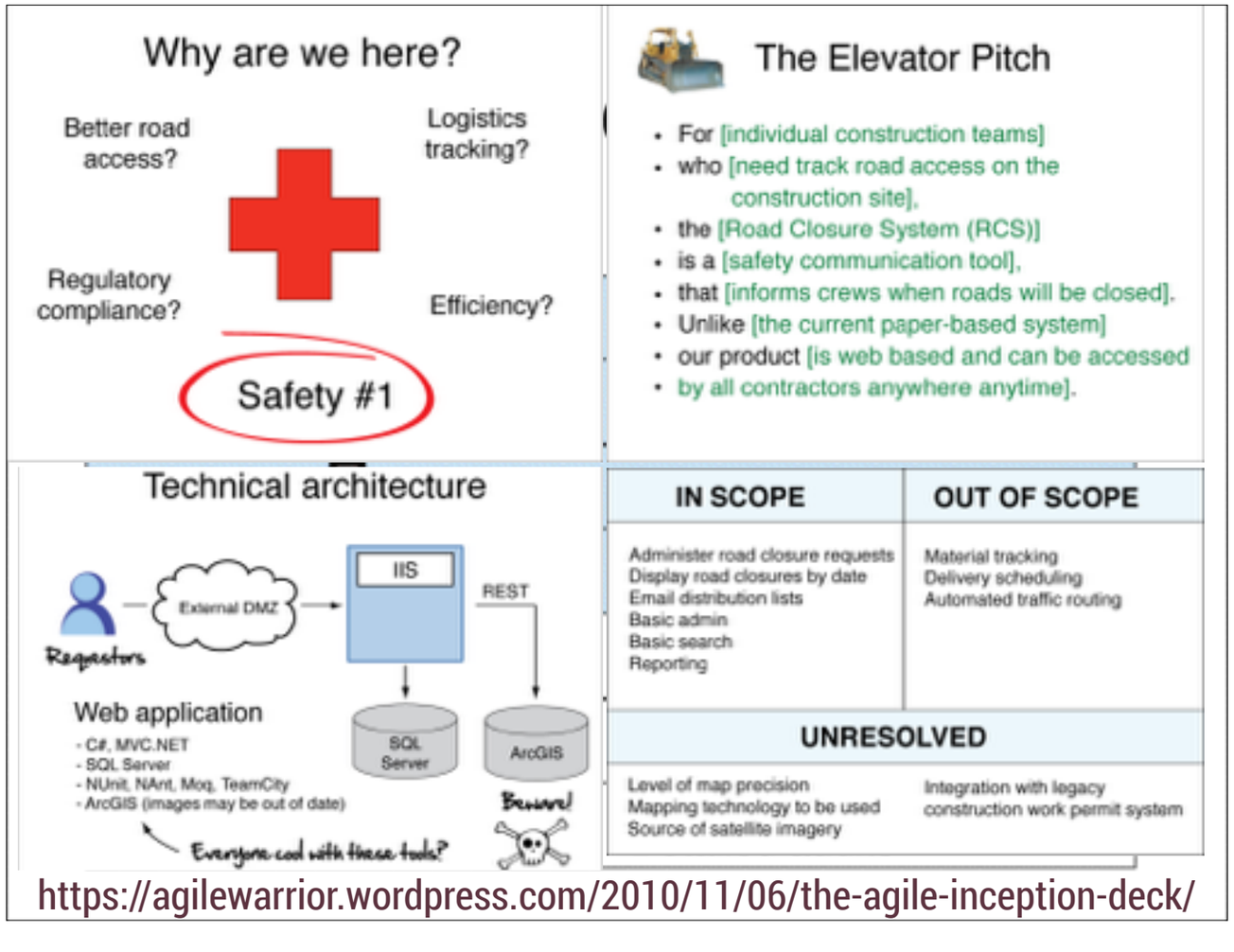
https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/



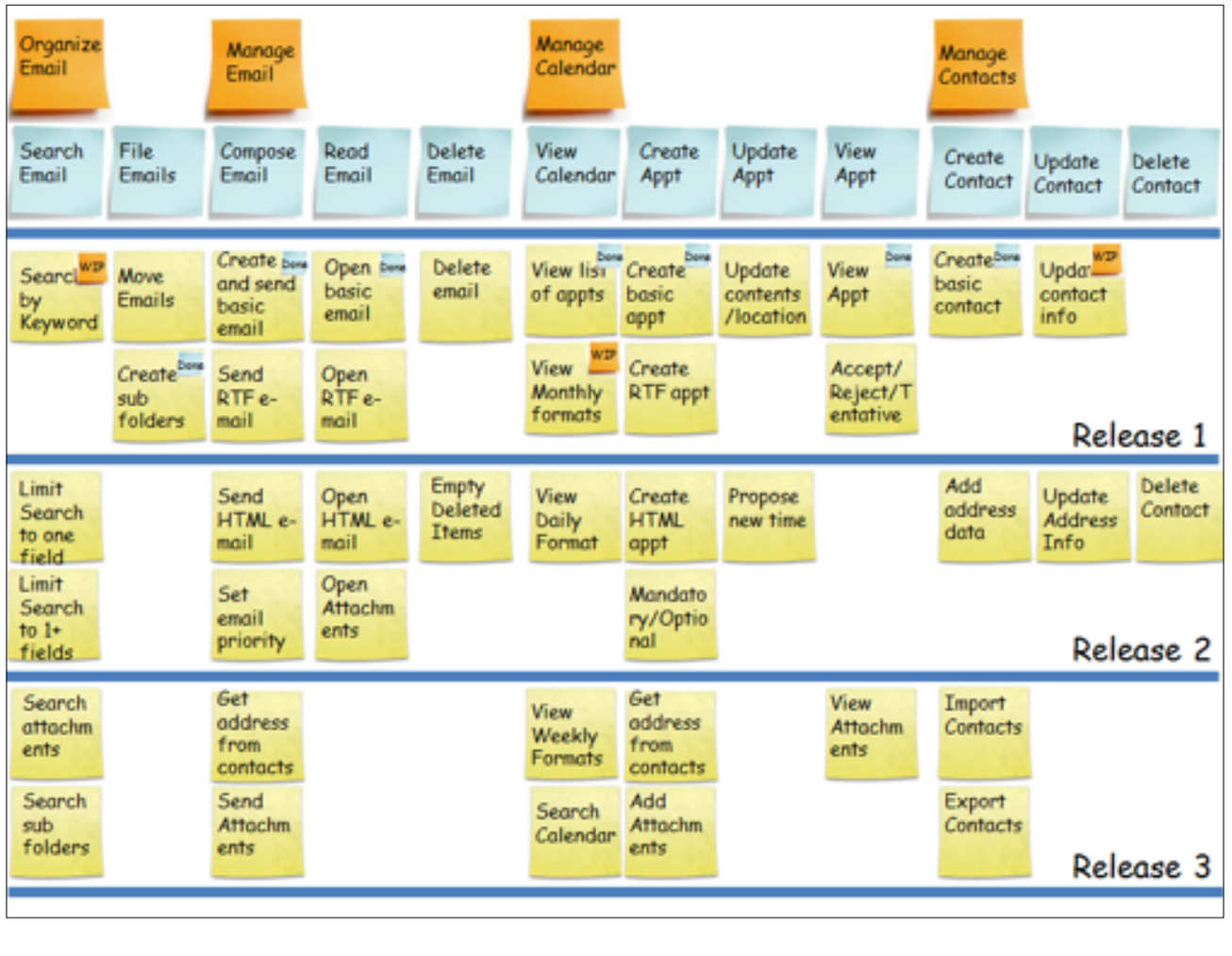
https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/

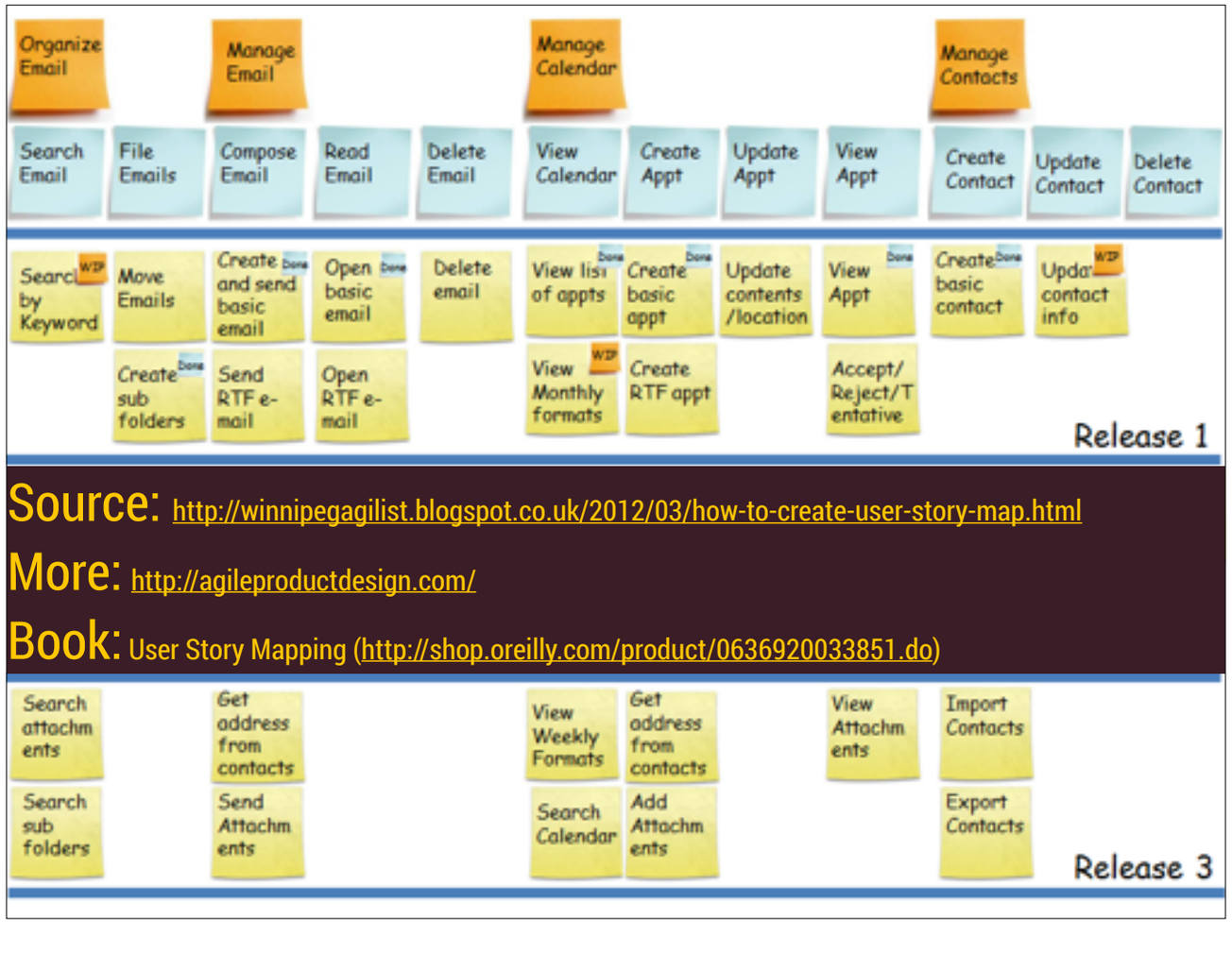


https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/



https://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/


Organize Email
Manage Email
Manage Calendar
Manage Contacts
Search Email
File Emails
Compose Email
Read Email
Delete Email
View Calendar
Create Appt
Update Appt
View Appt
Create Contact
Update Contact
Delete Contact
Search by Keyword
Move Emails
Create and send basic email
Open basic email
Delete email
View list of appts
Create basic appt
Update contents /location
View Appt
Create basic contact
Update contact info
Create sub folders
Send RTF e-mail
Open RTF e-mail
View Monthly formats
Create RTF appt
Accept/ Reject/T entative
Release 1
Limit Search to one field
Send HTML e-mail
Open HTML e-mail
Empty Deleted Items
View Daily Format
Create HTML appt
Propose new time
Add address data
Update Address Info
Delete Contact
Limit Search to 1+ fields
Set email priority
Open Attachm ents
Mandato ry/Optio nal
Release 2
Search attachm ents
Get address from contacts
View Weekly Formats
Get address from contacts
View Attachm ents
Import Contacts
Search sub folders
Send Attachm ents
Search Calendar
Add Attachm ents
Export Contacts
Release 3


Organize Email
Manage Email
Manage Calendar
Manage Contacts
Search Email
File Emails
Compose Email
Read Email
Delete Email
View Calendar
Create Appt
Update Appt
View Appt
Create Contact
Update Contact
Delete Contact
Search by Keyword
Move Emails
Create and send basic email
Open basic email
Delete email
View list of appts
Create basic appt
Update contents /location
View Appt
Create basic contact
Update contact info
Create sub folders
Send RTF e-mail
Open RTF e-mail
View Monthly formats
Create RTF appt
Accept/ Reject/T entative
Release 1
Source: http://winnipegagilist.blogspot.co.uk/2012/03/how-to-create-user-story-map.html
More: http://agileproductdesign.com/
Book: User Story Mapping (http://shop.oreilly.com/product/0636920033851.do)
Search attachm ents
Get address from contacts
View Weekly Formats
Get address from contacts
View Attachm ents
Import Contacts
Search sub folders
Send Attachm ents
Search Calendar
Add Attachm ents
Export Contacts
Release 3

Week	Beginning Date	Introduction of What New Tasks	Demo Week?
1	August 25	Kickoff	no
2	September 1	Discovery review; begin design	no
3	September 8	Server configuration, site building / configuration tasks	no
4	September 15	Front end development + theme	no
5	September 22	Migration begins	**yes**
6	September 29	Document storage and access	no
7	October 6	User management and directories	**yes**
8	October 13	Configuring events for meetings	no
9	October 20	Notifications	**yes**
10	October 27	User acceptance testing from core stakeholders	TBD
11	November 3	Content editor training	n/a
12	November 10	Soft launch	n/a
13	November 17	Feedback from wider stakeholder community	n/a
14	November 24	Feedback from wider stakeholder community	n/a

When are we going to do it?

PLAN THE SPRINTS; SPRINT THE PLAN.

Working in an Agile fashion does not mean you avoid planning completely!!

Plan your project

REWARDS & RISKS

Allows team to understand the scaffolding they should put in place for features they'll build.

Place "hard" tasks when team is likely to be most engaged (e.g., consider holidays).

Build in capacity for iteration; plan to replace elements with increasingly more complex code.

If your plan is too rigid, you start getting into waterfall-style promises.

A PROJECT IS A MARATHON

- Your project approach document outlines the Epics you are going to build.
- From the project approach document, schedule easy wins every now and then.

Structure an Agile schedule to ensure the best possible relationships between developers and stakeholders, through radical transparency and by breaking down a project into achievable components.

- Working with clients on schedules: per-Epic create an MVP through to Cadillac definition of how the business value could be exposed. Then work with the team to **decide what the cheapest way is to expose a given function long term and short term**. Decide: should we duct tape now and plan to replace; or should we scaffold now and see faster results later? http://blog.codeship.com/five-suggestions-for-building-an-infrastructure-for-innovation/
- **Coach teams into building more complicated functionality**: e.g. the PDF server. We started with an MVP of print-friendly style sheets. This exposed the functionality to early adopters while we did more research on how to set up the server + choose best tech for implementation. Then checked stats to see how many people were using the print-friendly button + what browsers were supported + who the target market was for the functionality. Once we had the data to make a decision, then we implemented the research that had been going on "behind the scenes" and implemented the PDF server. A few sprints later we phased out the print-friendly style sheet button because it was causing confusion and was an extra set of style sheets to maintain (we couldn't use exactly the same ones for wkhtmltopdf and so things were getting out of sync).
- Like training for a marathon: progressively harder for three weeks; and then every fourth week is a recovery week where you do as much as the first week (or less!).

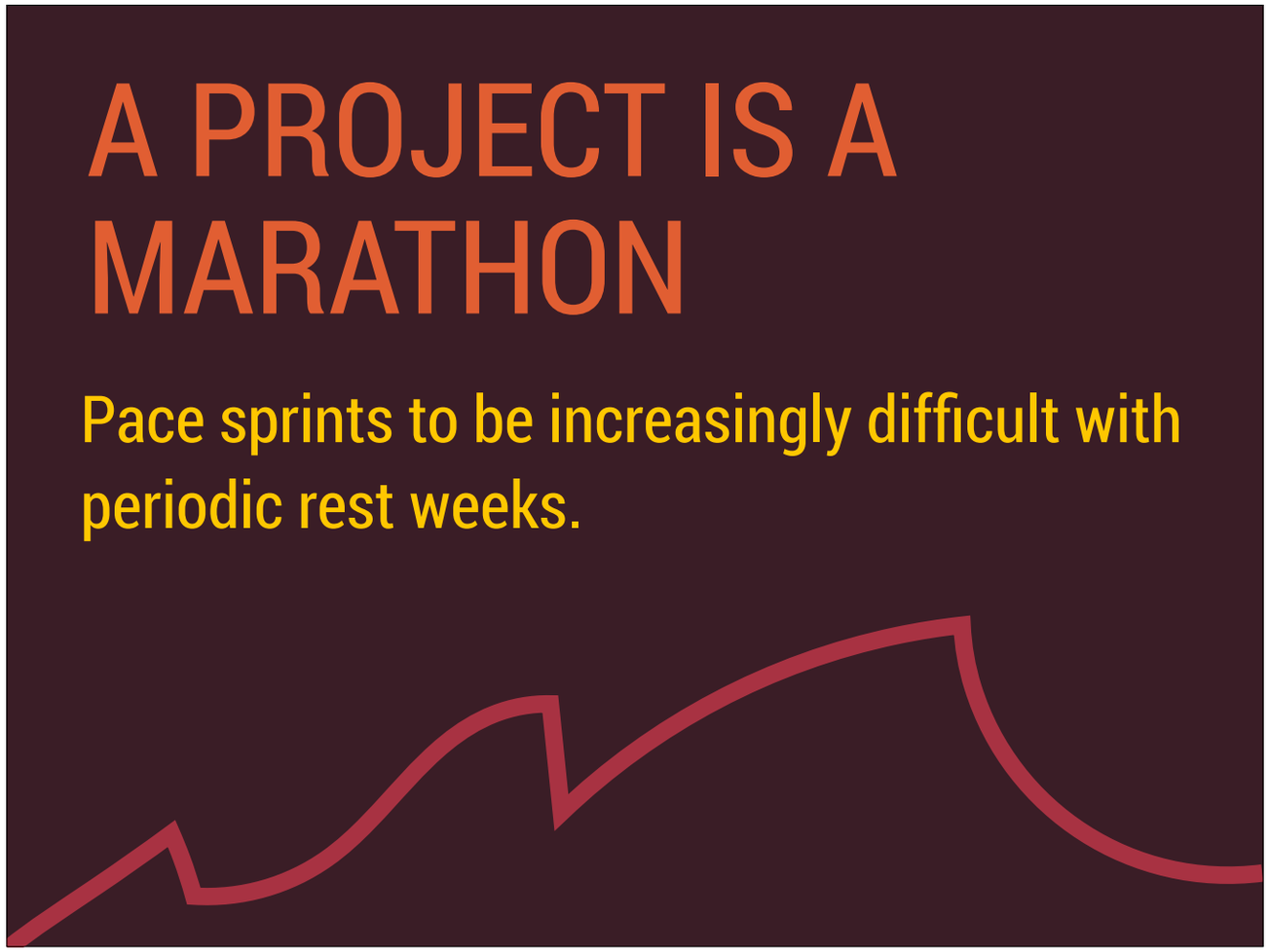
A PROJECT IS A MARATHON

Pace sprints to be increasingly difficult with periodic rest weeks.

- Your project approach document outlines the Epics you are going to build.
- From the project approach document, schedule easy wins every now and then.

Structure an Agile schedule to ensure the best possible relationships between developers and stakeholders, through radical transparency and by breaking down a project into achievable components.

- Working with clients on schedules: per-Epic create an MVP through to Cadillac definition of how the business value could be exposed. Then work with the team to **decide what the cheapest way is to expose a given function long term and short term**. Decide: should we duct tape now and plan to replace; or should we scaffold now and see faster results later? http://blog.codeship.com/five-suggestions-for-building-an-infrastructure-for-innovation/
- **Coach teams into building more complicated functionality**: e.g. the PDF server. We started with an MVP of print-friendly style sheets. This exposed the functionality to early adopters while we did more research on how to set up the server + choose best tech for implementation. Then checked stats to see how many people were using the print-friendly button + what browsers were supported + who the target market was for the functionality. Once we had the data to make a decision, then we implemented the research that had been going on "behind the scenes" and implemented the PDF server. A few sprints later we phased out the print-friendly style sheet button because it was causing confusion and was an extra set of style sheets to maintain (we couldn't use exactly the same ones for wkhtmltopdf and so things were getting out of sync).
- Like training for a marathon: progressively harder for three weeks; and then every fourth week is a recovery week where you do as much as the first week (or less!).



- Your project approach document outlines the Epics you are going to build.
- From the project approach document, schedule easy wins every now and then.

Structure an Agile schedule to ensure the best possible relationships between developers and stakeholders, through radical transparency and by breaking down a project into achievable components.

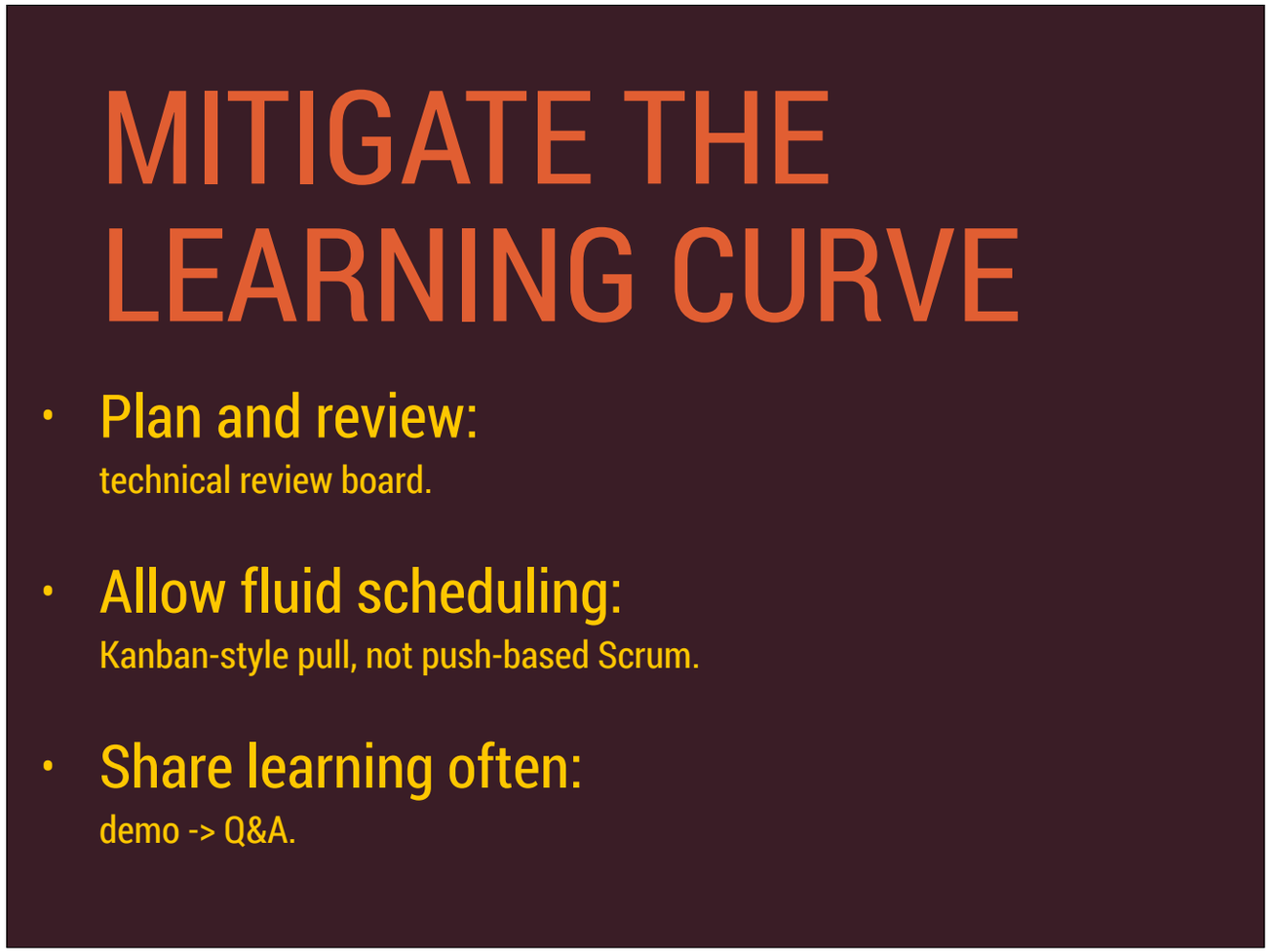
- Working with clients on schedules: per-Epic create an MVP through to Cadillac definition of how the business value could be exposed. Then work with the team to **decide what the cheapest way is to expose a given function long term and short term**. Decide: should we duct tape now and plan to replace; or should we scaffold now and see faster results later? http://blog.codeship.com/five-suggestions-for-building-an-infrastructure-for-innovation/
- **Coach teams into building more complicated functionality**: e.g. the PDF server. We started with an MVP of print-friendly style sheets. This exposed the functionality to early adopters while we did more research on how to set up the server + choose best tech for implementation. Then checked stats to see how many people were using the print-friendly button + what browsers were supported + who the target market was for the functionality. Once we had the data to make a decision, then we implemented the research that had been going on "behind the scenes" and implemented the PDF server. A few sprints later we phased out the print-friendly style sheet button because it was causing confusion and was an extra set of style sheets to maintain (we couldn't use exactly the same ones for wkhtmltopdf and so things were getting out of sync).
- Like training for a marathon: progressively harder for three weeks; and then every fourth week is a recovery week where you do as much as the first week (or less!).

MITIGATE THE LEARNING CURVE

Mitigate the learning curve of a new platform by building on existing best practices, and limiting the avalanche of new information through just-in-time learning.

On the IASC build none of the developers had OpenAtrium experience, but several had Drupal experience. They built on what they knew, and the connections they'd made at various points:

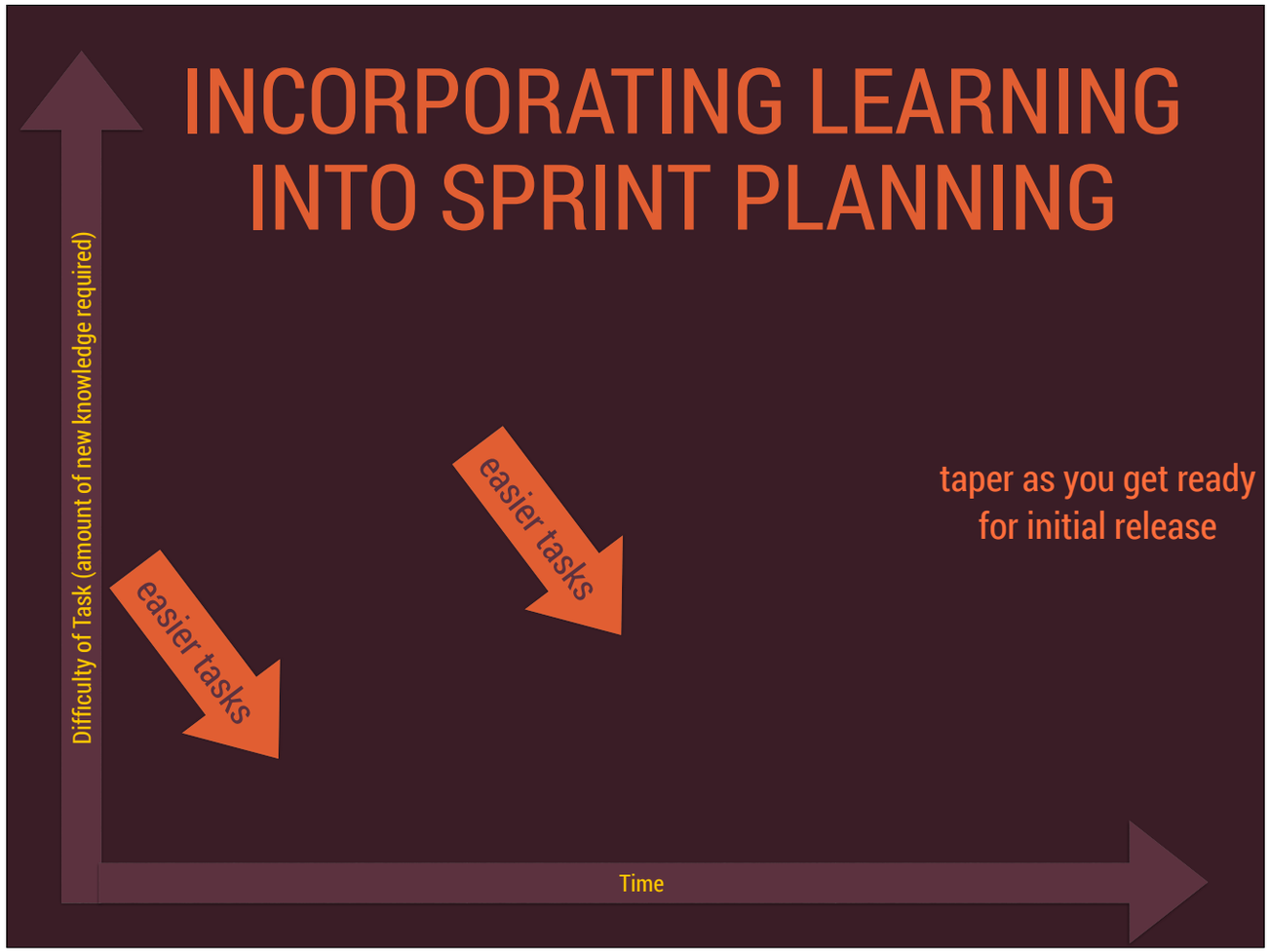
1. Plan your build. **Technical review board**: write up a loose plan for implementation + verify with others before reviewing.
2. **Work in a Kanban style** where unfinished tickets rolled over into the next week, but every sprint was still planned + loaded.
3. Share learning **internal weekly demo into a hands-on Q&A session** where devs knew they would get extra eyes on the problem they were having.



Mitigate the learning curve of a new platform by building on existing best practices, and limiting the avalanche of new information through just-in-time learning.

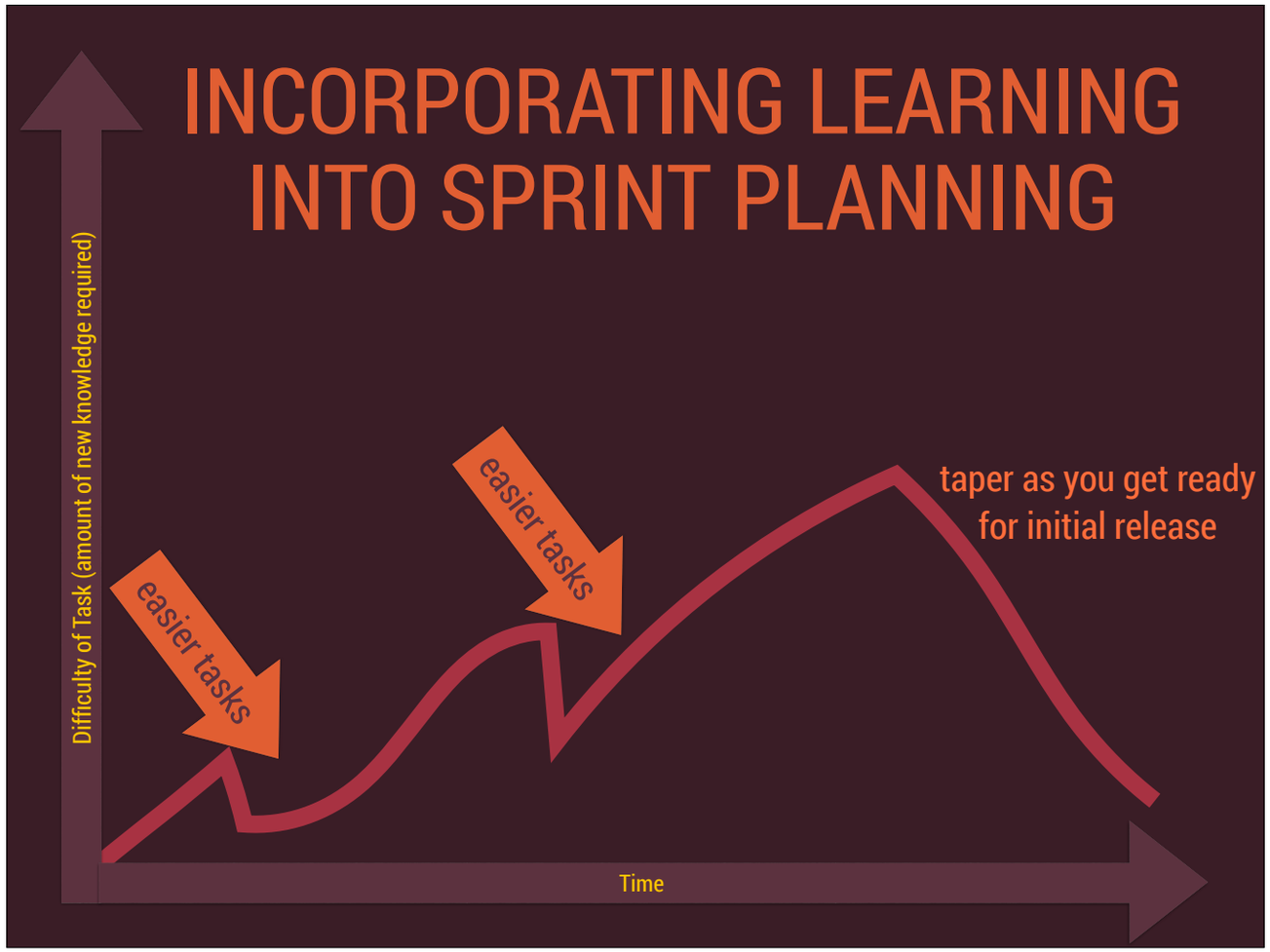
On the IASC build none of the developers had OpenAtrium experience, but several had Drupal experience. They built on what they knew, and the connections they'd made at various points:

1. Plan your build. **Technical review board**: write up a loose plan for implementation + verify with others before reviewing.
2. **Work in a Kanban style** where unfinished tickets rolled over into the next week, but every sprint was still planned + loaded.
3. Share learning **internal weekly demo into a hands-on Q&A session** where devs knew they would get extra eyes on the problem they were having.



In the “taper” period you are also going through QA and making sure there are no outstanding tasks.

This ends up being your buffer. If things have gone poorly, you can eat up this time by extending some sprints. If things have gone well, you might be able to squeak in an enhancement from your project approach document.



In the “taper” period you are also going through QA and making sure there are no outstanding tasks.

This ends up being your buffer. If things have gone poorly, you can eat up this time by extending some sprints. If things have gone well, you might be able to squeak in an enhancement from your project approach document.

LEAVE ROOM FOR
UNEXPECTED DELIGHTS.

Are we almost there yet?

MAINTAINING MOMENTUM

This is assuming a short running team that will disband after the project.

The longer the team is together, the more you should empower the team members to run their own project with your support.

MOTIVATE ME: WHISK(E)Y NOT REQUIRED

Ask developers what motivates them.

It’s not always parties (or cash) that they’re after.

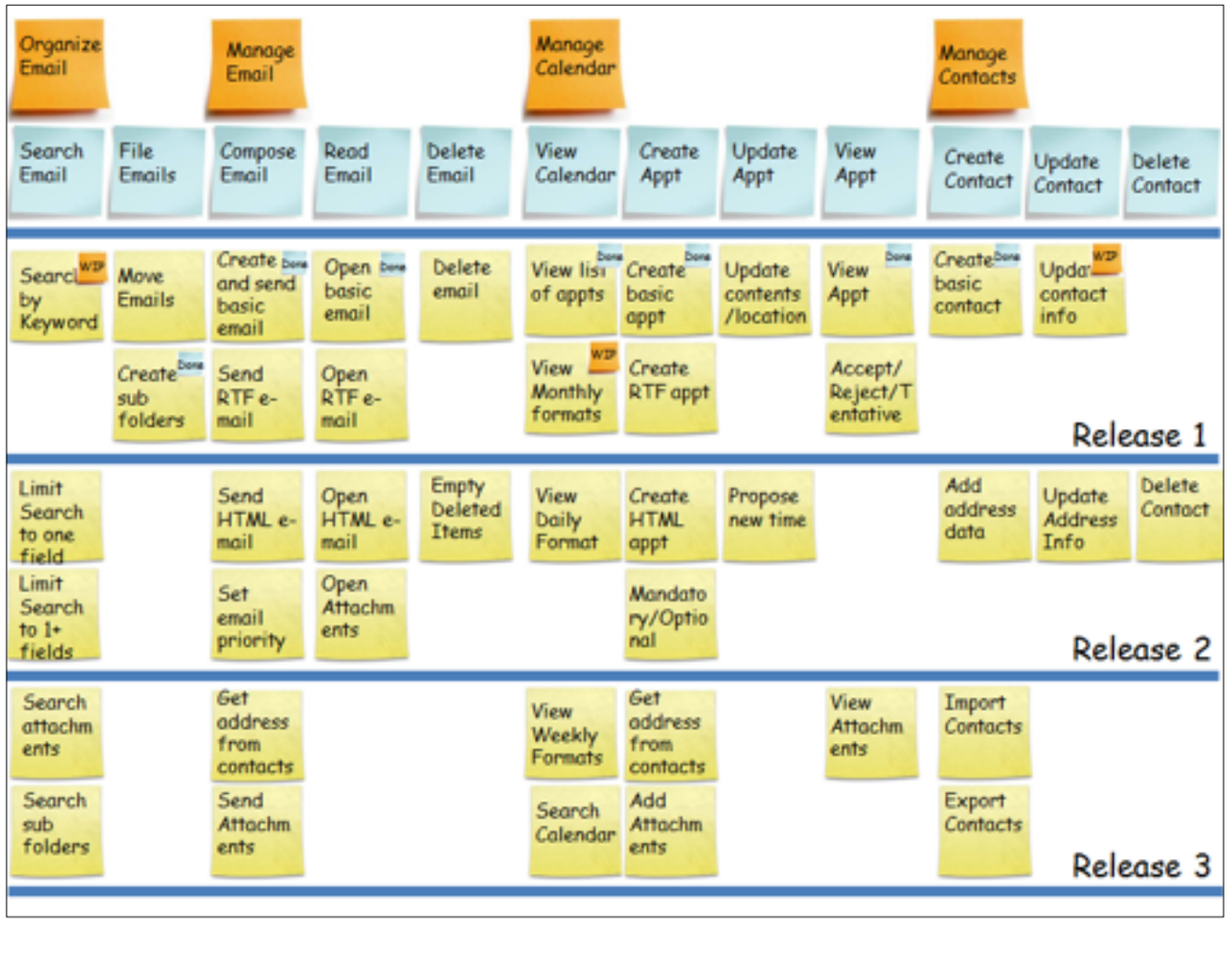
Committed	Bigger (choose ONE to move to Committed)	Smaller (*OR* choose up to FOUR to move to
Ed		
PROJ-347 Personal contact list	PROJ-202 Global Groups	
PROJ-652 Text update - orgs require search		PROJ-233 Rename "About" to "Help"
PROJ-609 IE11 not redirecting after form save		
PROJ-524 Management-friendly shortcuts		PROJ-631 Phone form field too narrow
PROJ-356 Encourage user to check-in after Global profile is		PROJ-641 Allow dates in the past
		PROJ-651 Add "old browser" notice for IE9
Joe		
~~PROJ-577 Cron to sync names on groups~~		PROJ-381 Update 404 page
PROJ-110 Implement Helmet middleware for node services	PROJ-541 Archive only the selected profile	PROJ-637 Orphan filter leaks
Final deployment		PROJ-483 Change domain name of dev/stage
Bare bones developer documentation		
PROJ-442 Contact list by "org type"	**Recommend not choosing**	
	PROJ-583 Organisational pre-filtering by contact list	
	PROJ-542 Sort order: case + accents	
Tim - FULLY BOOKED		
PROJ-185 Report incorrect info		
PROJ-620 Last edited stamp		
Backlog		
	PROJ-659 Audit + fixup of system notifications	PROJ-644 Easier access to own profile (for easier sharing)
	PROJ-527 Add cancel to confirmations	PROJ-516 Hide "send claim" email from non-orphans
	PROJ-446 Default values for orphan global profile fields	PROJ-575 Add Magyar, Romanian translations
	PROJ-544 Browser preserves form state (makes un-	PROJ-436 Update edit links on the dashboard to include text
	PROJ-655 Ensure API cannot allow duplicate profiles to	PROJ-523 Profile font size increase
	PROJ-171 Create vanity URLs for profiles	
	PROJ-508 Refine BlackMesh health checks for HID apps	PROJ-657 Include regions in checkin list
	PROJ-498 Update logstash processing to map HID app	

Give choice: with a set number of features, some will probably be easier and some will probably be harder. Work with the developers to figure out a nice balance that doesn't burn them out. You may find that some developers find the choice STRESSFUL because they don't have sufficient context / contact with the client to know what is the right thing to work on. In this case they may *prefer* to have their tickets picked for them.

Your backlog tool is a great place to record decisions. but it's not always the best place to make decisions.

Stakeholder planning matrix for the last few days of an overloaded sprint.

Many people love choice. Format things in a way that people can easily make choice.



Force people to think creatively to find the best possible solution for any given problem.

Don’t rely on what’s in the EPIC or the USER STORY without the context of the whole project. Where necessary, go back to the BUSINESS VALUE, and check your USER STORY MAP to see if there’s a “cheaper” way to get closer to the functionality you’re trying to expose.

Celebrate wins; share losses.

Put GIFs in tickets and email. Take the WORK seriously, but not yourself.

Take blame where it’s warranted. Fix broken processes, and never allow your culture to point blame at a single person. Our processes are what fail us; individuals are rarely the sole point of blame. “Five why” every problem: Did they need more training? More time? Stricter peer review? Better QA?

Celebrate wins; share losses.

Put GIFs in tickets and email. Take the WORK seriously, but not yourself.

Take blame where it’s warranted. Fix broken processes, and never allow your culture to point blame at a single person. Our processes are what fail us; individuals are rarely the sole point of blame. “Five why” every problem: Did they need more training? More time? Stricter peer review? Better QA?

MOTIVATE

Keep each person on the team motivated and in the zone by customising how you engage, and by addressing--head-on--the anxiety which comes from building software when it feels like all your tools have changed.

MOTIVATE

- Ask the team what motivates them.
- Give choice.
- Have high standards
  which allow for creative solutions.
- Celebrate wins.

Keep each person on the team motivated and in the zone by customising how you engage, and by addressing--head-on--the anxiety which comes from building software when it feels like all your tools have changed.

RESOURCES

Managing Change
http://gitforteams.com/resources/change-management.html

A Developer's Primer To Managing Developers
https://austin2014.drupal.org/session/developers-primer-managing-developers.html

Things I Learned From Managing My First Project
https://drupalize.me/blog/201312/things-i-learned-managing-my-first-project