

Agile Release Strategies

Niklas Björnerstedt, Core group
Johannes Brodwall, Steria

Agenda

- Introduction
- What are Agile Release Strategies
- Why does Release Strategies deserve a second look
- Some Release Patterns
- Case study tutorial
- Debriefing

Who are you?

Participant story 1:
As a *product owner* I want to
learn how to *release earlier* so
that I can *deliver value*

Participant story 2:
As a *developer* I want to learn
how to *make the old and the
new system work together* so
that I can *gradually replace a
system*

Participant story 3:
As a *agile coach* I want to
learn how to *safely execute*
releases so that I can *help*
projects avoid trouble

Who are you?

- As a *product owner* I want to learn how to *release earlier* so that I can *deliver value*
- As a *developer* I want to learn how to *make the old and the new system work together* so that I can *gradually replace a system*
- As a *agile coach* I want to learn how to *safely execute releases* so that I can *help projects avoid trouble*
- *Something else?*

What are Agile Release Strategies?

- Patterns for how to get your project into production
- Collection of patterns <http://wiki.cantara.no/display/ARS/>
- Book project?
- "I have that pattern"

Many projects do not put enough effort into reducing release length

Find your
Minimal Releaseable Product!

Case study: Book club system

- *Goal: replace legacy system, increase cross selling between channels, improve customer support*
- *Many clubs*
- *Differences in rules, no. of members, frequency...*

Replacement or enhancement?

Support
Expertise
Flexibility

Replacement or enhancement?

Support
Expertise
Flexibility

Self service
Automation
New services

Why release often?

- **Self service** creates demand
 - *Pain informs priority*
- **Automation** only benefits us if used
 - *Small inventory creates value faster*
- **New services** require understanding of market
 - *Real learning reduces risk*

Why?

- Pain informs priority
- Small inventory creates value faster
- Real learning reduces risk

How? - Patterns

The importance of frequent releases is widely recognized

We want to examine the **practices** and **patterns** that people use to get there

wiki.cantara.no/display/ARS/Patterns

has 40 (more or less defined) patterns

Product backlog patterns

- New user set
 - (book club members)
- Cut to non-negotiables
 - (customer loyalty)
- Partition the workflow
 - (ordering, shipping, billing, etc)

Legacy data patterns

- Shared database
- Replicated database
- Data service layer

Risk reduction patterns

- Limited releases
 - (focus group users, beta users)
- Facilitate switching
 - (link to open customer in 3270 system)
- Update downstream first
 - (new billing system)

Case study

- Select or discover **three patterns** you would apply
- Find two arguments why these work in the case study
- Find two adjustments to make them fit better
- Find two preconditions to using them

Spend 20 minutes doing this!

Case study - result template

Which pattern did you choose?

- New user set

How would you make it fit?

- Allow users to access existing data on existing system to see e.g. upcoming books

What preconditions would it require?

- Can access the database. Compatible data system, no firewall between old and new system

Case study - result template

Which pattern did you choose?

- *Data service layer*
- *(+ new user set)*

How would you make it fit?

- *Avoid dependencies of underlying legacy database*
- *Goal is to replace the data storage - successful project example (input split)*
- *Switch between database vendors*
- *Switch between COBOL-based and SQL-based data store (partial migration)*

What preconditions would it require?

Case study - result template

Which pattern did you choose?

- *Partition of workflow*

How would you make it fit?

- *Start with the registration process (or the step after it)*
- ***Alternative: Find a shared workflow step***
- *Control users*
- *Automate/support duplicate checking, "bounty hunter" detection*

What preconditions would it require?

- *There is a new business requirement for duplicate checking, fraud detection, "bounty hunter" detection etc*

Case study - result template

Which pattern did you choose?

- *Facilitate switching*

How would you make it fit?

- *New features to old customer*
- *Add a new feature using the old data and support switching between the old and the new system*
- *The customer service reps accessing a new feature with a web based system*
- *Gain goodwill from customer support department*

What preconditions would it require?

- *Technology of the old system*

Case study - result template

Which pattern did you choose?

- *Limited release*

How would you make it fit?

- *Expose new web based interface to customers that communicate with us a lot*

What preconditions would it require?

- *Can we find users that are representative?*
- *Can we find users that are demanding enough?*

Case study - result template

Which pattern did you choose?

- *Product-by-product*

How would you make it fit?

- *Find a simple book club (simplest? most complex? medium complex?)*

What preconditions would it require?

- *Will we make assumptions that we will make from the first club that will make it harder to implement more complex clubs*

Case study - result template

Which pattern did you choose?

-

How would you make it fit?

-

What preconditions would it require?

-

Audience questions

Which of these patterns did you already know? □

-

Which patterns or categories did you find useful?

-

Which patterns or categories do you need to succeed?

-

Homework

How does the following change in the context of the case study change your mind?

The legacy system is operated by a third party that is not interested in cooperating in the transition

As a *developer* I want to learn
how to ***make the old and the
new system work
together*** so that I
can *gradually replace a
system*