Model Catalog TODO

DCA

Not in order

Remember to:

* Pull from remote repo to local (hard drive)
* Modify
* Push from local to remote
* Only once modification and testing are complete (*once in a release state*): copy from local to ccsp for use by others

Get Andrew to create a new TEST.MC db to keep developing on. We want to be able to map from what we currently have on test but need to keep developing as well. – ***checking into option to just save out static copy to gdb for our current mapping. Check with BW on Taggart status. If we can keep things static for a while then just save out a copy and map from that if needed.***

***OR – can we go ahead and move what we have currently in TEST to PROD?***

If implemented then need to add 3rd option for append output location: in addition to existing TEST/PROD we’d need like a TEST2 – can’t remember if this is in pyt or config

* *Made a copy of MC in CCSP to map from (has the 3 char basins) and we can continue to throw in/take out data from TEST.MC for testing*

Account for changes to results.gdb that were made. Add required fields in MC table(s) and pipe it through the process – we know of LinkResults.MaxSurchargeUpstreamFt, anything else to include and pipe to the product? When we do include still need to account for 31 char issue (sde restriction) by truncating field(s). Watch out for situations where truncation results in fields with the same name and account for if an issue.

* *Added LinkResults*.*MaxSurchargeUpstreamFt and added Node\_Flooding\_Results.operation, construct\_method, include\_in\_cost and include\_in\_nbcr. Don*’*t know if these are needed right now but does not cost anything to include. Arnel has been asked to modify EMGAATS fields to remove 31 char issue. Only one field has that issue currently.*

**Add components to account for Alternative models – right now we probably cannot register them or it would act in a way we don’t want**

Add same required storms as Characterization to Alternatives required storms in required storms table

modified – per BW we need 2 EX, 5 EX, 10 EX, 25 EX, 4S6 EX and 25 FU

Clean up references in code to the RRAD, Rehab and RRAD Mapping Snapshot which are no longer being used – right now NOTHING goes to the RRAD (verify, of course). This includes references in config.

Deleting:

* Rrad.py
* Rrad\_db\_data\_io.py
* test\_rrad.py
* test\_rrad\_db\_data\_io.py
* rehab.py
* test\_rehab.py
* rrad\_mapping.py
* rrad\_mapping\_db\_data\_io.py

-  mapping\_snapshot.py

-  mapping\_snapshot\_exception.py

-  mapping\_snapshot\_dataio.py

- mapping\_area\_data\_io -- done

- mapping\_link\_data\_io -- done

- mapping\_node\_data\_io -- done

- mapping\_link -- done

- test\_mapping\_link -- done

- mapping\_node -- done

- mapping\_area – done

- test\_rrad\_mapping – done

- test\_area— done

- rehab\_data\_io -- done

* test\_mapping\_snapshot.py
* test\_rehab\_data\_io.py
* Test\_Rehab\_data\_io\_integration\_tests.py
* Config – all references to RRAD and Rehab and rrad mapping snapshot
* Existing table structure in the RRAD database – leave actual database (container) for now
  + Cleared all records for RRAD, RRAD\_MAPPING and set IDs to 0.
* KEEP but **rename**: Test\_RRAD\_capacity\_results\_integration\_tests.py (to ModelCatalog instead of RRAD)
* KEEP test\_emgaats\_registration\_integration\_test.py
* mapping\_snapshot\_data\_io.py
* Test\_characterization\_reporting\_integration\_tests

test\_simulation and test\_model\_data\_io still have failing tests related to storm names with a D – not sure how to fix test for these

run a model through (Char fine) to make sure removing all of these has not broken any core functionality

\*\*\*Modify code to account for Alternatives mapping – this was possibly stubbed out by Dan/Brent earlier but we did not get very far

Alt model must have parent model id of the parent Characterization model

Get alternative model naming conventions from BW & modify code to enforce (we already enforce for Capacity models)

add model.valid\_alternative\_simulations (very similar to valid\_calibration\_simulations) and piece in model.valid

* *We don’t care about this because we don’t enforce model names (we enforce sim names). They point to whatever folder they want and it can be called whatever they want. Its up to them to enforce*

\*\*\*Run integration test with Alternative model to test

* Add: Test\_emgaats\_registration\_integration\_tests.test\_model\_registration\_with\_model\_status\_final\_model\_purpose\_alternative\_add\_model\_and\_results\_to\_catalog

\*\*\*Run Alternative model through tool

* Test models coming from Karen soon

\*\*Create script/ module to automate copy of data for the Sitka data load. Its an easy process to do manually but has many layers and is easy to do something like get a name wrong which breaks things.

* Input data sources will be based on an json file and read into a dictionary. Provides easy maintenance for non-code types and allows us to iterate through inputs in copy method.
* *Script is functional – takes a bit more than 10 min to copy (currently) 47 data inputs*

create tests – *although job is working, I’m* *not having luck with tests for the methods I created – need to sit down with Sam/ Brent and get these figured out*

relocate/refactor pieces

rework input so that it is also json (dict input instead of converting xls to dict)

validate input names against names in appsettings json

move process to besapp4 (“the app server”)

per Sam (he has migrated methods to his own service which will be incorporated by sitka) remove the DME mst hybrid portion of data load

Add storage tables to the Model Catalog for green streets areas.

Add table in model catalog – include model id, node id and area\_sqft

get append permissions set on this table

* *Note - Open ArcCatalog. Connect to the geodatabase using the “GIS” account.*

*Right click on the Feature Class or Table.  > Manage > Permissions.*

*Click on the “Add Button”.*

*Choose the User/Role. In this case it’s “Catalog\_Editor”.*

*Click on the desired permissions. (Select, Insert, Update, Delete)*

Add item in current id table

add new storage class

adapt existing code from other modules – this doesn’t really do anything new; we’re just copying and appending. Can prob use similar code to that used for geometry..

add tests for any new code pieces – *model\_data\_io.append\_storage\_table is based on similar pieces in model\_data\_io.append\_model\_network (for nodes, links, areas) which were not tested. Similarly, model\_catalog\_db\_data\_io.add\_model calls append\_model\_network (and now calls append\_storage\_table) but this was also never tested. I don’t know if there was a reason or if it just never happened but since these were not tested I’m not going to test these new pieces either.*

run registration tool – test that storages records are being appended

A few registrations failed but it does not seem like the ModelTracking records rolled back – *this is because versioning got removed when I replaced tables – duh- fixed*

add Storages to ETL loader

Switch all Storages references to “Storage” (singular) in the code to be consistent with other object types

Add Director to Model Catalog (tables already exist but need to plumb it through – use Storages as a template)

Add item in current id table

add new director class

import director class in model data io

create append director table function in model data io

add this to model catalog db data io.add model

add tests for any new code pieces – if possible/ time allows

run registration tool – test that Director records are being appended

* + Will have to whipe MC (again) since we’re also testing in it, plus Kevin & Karen want to re-register stuff, again

copy tool from local clone to CCSP model catalog tool

add Directors to ETL loader

OutVolumeCuFt added NodeResults in Results.gdb (Arnel)

add same field to ModelCatalog NodeResults so that value caries over. Field is a double, precision – 0, scale – 0.

Need to include Model Tracking table (from MC) in load gdb so that Alternatives can be connected up with Characterization models in future versions. (DCA – need a bit more clarity on how this is used but can just put it in load gdb for now) Also add ref in SitkaDataSource.xls.

Finish plumbing through the fields in Model Catalog that are filled based on EMGAATS config (deploy date, run date, extract date)

* Per Arnel: run date and extract are not yet available in config file but will be – they are in the queue

Added model\_data\_io.read\_root\_from\_config\_file

Added piece to read extract date

*Arnel has added pieces for sim deploy and extract history in the EMGAATS config file – add piece to read and write to model catalog.*

* [*#747*](https://github.com/PDXBES/EMGAATS/issues/747)*: Add deployment and results extraction date to config files*

Added piece to read deploy date

Added piece to read run date

Added model.set\_extract\_date which is referenced in pyt

Created tests for these

Create tool/ form to make a copy of the model dir for a specified model, delete registration file and make set dir to have read/write permissions

* Read existing models and give user a pull down choice
* Get (from model) the model dir of specified model
* Copy dir into same location (one folder up) of existing model dir
  + Will be called “same name”\_copy\_YYYYMMDD
* Change permissions of copy
* Delete registration file of copy

Create tool to select which model(s) get sent to the process (the PipXP, Cost Estimator, RUL, BPW, etc process) [Existing mapping snapshot pyt *could* be modified for this] – *component to select model and associated parts by input ID(s) can be applied to both this tool and the tool to delete selected models – they can share those methods*

* This has been created as a pyt/ arctoolbox tool
* Outputs both a gdb and zipped version of same gdb (zipped version required for sitka input)

Add table/module to track models extracted for mapping (closely related to the above task). That way we know what has been used out of the MC, when, how many times, etc. This is will be connected to the tool to select model(s) out.

* If we set this up (to me) it basically puts the nail in the coffin for the RRAD db. Remaining question – do we retain the snapshot data somewhere and where? We obviously use it for mapping/reporting but then what becomes of it?

Arnel has added fields to EMGAATS for tracking Alternatives info : Operation(domain) , Construction Method(domain), nBCR(YN), cost estimates (YN) – what accommodations need to be made in Model Catalog to account for these fields? *Done, these are in the node, link, area geometry results in MC.*

Cleanup master\_DME\_hybrid (and other) to use Views instead of hard coded queries

* *This has become irrelevant – DME master hybrid (both “ways” – with DME as base or with master data as the base) is now a service created by Sam, to be used by sitka*

ETL Loader

modify so that the process requires all sources from the list be in the appsettings but that any ‘extras’ in the appsettings will pass through without throwing an error

* *Loader process is now established on besapp4*
* **Other/ Future**

Remove RRAD from general CCSP process diagram for BW as its function is ill defined at this point. Current vision is that full risk process gets re-run for each snaphot. A snapshot is basically a package of capacity (models/sims) and structural (Hansen, etc) paired with process/criteria assumptions about risk. We basically cannot compare back across previous snapshots because data/criteria will change.

Add tool to do model level cleanup (not a public tool) – remove all related records in all tables (will test – not all tables will have associated model records) for a specified ID/ IDs – *same base functionality as ‘Create tool to select which model(s) get sent to the process’, above*

**This will basically do what model.add\_model does but deletes instead of appends.**

in model\_catalog (for each item with a model ID) – use modelcatalogdbdataio (ie dbdataio).copy to memory with id filter

for simulations – do same as item 1 but then get list of sim IDs (search cursor)

use sim ID list as input for same query as item 1 but with simulation related items

include logic to delete associate downstream models as well (avoid orphaned records) – for example: if a calibration model is to be deleted then any associated characterization, alternatives, etc should also be deleted.

note – make query table (used in copy to memory with id filter) does not error if ID list is invalid, it completes and just includes the valid ones. This means we need to test that the in memory IDs match with the ones from the input list before deleting the records

all of this ***needs to be done in an edit session*** – roll back all deletes if any of them fail – we don’t want some records deleted and others remaining

create table and methods to track metadata about the delete – at least basics of model ID, who and when

Clean up CCSP mapping directory and enforce structure

Figure out ArcPro project mapping file structure

Figure out mapping method(s)

* As of right now spooling will not work since the model does not know its sewer basin and vice versa. This means a lot of manual mapping (eventually ~ a half dozen map types per basin with lets just say 80 basins (we can largely ignore sanitary only parts of basins). This is for just one type (like characterization) so there could be something like 1000 maps in total if these assumptions hold.
* *Update for spooling* (1/6/2021) – it is working to spool through basins and use a dynamic mask but it is not perfect. Basin boundaries don’t necessarily match with model results some other basin results can “poke out” from under the mask.
  + an alternate (per Sam) would be to spool using the Model Tracking dissolved lines for each model registration, spooling on model ID and turning off the features that <> the current ID
* I’m talking with BW and SR about web maps in general and for TMSs and doing some exploration. I’ve had some hangups with publishing services which hopefully can be resolved by Andrew/CGIS.
  + Received tentative go ahead for web mapping from Shannon
  + Check back with CGIS on publishing status so I can move forward with testing
* *I was able to publish all CCSP layers to Portal.* ***Some hangups: cannot enforce feature level drawing so the “nested” nature of BSBR is problematic unless we flatten that result, cannot use graduated colors with polygons (nodes, links seem to work ok with this though).***

Check off which layers worked as services and which did not – verify what I needed to make each work

Set up demo web app to show BW and Shannon what this can look like (multiple maps – may need a CCSP group created which I’ve asked for)

redo web maps once symbology is settled – work with Brent to pilot these

then get setup with Andrew in “production” method (need to sync with network server data, not just create static snapshot services)

* **Sitka stuff**

Sitka is using the Feb 14, 2020 input file still, for now, so that they don’t have to deal with any actual changes OR things like accidental renaming issues

for DCA testing – get their current version to run vs the 20200214 version for now (this is what Corina is doing) and at some point we’ll shift them to the newer load gdb

get instructions from Sam for running Sitka modules and required installs – install in VM and run batch file in VM to launch pycharm – want to get set up end of week 4/10.