
Case Report Form Version Management

Paulina Leszczynski, Communications Coordinator
Clinical Research Unit
Cumming School of Medicine
University of Calgary

CRU Background

- The Clinical Research Unit (CRU) is located in the Cumming School of Medicine at the University of Calgary
- Offers research data support services to University researchers
- Has always been DataFax-centric

Abstract

- CRU clients routinely request changes to existing CRF content in live studies
- DataFax is quite safe to make these changes
- However, we often found ourselves in never-ending loops of overlapping change requests
- Implementation of document change management system based on the Git distributed version control toolset

CRFs, DataFax, and a Growing Team

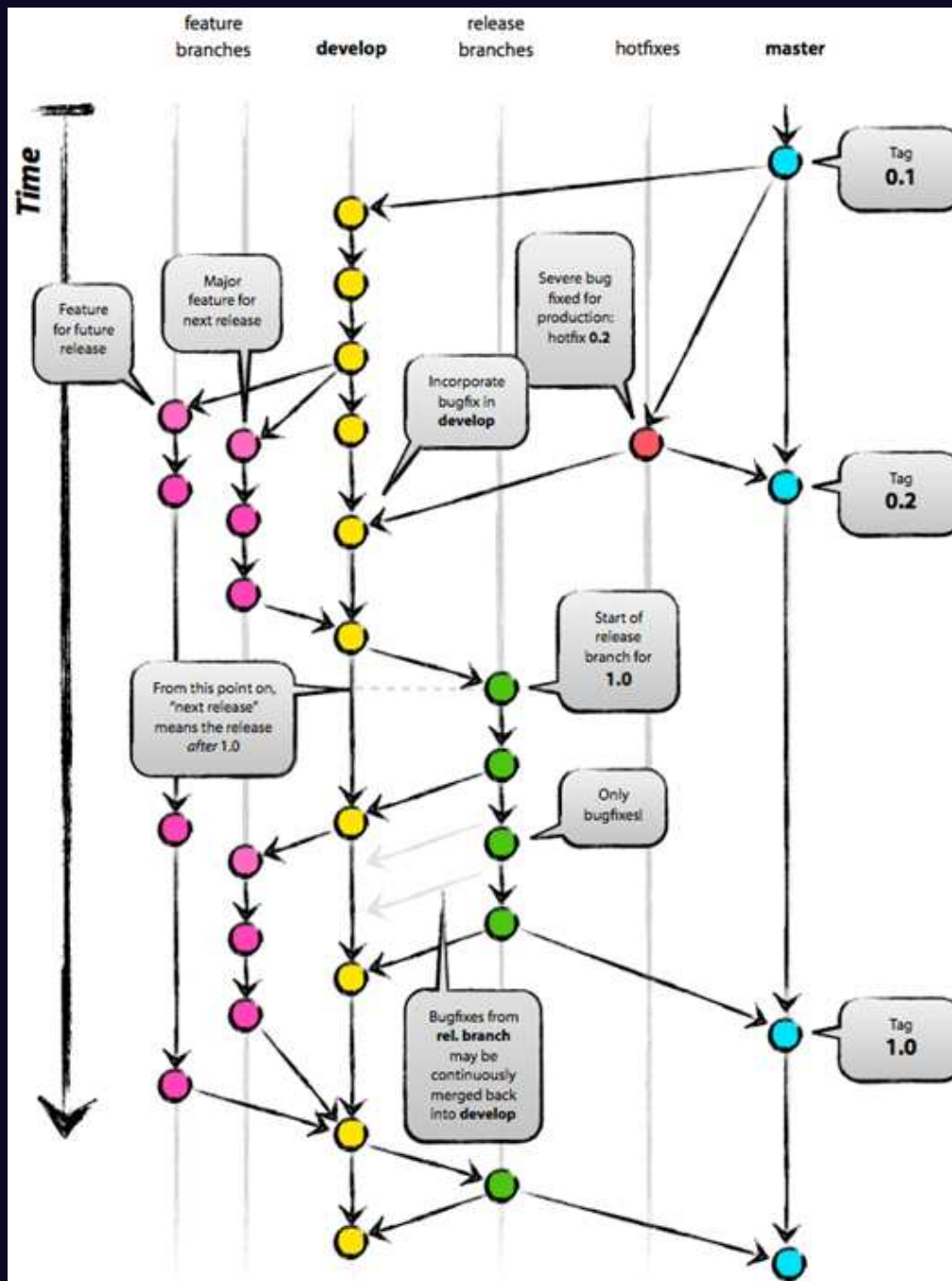
- Framemaker for first 5 years using DF 3.8.2
- Transition to InDesign since DF 4.3.0
- Change Management reared its head as we scaled our team of form developers and coordinators
- The notion of parallel development on a single CRF set drove the selection of GIT/Git-Flow as our CVS solution and branching paradigm

Change Management Workflow

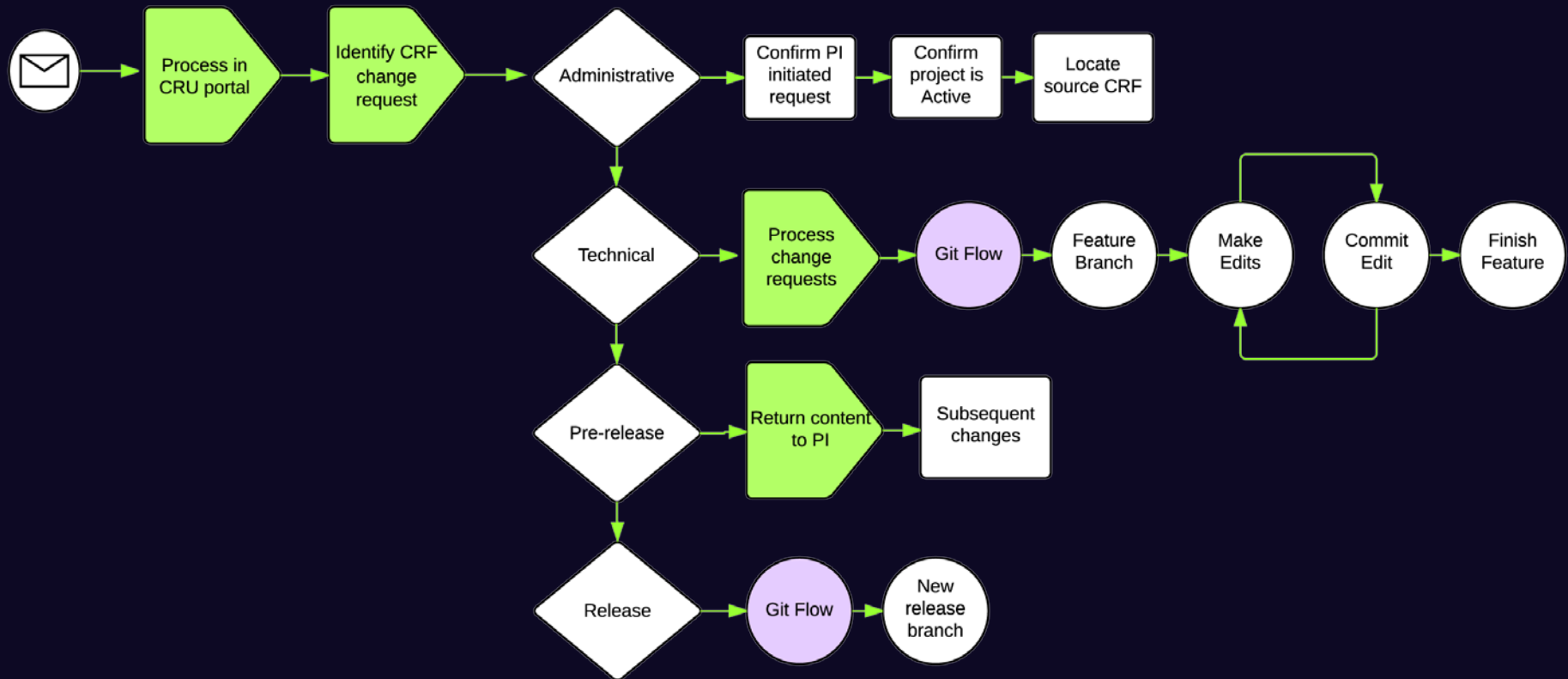
- Standard Operating Procedure-driven
- Version control and project management system integration
- Appropriate sign-off and PM processes integrated into document change process

Tools

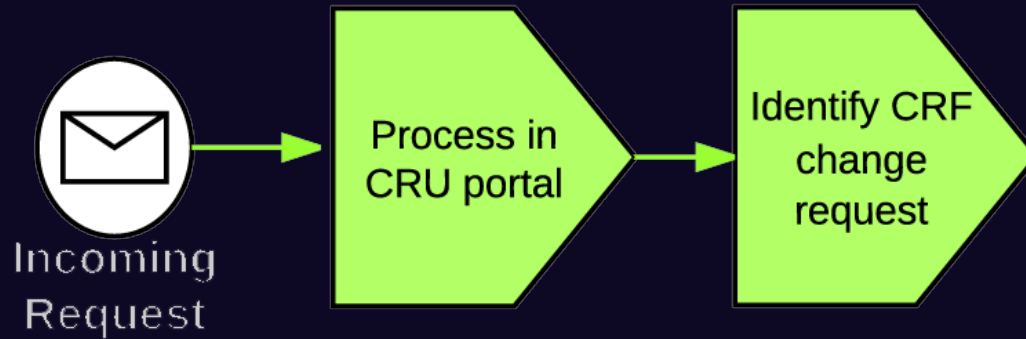
- Git Source Control Manager
 - <http://git-scm.com>
- Git Flow Branch Management Extension
 - <http://nvie.com/posts/a-successful-git-branching-model/>
- Git Lab Web-Based Repository Management Application
 - <https://about.gitlab.com/>



CRF Change Management Workflow

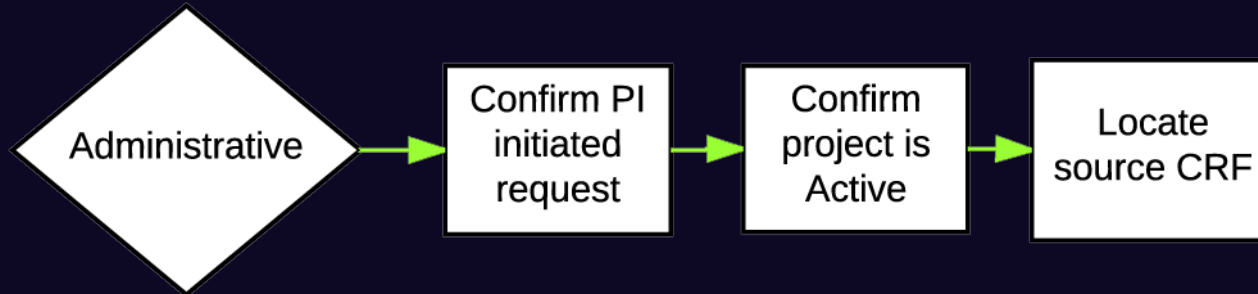


CRF Change Management



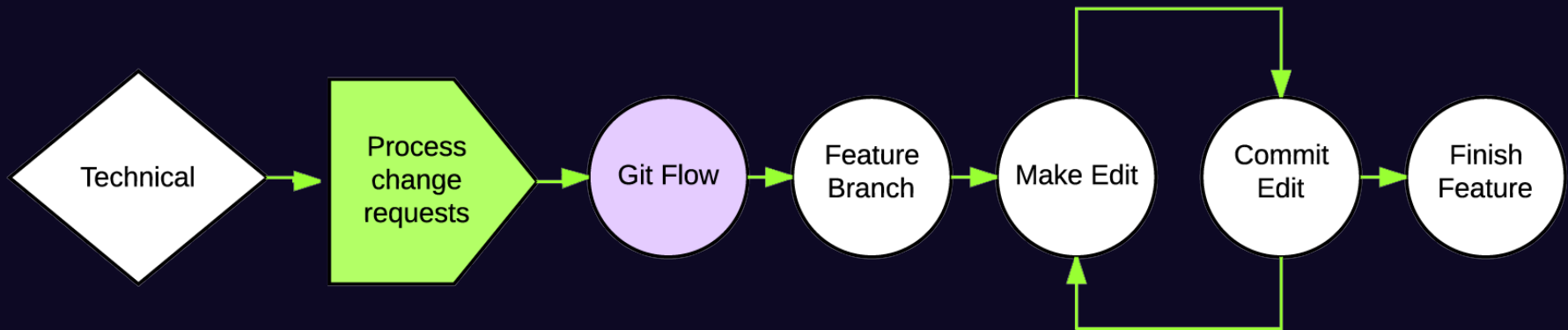
- All incoming work requests to the CRU are processed in the CRU project management portal
- Inbound requests are identified as a CRF change request

CRF Change Management



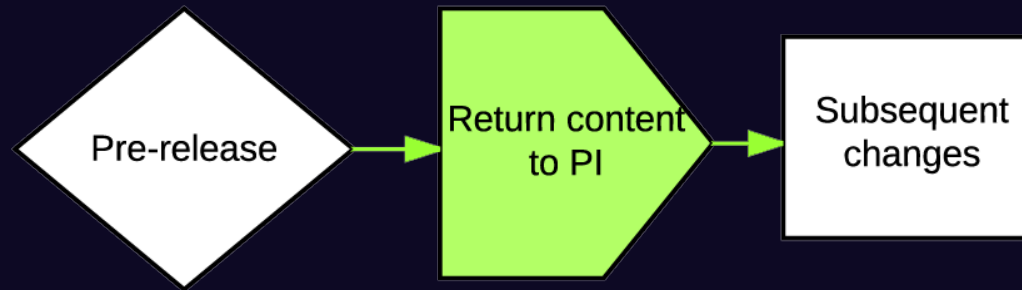
- Confirm the request initiator is identified as project PI/PI is copied on the request
- Verify with CRU project management portal that project status is Active
- Confirm project CRF source files are available in the CRU repository
- Identify most recent release

CRF Change Management



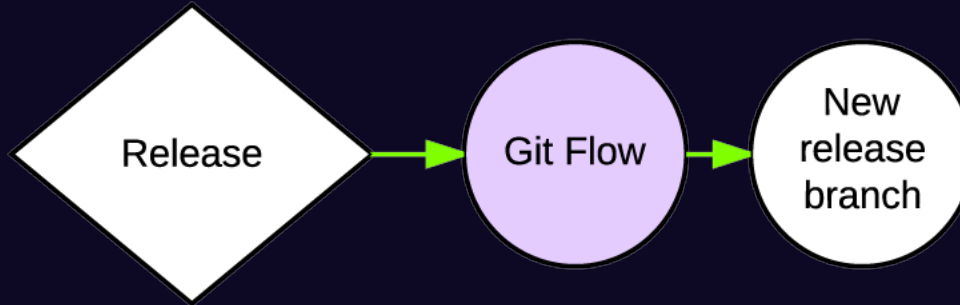
- Process change request content into single edit task list
- Using **Git Flow**, create a feature branch off of the develop branch
 - Give branch a name including the id of the initial change request
 - Make and commit each change listed in edit task list to the feature branch
 - Finish the feature branch merging content back into main development trunk

Pre-Release Review



- Provide change request initiator with request content
- Should review identify subsequent changes → entire process will start again until sign-off for release is obtained
- If changes satisfy request → initialize a release process from the development branch

Release



- Create a release branch using the appropriate numbering convention
 - n.n for non-urgent
 - n.n.n for urgent releases
- Using Git Flow, finalize the release branch
- Push master, develop and release branch changes to remote
- Delete the release branch on remote
- Push tags to the master branch of remote

Release

- A release branch is similar to a feature branch with the following exceptions:
 - Changes to content in the release branch are release specific
 - Finishing the release branch results in a merge back to develop
 - Tagged merge into the master branch
 - Content associated with tagged commits to the master branch are officially released content

Post-Release Actions

- Upload CRFs to DataFax
- Modify Setup to accommodate change that should be readily identified by working through the edit task list item by item

Other Git Thoughts

- Your team has access to the history of changes made to a set of content
- Users can cherry-pick commits
- Granularity with commits
- One true source of content
- Free

Questions?

Contact us: cru@ucalgary.ca