OBSOLETE OBSOLETE OBSOLETE OBSOLETE

Not relevant with revised TCT & SAT responsibilities. (2014)

OBSOLETE OBSOLETE OBSOLETE OBSOLETE

Clarification on TCT Pre- and Post-Baseline Duties

The following was extracted from email sent by the DM Project Manager to the TCT in order to clarify the role of the TCT.

> We should separately ask "when should we place the database schema (or any other major element of the DMS) under change control?" which would effectively mean that the TCT is going to control further changes to it. I think there are two answers to this depending on whether the context is a Data Challenge or the LSST Reference Design.
>
> For the LSST Reference Design, note that the DM functional requirements (FRS) ultimately drive the software and data requirements, which drive the schema including the row size and column size, which in turn drive the storage size, which in turn drives the storage cost. Since the last item is baselined for PDR, I would suggest we must consider the schema to be under change control for LSST reference design purposes. We should submit it with the final FRS and FRS summary into LSST change control in docushare prior to the PDR. There is some cleanup to be done to the FRS, summary, schema, sizing model, and infrastructure estimate to do, but that should be doable before PDR.
>
> This LSST reference design baseline will change infrequently, for example at the end of a DC when we have the results of the DC in hand, or when a significant new requirement has been added/changed in the FRS. Such changes can only occur with TCT approval and proper impact analysis on the DMS cost. Also note that if requirements changes want to change the overall cost/schedule/performance envelope of the DMS, the TCT can't approve the change, it has to go to the LSST CCB.
>
> For Data Challenge purposes, the situation is different. In the DC, we expect to evolve portions of the design as part of a DC process, and don't have to have TCT approval to do so, at least until that portion of the design is baselined for that DC specifically.
>
> That is, at the start of each DC, we take the existing reference design (including the schema) as a starting point and modify it as needed for the DC, including addition of new components and concessions made to the fact that the DC is not a full operational DMS. This can occur in a piecewise fashion, we don't need to move the entire design along at one time. However, once we decide on a given portion of the design, we should baseline it for the remainder of the DC, that is control further changes to it via the TCT.
>
> I think the DCs need a more formal design review process covering the design activities prior to baselining. In this process, which is currently done pretty informally if at all, a designer documents the design, presents it to other developers, and the group discusses it and decides whether and how to proceed.

This process does not need to be overseen by the TCT; working groups or responsible developers can do this. This can occur multiple times in a DC until that part of the design is "ready" to be baselined. It is only after that portion of the design is baselined for the DC that implementation should proceed, and only after baselining does the TCT need to get involved with subsequent changes.

This applies to the entire DC design (in the UML model the use cases, activities, domain model, schema, and descriptions of controllers on the robustness diagrams).

This is exactly analogous to a code review, but is done BEFORE the implementation is coded (although some small amount of exploratory coding can be allowed in order to frame or better understand the proposed design). Necessarily this requires a written form of the design to be available for review. Our preferred form is UML, but trac pages can suffice if there is a strong reason.

In order for this not be an huge overhead up front, there is a level of detail issue that must be addressed. That is, this process should be done with "significant" or "larger" design changes, i.e. changes to exposed interfaces, pipeline/stage/slice structure, overall workflow, selections of third party libraries, decisions to use a particular algorithm (e.g. xyz convolution) etc. It is not necessary for changes purely internal to a class or stage that don't change the overall flow or interfaces.

Relatively speaking, it is clear that much more of our effort is in implementing the detailed algorithms (the individual robustness controllers) than in the larger design. That is all the more reason to do the larger design up front, quickly, and then move on to the detailed controller design. Changes to the overall design will "bubble up" from the detailed process, but we should not operate for long periods of time without the higher level context.

```
Reference:

Date: Wed, 27 May 2009 14:28:14 -0700
Subject: Re: db schema redesign - planning
From: Jeffrey Kantor <jkantor@lsst.org>
To: Tim Axelrod <taxelrod@email.arizona.edu>,
        Robert Lupton <rhl@astro.princeton.edu>
CC: Jacek Becla <becla@slac.stanford.edu>, Tim Axelrod <taxelrod@lsst.org>,
        Gregory Dubois-Felsmann <gpdf@slac.stanford.edu>,
        Suzanne Dodd <sdodd@ipac.caltech.edu>,
        Roc Cutri <roc@ipac.caltech.edu>, Robyn Allsman <robyn@lsst.org>
```