Workflow Design Workshop

Adam Hyde, March 2018

V 0.3

Introduction

Publishing workflows have evolved on a per-organisation basis through:

- the acquisition of software, which projects a software vendors view of workflow onto the publisher, and
- augmentation of the workflow by staffing and adhoc systems external to the original workflow management system

What we end up with is unique workflows with many culdesacs and eddies that do not reflect an ideal workflow. Rather a typical workflow reflects a historical process of work-arounds to best reconcile the difference between the original softwares prescribed workflow with what the publisher actually needs.

With PubSweet we have the opportunity to break this unhealthy cycle, and enable ourselves to imagine and build the workflow we want.

This workshop will provide the tools to do exactly that.

We will look at :

1. Understanding and Optimising Workflow
2. SignalPath System Design
3. Mapping workflows onto PubSweet
4. Documenting Per-Role Pathways

1. Understanding and Optimising Workflow

Often publishers believe they know their workflow, or that someone within their organisation knows it in its entirety. However, in my experience this is not the case. Publishing workflows contain so much nuance that often colleagues who are otherwise fully versed in the needs of their own job don't understand the nuance and detail of their colleagues work.

There are two ways to resolve this. The first is to put one person in charge of interviewing all those people in the organisation and documenting the workflow as they describe it. This will likely take weeks or months. As a result you will get a document describing in nice flowcharts, with some narrative, what happens now.

The second way is to get everyone in a room for an afternoon and do it yourself.

Not only is the second option more attractive from an efficiency point of view, it also produces better results. This is because open dialog within an organisation that enables everyone to understand what their colleagues actually do will lead to a commitment from each to help optimise the workflow for all, and ideas on how each person could change what they do to assist with lowering the workload for their colleagues. In other words, this second way enables your organistion to designed an optimsed workflow instead of purely documenting what happens now.

To do this we need the following:

- an afternoons commitment from at least one person per role within the publisher
- a room
- a whiteboard
- a facilitator
- some nice snacks and coffee!

The process of documenting and optimising a workflow at the same time is a process of open discussion. Like all good stories we start at the beginning and ask what happens first. We want those people that are actually involved in that part of the story to tell it. So, if you can, involve an author in the process to say what happens from their experience. As we go we write up the workflow in a very basic form. I prefer to describe the workflow in terms of :

1. Role

2. Action(s)

For each step. See below for an example from a recent client (Appendix1: Wormbase).

A basic step will look something like this:

Author *starts new submission*

1. Starts new submission
2. Fills out article metadata
    1. Title
    2. Author(s)
    3. Funder(s)
    4. References
    5. Keywords
3. Adds article text
4. Uploads figure
5. Suggests Reviewers
6. Submits Manuscript

When documenting these steps write them down in as clear a fashion as you can see on a surface that all can easily see (eg a whiteboard)…you will also revisit some of these as you go so leave room for alterations and additions.

Notice that we are not documenting the software, we are documenting the workflow. This is very important as we want the participants to leave behind their legacy understanding of workflow based on their current softwares so as to enable a new way of thinking about what they do. Later we will design a system to meet this optimal workflow design.

As we go through this process we are facilitating discussion. What we want is for anyone with a query to be able to ask a question about the step being discussed. We also want people to suggest better ways of doing things as we go, and to tease out as much nuance as possible about how things are currently done. This will require you to ask many clarifying questions about the current step, and it is important to drill down and get absolute clarity. It often means asking apparently stupid questions, but it is only by doing this that you will uncover what is actually happening and this is absolutely critical to understand before you can attempt improving upon the current way of doing things.

For example, I have found in several occasions that one of more parts of the workflow involve someone moving to a specific computer with the tools for the task, which may actually be in another room *or building*, to execute the task. It is important this detail is teased out so that everyone can understand how absurd some of the things are that their colleagues are forced to do. If they understand this they will be motivated to help optimise the workflow to flush out these absurdities and improve their colleagues workflow.

A word on facilitation of this process. You can facilitate this process for you own organisation but I recommend you avoid this if possible. Experience has shown me that existing hierarchies and inter-personal dynamics within an organisation will hamper an internal facilitators ability to challenge what is said and provoke discussion on new ways of doing things. It is far better that a facilitator is not subject to these dynamics, so if possible bring in a external facilitator.

The trick here is also to get clarity but not get stuck in a quagmire of nuance. You will need to move this conversation on relatively quickly and dig out the detail quickly. In addition, be very careful to avoid magical thinking with regard to the role technology can play in this process. Often someone will say something like "Imagine if we could automatically…" that is one of a number of indicators that there is an investment in magical thinking. Be very realistic with what technology can and can't do *now*. Do not make the mistake of including a workflow step that magically makes a lot of steps disappear through automation unless you have that technology available now and know it's limits as illustrated through use in production.

Avoiding quagmires and magical thinking is another reason why an experienced facilitator is recommended. It should take no longer than an afternoon to map out the optimised workflow. Don't worry if this is not yet 'as optimised' as you will like, plenty of time in the following processes to improve on things.

Note, you should be able to document and optimise at the same time. However, should you feel more comfortable with breaking this into a two stage process you can first document the current workflow, and then in a second session you can discuss and work on optimising it.

2. SignalPath System Design

It is hard to design software based on a typical flow diagram representation. The workflow looks horribly convoluted, linear, inflexible, and recursive processes are awfully hard to document and parse.

So, lets find another way to think about representing workflow. Let's first start by throwing away the flow diagram approach to describing workflow as the format itself forces us to think of workflow as linear, non-recursive, inflexible etc.

Instead lets consider workflow as a series of Signals, and SignalPaths.

1. a S*ignal* is a message of some kind (email, desktop notification, dashboard message) that notifies the user that something needs to be done (an operation/action).
2. a *SignalPath* is simply a link to the space where they can carry out those actions.

Let's look at how this applies with an example from the world of journals. For example, imagine a Managing Editor needs to sanity check all new submissions. They login to the system (or they might receive an email notification) and see a new submission listed. This is what I call the Signal. Contained in the Signal (email/Dashboard etc) is a link, and they click through to the submission and read through it. That link is a simple SignalPath.

So this is pretty simple. Now we simply go through every step in a workflow, as described in the section above, and map out the SignalPaths.

To do this we need to understand a final concept - the notion of spaces. Spaces are the end point or destination of the SignalPath. Since our workflows are all web based, a space can be thought about as simple a single web page. For example, a Dashboard is a single view in our browser – we call that a 'space'. A page that contains the Submission Information is also a space.

If we listed this out for any workflow, it would look something like this, formatted as **who (role)** does *what (actions)* and <u>where (which space)</u>:

1. **author** *fills out submission data* <u>on the submission form</u>
2. **managing editor** *checks submission data* <u>on the submission form</u>
3. **managing editor** *assigns handling editor* <u>from the dashboard</u>
4. ...etc

3. Mapping Workflow onto PubSweet

So, we could map this out one step after the other and describe a simple SignalPath for each step, until the whole workflow is accounted for. The problem here, is that if we were to design a system like this, there is a danger we create a whole lot of unique spaces with each step showing a Signal originating on a dashboard and then 'carrying the user' to a unique space. That's not very helpful as you would soon end up with hundreds of one-action unique spaces.

Instead, what we must do is follow some simple constraints. The basic constraints are as follows:

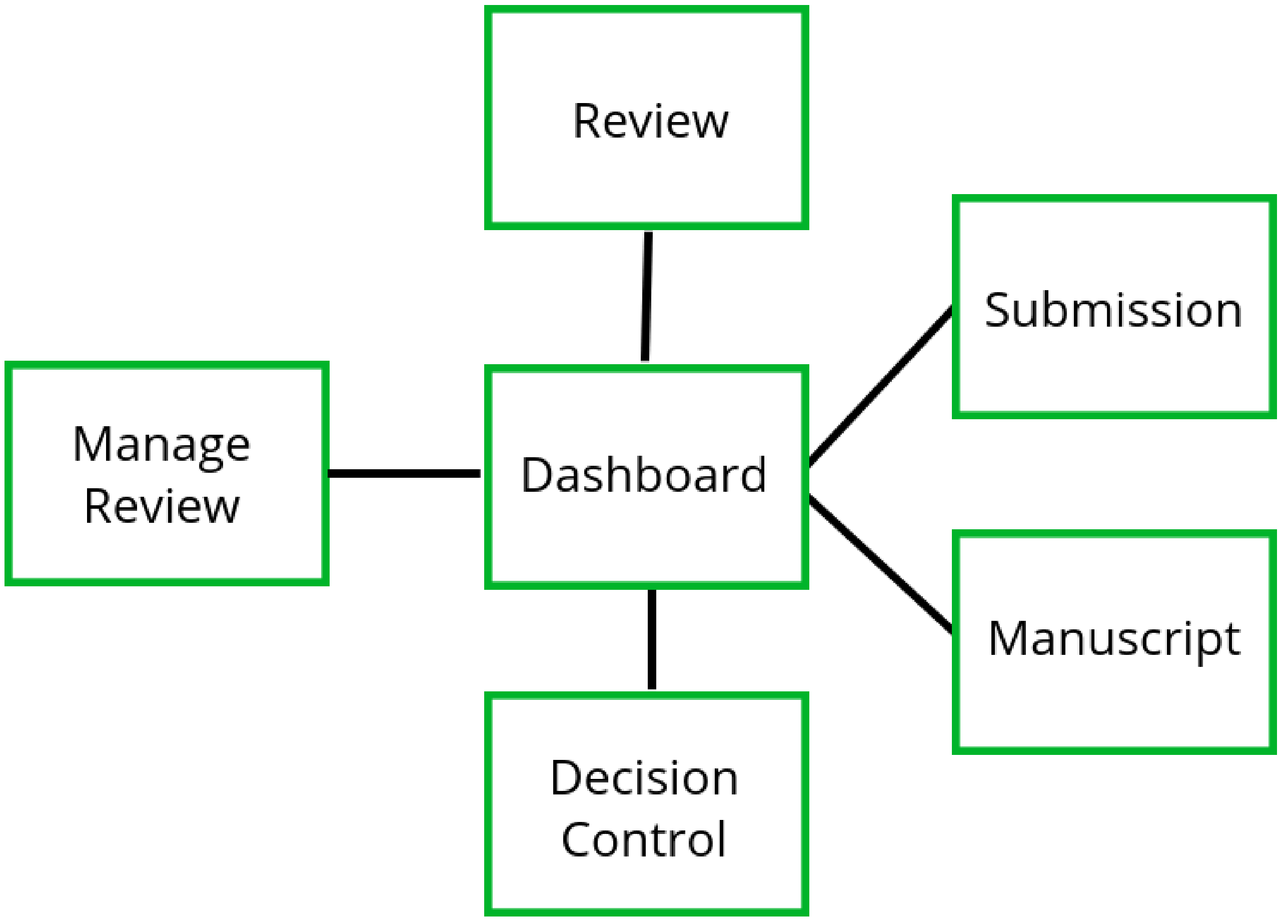
1. reuse as many spaces as possible.
2. only add a new space when we absolutely can be sure the existing spaces can't be reused.
3. Push single actions to the Dash (covered below).

So, for this example…

1. **author** *fills out submission data* on the submission form
2. **managing editor** *checks submission data* on the submission form
3. **managing editor** *assigns handling editor* from the dashboard

You will note that the first 2 steps involve doing something with submission data. So, let's just reuse that same space (Submission Form). That way we have covered two-thirds of the above 3 steps with just 1 new space (apart from the dash).

If we can do this through the entire workflow, we will, if disciplined, end up with a very simple diagram of spaces. For example, for Collabra Psychology Journal we have the following:



The above shows a ‘birds eye view’ of the space architecture for the Collabra Psychology Journal. These 6 spaces cover the entire workflow. Your goal is to design a similar view for your own workflow.

Single Actions

As mentioned above, it is also important to place a final constraint on the system design – single actions.

A single action is what it sounds like, a step in the workflow where one very simple simple action is required. These can be executed from the Dashboard, which in effect turns your Dashboard into a lightweight queue manager.

To do this we need to recognize the difference between a *single action* that could be easily executed *from the dashboard* (ie. without ‘going anywhere’) and a more complex action that requires an additional space to be executed.

In the above example for step 2 the Managing Editor can view the submission and then the only thing they need to do is assign a Handling Editor. If we know the preset list of Handling Editors (Journals always do) then we can simply choose one from a dropdown list. This is a simple enough action we can easily place it on the Dashboard.

There are some caveats to this… I would argue with the above example that this action is best placed on the Dashboard. However, that *does* mean that the Managing Editor has to follow the 'SignalPath' from Dash to Submission, read the submission data, and then 'travel back' to the Dash to execute this assignment. That isn't a terrible burden on the Managing Editor, but I can see why someone would argue that this action should instead be placed on the Submission page to simplify things and avoid this 'additional travel'. This is one of several options for solving the requirements for this step.

I can see that argument but if we do place the assignment on the Dashboard then we create very conditional interfaces that are fixed to one prescribed order of signals and is hard to change later. We cannot, for example, share the Submission Page with other roles without building in conditional logic to hide the dropdown assignment feature if the user is not a Managing Editor. That can quickly make things less reusable, hard to develop, and more difficult to change at a later date. Hence I believe we should try and avoid as much conditional logic as possible in spaces *other than the dashboard*. If we embed the conditional logic only in the dash, then we have only one place to change when we decide at a later date to further optimise the workflow.

The knock on effect of this is that each space really is a reusable operational context where actions of a certain kind take place… for example, in the Collabra spaces diagram above, we have a 'Manage Reviewer' space. All those that see this space, to avoid embedded conditional logic, should see the same thing. Whoever sees this space, sees all the tools necessary to manage a review for a specific paper. The trick is then only to enable or disable access to these spaces according to a set of predetermined attributes or criteria. If we can do that, then optimising a workflow really is a matter of re-ordering signals, and very little other system tweaking needs to be applied.

Lastly, don't put the same feature in two places. One of those places is wrong… make a decision!

As with every part of system design however, these tradeoffs are up to you to decide on. There are no hard and fast rules, just hard won lessons, some of which I am attempting to pass on now!

How to use SignalPath Design

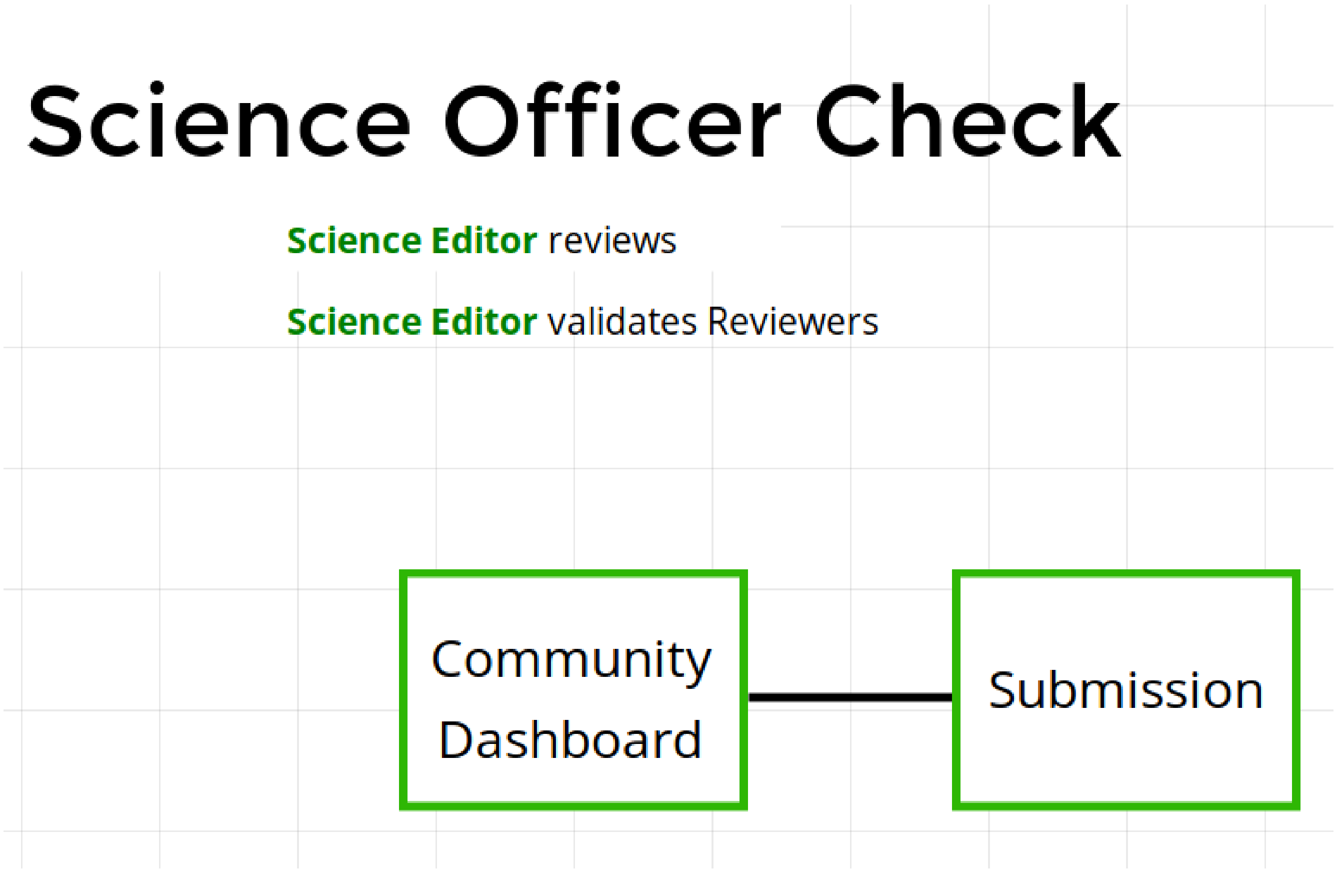
The process is actually quite simple:

1. right down a **who** *does what* and <u>where</u> order of signals, optimise as you go
2. start with a dashboard (its a handy starting point) and go through the workflow step by step
3. at each step ask yourself, is this a single action best handled on the dashboard? or do I need another space?
4. if you need another space, see if you can use an existing one. If you can, use that.
5. If you can't use an existing space create a new one

There is a bit of wrangling needed, but this a pretty effective way of capturing seemingly complex workflows in relatively simple systems, systems that can also be 'easily' optimised over time.

4. Documenting Per-Role Pathways

Now we have one simple task left to document our newly optimised worklow – drawing out per-role pathways. This is simply a process of going through each of the steps and drawing connected boxes to represent the spaces where a role must execute their actions. This can be simply illustrated. For example, here is a representation of a role based step in the Wormbase workflow:



Moving through the entire workflow like this and documenting means you also challenge some of the earlier decisions you made when optimising the workflow. Some things will become obviously too complex if a role needs to go to 'too many' spaces for any one step in a workflow. Again, this is a fuzzy line but a good system designer will work out how to juggle these trade offs and how to communicate the value of these decisions to the rest of the organisation in a way that will help them understand how the new workflow design will make everyones life easier....

Appendix 1. Wormbase Workflow

Author *starts new submission*

1. Starts new submission
2. Fills out article metadata
    1. Title
    2. Author(s)
    3. Funder(s)
    4. References
    5. Keywords
3. Adds article text
4. Uploads figure
5. Suggests Reviewers
6. Submits Manuscript

Editor *reviews new submission*

1. Reviews article content, metadata
2. Adds datatype designation
3. Sends to Author

Author *adds datatype specific metadata*

1. Adds datatype metadata
2. Adds comments to Editor
3. Re-submits Manuscript

Editor *reviews, assigns editors*

1. Adds Reviewers
    1. Accept/Reject Author suggestions
    2. Adds new suggestions
2. Adds comments for Science Officer
3. Sends to Science Officer

Science Officer *reviews, signs off on reviewers*

1. Cursory check of article
2. Approve Reject Article
3. Reviews Reviewers
    1. Accept/reject Editor suggestions
    2. Add new suggestions
4. Adds comments for Editor
5. Sends to Editor

Editor *invites reviewers*

1. Editor actions invitations to Reviewers

Reviewers *review*

1. Accept or reject invitation
2. Views article, metadata, datatype metadata, etc.
3. Adds annotations
4. Adds comments
5. Recommendation for Accept/Reject/Revise
6. Sends to Editor

Editor *reviews, modifies, makes decision*

1. Views Reviewer's review
2. Consults with the team, could return to the SO, could return to the reviewer
3. Make decision to Accept/Reject/Revise
4. Add comments for Author
5. Sends decision to Author