Initial DC3 Task List

This document represents a wish list of broad tasks to be included in the DC3 Plan. The overall goal is to extend our pipelines to support data release pipelines as well as for better coverage of the nightly processing.

Applications

Applications Framework

- clean up implementation from DC2. Maybe this is pre-DC3, but there's quite a lot of non-standard-conformant or simply messy code
- make the pipeline scripts themselves importable (and runnable!) from a python command line; v. useful in testing/debugging
- work on a more complete set of unittests. Move code from examples as appropriate, and stop running (as opposed to compiling) examples codes
- where possible, work on automated acceptance testing (within the unittest framework).
- handle exceptions consistently, passing information up the stack as appropriate
- address vw performance issues and decide whether we wish to retain it
- redo Image/Mask/MaskedImage? APIs to hide any mention of vw classes (this does not imply vw's removal). Clean up APIs and make consistent and more idiot-proof (e.g. consistent deep v. shallow copies)
- address other performance issues e.g. Kernel -- cf. #277 which still isn't merged.
- adopt use of a math library (e.g. gsl)
- adopt some version of lapack/BLAS (e.g. goto)
- adopt FFTW3 or other FFT library, and use as needed e.g. in some convolution code
- adopt statistical library (e.g. robust unbiassed clipped means)
- adopt a pixel coordinate standard for LSST
- complete implementation of DC2's detection pipeline (e.g. detecting Footprints at a point; including Peak/Source? lists in Footprints)

Data Release Pipelines

Image Coadd Pipeline

- **(AB/AC)** Image resampling and stacking
    - For pretty images
    - For deep detection
    - For diffim templates
- **(AC/AB)** Chi2 detection algorithms (RHL: does this belong here? I thought that detection's going to afw)

Deep Detection Pipeline

- **(DW/AC/AB/RHL)** Detection on a stack (if we don't use the coadd) (DW: I expect to use the coadd. In that case, is it the same cast of characters?)
- **(AB/RHL)** background estimation
- **(AB/RHL)** deblending (RHL: stretch goal?)
- **(AB/RHL)** PSF photometry
- **(DW/AB/RHL)** galaxy photometry

Photometric Calibration Pipeline

- **(TA)** develop detailed use cases for nightly iteration, exposure iteration, and object iteration
- **(TA)** perform simulations of calibration pipeline algorithms, preferably fed by CalSim?
- **(TA)** clarify the way in which time dependent photometry is measured

Astrometric Calibration Pipeline

- **(TA/AB)** Nightly pipeline (determine only low order terms)
- Archive processing (determine high order terms; out of scope for DC3)

Science Data Quality Analysis Pipeline Components

- moved to its own section

Nightly Processing

Image Processing Stages

- **(RO/AC/AB)** develop strategy for template generation
- **(NMS)** Instrument Signature Removal (ISR) stage (Ticket #327)
    - ♦ create use case diagram and add use case text to EA for nightly ISR (DONE)
    - ♦ flesh out and/or retain place holders for archive ISR use cases (eg. crosstalk, detailed flat field correction)
    - ♦ compile list of dependencies and metrics for the creation of the master calibration exposures needed by the nightly ISR (very rough use cases are in EA)
    - ♦ coordinate with Jacek's database schema effort to update precursor schema with additional metadata information needed and/or generated by the ISR
    - ♦ coordinate with Debora's DQA effort - send list of ideas for QA for nightly ISR ('stretch goal' for DC3)
    - ♦ write unit tests for each use case
- **(NMS)** work with Tim A. on data-release pipeline requirements for ISR
- **(NMS)** speed up wcsMatch, add separable kernel to options, update EA diagrams
- **(RHL/AB)** PSF generation
- **(RHL/AB)**Source characterization (RHL: how much of this belongs in deep detection? How valid is this distinction?)
    - ♦ stars
    - ♦ galaxies
- **(AB/RHL)** Basis functions and their spatial variation for DiffIm? Kernels and the PSF
- **(TA/AB)** Create robust WCS determination stage - possibly based on SCAMP

Detection Stages

- Do we do characterization of sources in DC3?

Association Pipeline

- **(SMM/KTL)** track down DC2 SQL script performance anomalies
- **(SMM/KTL)** Move spatial-match code to afw
- **(SMM/KTL)** Provide Python-level access to spatial match routines

- **(SMM/KTL)** reimplement AP in terms of slice-to-slice communication primitives (instead of using shared memory)
    - ♦ Note: depends on **Pipeline Harness extensions**
- **(SMM/KTL)** implement and examine performance of visit-level transactions
- **(SMM/KTL)** probabilistic spatial matching
- **(SMM/KTL)** use proper motions to extrapolate object positions to time-of-visit prior to matching
- **(SMM/KTL)** use stored colors for each object and the zenith angle for a visit to correct for color dependent refractive atmospheric effects

MOPS

- **(JMyers)** Discover, fix source of very large error ellipses in NightMOPS
- **(JMyers)** Integrate DayMOPS into LSST

Software Quality Control (Applications)

- **(Applications Developers)** Tests for each Class
    - ♦ Implement unittests with associated datasets for each class
        - ◊ test normal, boundary, and failure modes (aka white box testing)
        - ◊ test each class' APIs (aka black box tests).
    - ♦ Some tests may be derived from low-level use cases.
- **(Applications Developers)** Tests for each Component logically comprised of multiple classes
    - ♦ Implement unittests with associated datasets for each component
        - ◊ test normal, boundary, and failure modes (aka white box testing)
        - ◊ test each component's APIs (aka black box tests).
    - ♦ Some tests may be derived from low-level use cases.

Science Data Quality Analysis

- Exercise use case for automated component of SDQA (measure distribution of delivered PSFs for bright point sources).
- Develop prototype visual display tool for SDQA.

Science Data Quality Analysis Pipeline Components

- define architecture, identify DC3 components
- implement any pipelines necessary to support the selected use case
- work with individual pipeline developers to implement any diagnostics which need to be embedded in pipelines to support the selected use case

Data

Simulations

- **(AC)** realistic dither patterns

Preparing existing Data (CFHT-LS, TALCS)

- Tasks?

Test and Evaluation System for DC3

- **(Teresa Cottom (TC))** Configure LLNL Cluster for DC3 test environment including data preparation, database configuration and software environment.

Middleware

Pipeline Orchestration Layer

- fill out the design of the orchestration layer, identify the components for DC3
- implement an orchestration layer

Pipeline Harness extensions

- **(GD)** Complete design for data sharing among Slices
    - ♦ Support use cases for interSlice communication
        - ◊ identidy neighbors of a Slice: known in advance, or dynamic?
        - ◊ identify data types that are transmitted in each use case
        - ◊ Determine level at which communication occurs
            - · within an `AppStage`
            - · within `ShareDataStage`
            - · within lsst::dps
- **(GD)** Implement data sharing mechanisms
    - ♦ Work to support communication of general serializable types over MPI
- **(GD)** Develop mechanisms for identifying/accessing

Event Framework

- **(SRP)** provide configurable on-the fly event processing

Middleware Interface

- merge Policy and DataProperty?
- merge Trace and Logging

Database

- **(KTL)** eliminate dependency on CORAL/SEAL
- **(KTL)** regularize the storage and access to image metadata
- **(KTL)** support persistence of deep detection products
- **(JB)** extend schema to support calibration and pluggable classifiers
- **(JB)** integrate MOPS and LSST schemas
- **(JB)** add provenance capabilities, including re-creation of data products

Prototype Provenance Infrastructure

- **(Teresa Cottom (TC))** In conjunction with SLAC develop use cases and prepare design for provenance prototype
- **(TC)** In conjunction with SLAC, implement provenance within DC3 framework
- **(TC)** Prepare test plan and test provenance implementation for DC3

Software Quality Control (Middleware)

- **(Middleware Developers)** Tests for each Class
    - ♦ Implement unittests with associated datasets for each class
        - ◊ test normal, boundary, and failure modes (aka white box testing)
        - ◊ test each class' APIs (aka black box tests).
    - ♦ Some tests may be derived from low-level use cases.
- **(Middleware Developers)** Tests for each Component logically comprised of multiple classes
    - ♦ Implement unittests with associated datasets for each component
        - ◊ test normal, boundary, and failure modes (aka white box testing)
        - ◊ test each component's APIs (aka black box tests).
    - ♦ Some tests may be derived from low-level use cases.

Build System

- **(DG,RL,RP)** convert to new package build mechanism (remove dependency on pacman)
- **(DG)** update software distribution server
- **(DG)** enable support for 64-bit Linux
- **(BB)** support distributable VMs for new developers
- **(BB)** deploy automated, multi-platform testing system
- **(KTL)** create better templates for SConstruct, SConscript, and Swig .i files
- **(DG)** implement Software Quality Assurance procedures relevant to build process
    - ♦ within Ticket branch development
        - ◊ build warnings if coding standards check fails
        - ◊ build warnings if unittest doesn't satisfy DC3 coverage requirement
    - ♦ within Trunk branch development (generally merge from Ticket branch)
        - ◊ build failure if unittest doesn't satisfy DC3 coverage requirement
        - ◊ build failure if coding standards check fails
    - ♦ Nightly build of development trunk
        - ◊ any failures cause automatic 'Failed Nightly Build' ticket at high priority
    - ♦ Production Release sanity check
        - ◊ any failure causes automatic 'Failed Production sanity check' ticket at high priority
        - ◊ auxiliary automatic notification to highest authority on failure (separate from ticket notification)
- **(RA/DG)** Configuration Management: 'Build' upgrade
    - ♦ Standards: specify requirement to version and archive build's provenance;
    - ♦ Configuration Management: define procedure to satisfy that requirement;
    - ♦ Configuration Management: implement its instantiation.

Software Quality Assurance

- **(RA)** Software Quality Assurance: Planning

- Develop Framework of LSST DM Software Development Plans
    - Software Quality Assurance Plan
    - Verification and Validation Plan
    - Configuration Management Plan
- Develop Data Challenge 3 specialization of those SQA plans

- **(RA)** Verification and Validation: Acquire SQA support tools
    - configurable tool to perform Lsst software standards checking
    - coverage analysis tool

- **(RA)** Verification and Validation: Tracking Requirement to Tested Product Release
    - Standards: develop standard labelling scheme for products and their provenance.
    - V and V: define the procedure whereby the initial requirements are traced to the validated product

- **(RA)** Verification and Validation: Increased Validation Testing
    - Standards: specify the testing requirements for each phase of Product development:
        - unit test: black box, white box, performance tests
        - integration tests: black box, white box, performance, regression tests
        - system tests: function, performance, interface, resource, security, portability, reliability, maintainability, safety
        - acceptance tests: capability and constraint tests
    - Standards: specify the pass/fail metrics for each test type (eg compliance levels)
    - V and V: Refine the workflow process to incorporate additional V & V testing.