ITSM portal

The one-and-only Change Management process

In one of ITSM PORTAL's LinkedIn groups (ITIL& ITSM Best Practice), the issue of Change Management (CHM) versus Release Management (RM) was brought up. Some contributors in the discussion argued that these were *separated disciplines*, others argued that *the one was controlled by the other*, and I argued that they are actually *one and the same* ;-)
Here are some of the comments collected from that discussion, and my response to them.

Arguing that RM must be added to CHM, one contributor wrote "*The simplest possible reason is that without a Release and Deployment Process you cannot exercise governance over the activity*".
This would mean that CHM cannot govern changes that are bundled as releases. As you may expect, I disagree with that statement. It would be a serious disqualification of what I have learned to call "change management".

Most contributors to the discussion acknowledge to some extent that changes and releases differ. And they're right: **releases are bundled changes**. This requires special attention (e.g. CAB activities), since the planning is more complicated. But releases and changes don't differ in any other aspect, apart from their single versus composed structure: both require acceptance, planning, building, testing, accepting for roll-out into production, actual implementation, evaluation, etc.

Some argued that changes and releases should be *planned separately*.
I think that would introduce serious problems. Both are in effect "changes". Planning both in separate planning environments would cause less grip on exactly the main goal of CHM: preventing that one change would negatively influence the result of others (the production environment). That would be rather silly, wouldn't it?
Imho changes and releases should run through a shared planning, testing, and accepting phase, because both will end up in the one and only production environment.

The question therefore can be narrowed down to: **What RM activities are NOT covered in CHM?**

If you want to specify activities that are in RM but not in CHM, you must define RM and CHM very carefully, because both describe how an A changes to a B. So RM and CHM are either **redundant**, or they're **part of each other**.

- If they're **redundant**, I know why I will avoid working with the ITIL type of RM: I prefer to work with clean non-redundant processes. Just imagine: what would be the real benefit of having two redundant processes? And what would be the cost?
- If they're **not redundant**, they must together describe the transition from an A to a B. Which I used to call 'Change Management' for decades. It was only with ITIL v3 and it's "lifecycle approach" (actually PDCA) that the change management process was split up in several components. This was probably caused by the desire of giving ITIL v2's "Software Control & Distribution" a position in the framework.

In ITIL v3 and beyond, the CHM process was split up into 5 components:

- Service validation and testing
- Transition planning & support
- Release and deployment management
- Evaluation
- and..... Change management.

That's when I stepped out. This definitely didn't resemble what I considered 'best practice' any more. The v3 construction seemed to be described by consultants that used to work in very large organizations, where they would apply their standard policy of introducing as much complexity as they could. The resulting 'collection' of CHM processes wasn't applicable to the *large majority* of organizations that didn't qualify as "global IT organizations that hire the big five for their organizational change programs".

These 'normal' organizations will of course need to manage changes to their environment. And they'll need to plan the bundled release of some of these changes. They may even have a **release manager** who specifically looks at *the bundling of changes*. But they still do CHM.

It's only when they separate RM from CHM, that they will suffer loss of quality. That's when they are at risk of creating a redundant system that will be inefficient, or where governance may even fail. The **solution** lies in a much simpler construction: *determine your change management process, pay adequate attention to the planning of changes in releases (within the CHM process), and apply one and only one CHM process to all changes*.
This makes it much easier for all IT staff to get a grip on what they're doing: managing changes to the controlled environment without damaging the existing systems in production.

If you manage complex environments and you need to pay considerable attention to the planning and roll-out of bundled changes (*releases*), you may install a **Release Management Function**. This may be one individual with a release manager role, up to a team of people in various derived roles that focus on the planning and coordinating activities of bundled changes in the CHM process.
This way, you can create any local solution based on local specifications, without having to change the one-and-only change management process.

The issues discussed here are caused by the misguided idea that ITIL describes **processes**. That is not so: ITIL describes **practices**, and each books starts with that statement. And practices are the local results of an organization's work policies *that are derived from their processes*.
The issues of change management versus release management as described above can thus be solved by creating a simpler, pure and integrated process model, and creating a release management function that pays specific attention the planning of bundled changes. This is documented in Designing and Transforming IT Organizations, and in The ISM Method. Past, Present, and Future of IT Service Management, two books recently published by TSO in their International Best Practice library.