How to make ugrid and gridspec more similar

The aim of this document is not to make ugrid and gridspec identical, although this might be possible. The aim is to make these two proposed conventions more similar, because I think that a similar structure will make the CF standard overall more coherent and easy to use. It is potentially confusing to adopt different approaches for similar tasks. The similarity between ugrid and gridspec is that in both of them the complete space is distributed among several index spaces, which are each self-contained, but have specified points of contact. Thus, the ugrid combined mesh, comprising several meshes, which Bert calls a "mosaic of meshes", is a conceptually similar idea to a gridspec mosaic of tiles.

ugrid

ugrid resembles the grid_mapping convention of CF, in that the mesh is defined by a "container" variable (the mesh_topology variable) that has no data and serves as a point of attachment for attributes that point to the definition of the mesh. ugrid provides also for the definition of a combined mesh

A combined mesh is defined by another "container" variable, which associates the meshes. It identifies the individual meshes by the names of their mesh_topology variables. Bert expects them all to be in one file, so they must have different names. However, they could equally well be in several files, and the container variable for the combined mesh might be in its own file or in the same file as one or more of the constituent mesh_topology variables. (E.g. in the example below, the variables CombinedMesh, Mesh1 and Mesh2 could be all in one file, each in a different file, or any other possibility.) Some other software would have to know that the several files comprised one dataset, but this is not a problem unique to ugrid; it arises in many other CF applications. Spreading the meshes over several files resembles the approach of gridspec. Earlier, I proposed that a mesh name could be used in ugrid, like the tile name of gridspec, but now I don't think that's needed.

The following example is a reduced version of Bert's, to illustrate the combined mesh. I'd like to propose some minor changes to ugrid, namely:

- Define the mesh functions with a cf_role rather than a standard_name attribute.
- Use a different cf_role for the combined mesh, not mesh_topology like the individual meshes, because this is a different sort of container variable, which has different attributes and serves a different purpose. How about parent_mesh_topology, since you already have a parent_mesh attribute?
- The mesh_topology_contact variable has two attributes contact_meshes and contact_type, which must correspond, I presume. That is, your example

  ```
  Contact1:contact_meshes = "Mesh1 Mesh2" ;
  Contact1:contact_type = "node face" ;
  ```

  implies that the variable indicates a contact between nodes of Mesh1 and faces of Mesh2. Instead of having two attributes which might accidentally be inconsistent in their order, I would propose a single attribute of CF-like format, as in the example below.
- Rename the sub_meshes attribute of the combined mesh topology variable as just "meshes". This is simply the plural of the "mesh" attribute which data variables have, and that seems logical since it supplies a list of meshes. I make this suggestion just for simplicity.
- Rename the dimension attribute as topology_dimension. I think Alex suggested this. It could avoid confusion with spatial dimensions.

My changes from Bert's document are in red. Also, I have not used the location index set, to make it simpler.

```
dimensions:
        nContact1 = 1 ;
        Two = 2 ;
        nMesh1_node = 3 ;
        nMesh2_face = 2 ;
variables:
// Topology of the combined mesh
        integer CombinedMesh ;
                CombinedMesh:cf_role = "parent_mesh_topology" ;
                CombinedMesh:long_name = "Topology data of CombinedMesh" ;
                CombinedMesh:meshes = "Mesh1 Mesh2" ;
                CombinedMesh:mesh_contacts = "Contact1" ;
        integer Contact1(nContact1, Two)
                Contact1:cf_role = "mesh_topology_contact" ;
```

```
                Contact1:contact = "Mesh1: node Mesh2: face" ;
                Contact1:start_index = 1 ;
// Topology of 1D network
        integer Mesh1 ;
                Mesh1:cf_role = "mesh_topology" ;
                Mesh1:long_name = "Topology data of Mesh1" ;
                Mesh1:topology_dimension = 1 ;
                Mesh1:node_coordinates = "Mesh1_node_x Mesh1_node_y" ;
                Mesh1:edge_coordinates = "Mesh1_edge_x Mesh1_edge_y" ;
                Mesh1:edge_node_connectivity = "Mesh1_edge_nodes" ;
                Mesh1:parent_mesh = "CombinedMesh" ;
// Topology of 2D mesh
        integer Mesh2 ;
                Mesh2:cf_role = "mesh_topology" ;
                Mesh2:long_name = "Topology data of Mesh2" ;
                Mesh2:topology_dimension = 2 ;
                Mesh2:node_coordinates = "Mesh2_node_x Mesh2_node_y" ;
                Mesh2:face_node_connectivity = "Mesh2_face_nodes" ;
                Mesh2:parent_mesh = "CombinedMesh" ;
// Data on Mesh1
        double Mesh1_zwl(time, nMesh1_node) ;
                Mesh1_zwl:standard_name = "sea_surface_height_above_geoid" ;
                Mesh1_zwl:units = "m" ;
                Mesh1_zwl:mesh = "Mesh2"
                Mesh1_zwl:coordinates = "Mesh1_node_x Mesh1_node_y" ;
// Data on Mesh2
        double Mesh2_zwl(time, nMesh2_face) ;
                Mesh2_zwl:standard_name = "sea_surface_height_above_geoid" ;
                Mesh2_zwl:units = "m" ;
                Mesh2_zwl:mesh = "Mesh2";
                Mesh2_zwl:location = "face" ;
                Mesh2_zwl:coordinates = "Mesh2_face_x Mesh2_face_y" ;
```

Bert writes that the mesh_topology_contact variable (Contact1) "contains a listing of the pairs of elements that (partially) coindice; each pair consists of an element index in the first mesh topology and an element index in the second mesh topology." This is similar in purpose to gridspec contacts, which are strings, specifying ranges of indices that coincide in the two tiles. However ugrid is less informative, because the range of partial overlap is not described. I assume this vagueness is unavoidable because of the unstructured grids involved.

It is still unclear to me what the cell_methods entry should be for data on meshes. CF now recommends cell_methods should be included for all dimensions of the data. *Are the existing methods in Appendix E adequate to describe data on meshes? What should the default interpretation be (if the recommendation to be explicit is not followed)?*

gridspec

Gridspec (M-SPEC) envisages a mosaic of tiles, in which each tile has its own index space, and the tiles have specified contacts along edges. The current gridspec proposal expects that each tile is in a separate file (the "data file"), and there is another file (the "mosaic file") which contains the information about connectivity. It is not possible for more than one tile to be stored in a given host file, because the data variables are associated with the tile only because they are in that tile's host file. It is also assumed that the data variables on the various tiles which together comprise a data mosaic will all have the same name, as there is no other indication that they belong together. Likewise it is assumed that corresponding coordinate variables on the tiles will have the same names.

I think these restrictions could be removed by making gridspec resemble the ugrid combined mesh in its use of container variables. I suspect it could sometimes be convenient for tiles and mosaics to be in the same file. Also, there might be more than one mosaic. Since there is no formal arrangement for grid staggering in gridspec, I suppose that the T and uv grids (for example, in Arakawa B) will be described as separate mosaics. I think it would be inconvenient for data on these grids, for a given tile, to have to be in different files. The use of container variables would also be more CF-like, in that it would resemble grid_mapping, and would make it less dependent on files and global attributes. CF is mostly focussed on data variables, taking the view that files should not be important.

For reference, here is Alex's example of a mosaic file, to which I have added data files. Click here if you want to open this example in a separate browser window or tab in order to compare it with my reworked version further down. The example describes two 2D tiles, which have names "left" and "right", that are in contact along an edge which has x-index 2 and y-index 0:3 in the left tile, and x-index 3 and y-index 0:4 and in the right tile. Note that in the contact_map, the y index is first, x second (in general the dimensions are given in C/CDL order for the contact).

Mosaic file, which is called "mosaic.nc":

```
dimensions:
```

```
  ndims = 2 ;
  nstring = 256 ;
  ntiles = 2 ;
  ncontacts = 1 ;
variables:
  char coordinate_names(ndims, nstring) ;
    coordinate_names:gridspec_type_name = "gridspec_coordinate_names" ;
  char tile_names(ntiles, nstring) ;
    tile_names:gridspec_type_name = "gridspec_tile_names" ;
  char tile_contacts(ncontacts, nstring) ;
    tile_contacts:gridspec_type_name = "gridspec_tile_contacts" ;
  char contact_map(ncontacts, nstring) ;
    contact_map:gridspec_type_name = "gridspec_contact_map" ;
// global attributes:
  :gridspec_file_type = "mosaic_file" ;
data:
  coordinate_names = "x", "y" ;
  tile_names = "left", "right" ;
  tile_contacts = "left | right" ;
  contact_map = "0:3 2:2 | 0:4 3:3" ;
```

Data file for the "left" tile:

```
dimensions:
  nstring = 256 ;
  x=4;
  y=3;
variables:
  float x(x);
    x:standard_name="longitude";
    x:units="degrees_east";
  float y(y):
    y:standard_name="latitude";
    y:units="degrees_north";
  float zwl(y,x);
    zwl:standard_name = "sea_surface_height_above_geoid" ;
    zwl:units = "m" ;
// global attributes:
  :gridspec_file_type = "data_file" ; // I am not sure what value this should have---Alex?
  :gridspec_tile_name="left";
data:
  x=0, 10, 20;
  y=0, 4, 8, 12;
```

Data file for the "right" tile:

```
dimensions:
  nstring = 256 ;
  x=8;
  y=5;
variables:
  float x(x);
    x:standard_name="longitude";
    x:units="degrees_east";
  float y(y):
    y:standard_name="latitude";
    y:units="degrees_north";
  float zwl(y,x);
    zwl:standard_name = "sea_surface_height_above_geoid" ;
    zwl:units = "m" ;
// global attributes:
  :gridspec_file_type = "data_file" ; // ?
  :gridspec_tile_name="right";
data:
  x=35, 30, 25, 20, 15;
  y=0, 3, 6, 9, 12, 15, 18, 21;
```

As you can see, y-index 0:3 on the left tile and 0:4 on the right tile both span the latitude range 0-12 degrees_north, while x-index 2 on the left and 3 on the right is longitude 20 degrees_east.

Alex, the gridspec proposal appears to suggest that the data files should have a gridspec_tile_names variable as well as a gridspec_tile_name global attribute. Why is that? Alex replies that the global attribute is the current proposal; it replaces the variable.

I would propose the following changes to gridspec to make it more like ugrid and grid_mapping:

- Introduce a container variable identified by a cf_role of gridspec_mosaic, with two mandatory attributes, namely tiles and

tile_contacts, analogous to meshes and mesh_contacts of the ugrid parent mesh topology container variable.

- The tiles attribute replaces the gridspec_tile_names variable. This lists the tiles which belong to the mosaic. It is included for convenience, and particularly for the sake of isolated tiles, which will not be named in the tile_contacts.
- The tile_contacts attribute replaces the gridspec_tile_contacts and gridspec_contact_map variables. These latter two existing variables have elements which must correspond. Putting both together avoids the possibility of them accidentally not corresponding. The proposed syntax of the attribute is "tilename spec | tilename spec", where each spec has the format "dimname bound : bound [dimname bound : bound ...]" and the dimensions correspond in order in the two specs. The purpose of naming the dimensions is twofold. First, it avoids replying on a particular order, and hence reduces the possibility of mistakes. Second, it means that the order of dimensions is still known even if there is no data variable in the dataset, which would be the case if the intention of the file was simply to describe a grid, but not to store data. (This is the point raised by Bert.) *I wonder if the colons are really needed. What do you think, Alex?*
- Drop the information contained in gridspec_coordinate_names, which is a list of the coordinates involved. I am tentative about this, because I have not understood its purpose. *Don't the data variables themselves identify their coordinates in the usual way, through the names of their dimensions?*
- To each data and coordinate variable, add a tile attribute, which names the tile to which the variable belongs, and a mosaic attribute, which names the mosaic container variable. The former is analogous to the mesh attribute of ugrid. The latter does not have an analogy in ugrid; ugrid data variables do not point to the parent mesh because (a) there might not be one and (b) the mesh variables point to the parent mesh if there is one. The analogy is incomplete because individual tiles do not need container variables to describe their topology, since they are always logically rectilinear.

Following these changes, the example looks like this, where the tiles and mosaic are now in the same file. To allow that, the variables on the tiles no longer have the same names, of course.

```
dimensions:
  nstring=256;
  ncontacts=1;
  xleft=4;
  yleft=3;
  xright=8;
  yright=5;
variables:
  char gridspec; // container variable of arbitrary name
    gridspec:cf_role="gridspec_mosaic";
    gridspec:tile_contacts="tile_contacts"; // identifies the contacts, like the mesh_contacts attribute in ugrid
    gridspec:tiles="left right"; // identifies the tiles, like the meshes attribute in ugrid
  char tile_contacts(ncontacts,nstring);
    tile_contacts:cf_role="gridspec_contacts";
  float xleft(xleft);
    xleft:standard_name="longitude";
    xleft:units="degrees_east";
    xleft:tile="left"; // names the tile this variable belongs to
    xleft:mosaic="gridspec"; // names the mosaic container variable
  float yleft(yleft):
    yleft:standard_name="latitude";
    yleft:units="degrees_north";
    yleft:tile="left";
    yleft:mosaic="gridspec";
  float zwlleft(yleft,xleft);
    zwlleft:standard_name = "sea_surface_height_above_geoid" ;
    zwlleft:units = "m" ;
    zwlleft:tile="left";
    zwlleft:mosaic="gridspec";
  float xright(xright);
    xright:standard_name="longitude";
    xright:units="degrees_east";
    xright:tile="right";
    xright:mosaic="gridspec";
  float yright(yright):
    yright:standard_name="latitude";
    yright:units="degrees_north";
    yright:tile="right";
    yright:mosaic="gridspec";
  float zwlright(yright,xright);
    zwlright:standard_name = "sea_surface_height_above_geoid" ;
    zwlright:units = "m" ;
    zwlright:tile="right";
    zwlright:mosaic="gridspec";
data:
  tile_contacts="left yleft 0:3 xleft 2:2 | right yright 0:4 xright 3:3";
```

Of course, the tiles and mosaic could still be in different files. However, as far as I can see, there would be no need for the

gridspec_tile_name and gridspec_file_type global attributes for M-SPEC. *I am not sure where the gridspec_coordinates_id and gridspec_data_id global attributes belong, because I am not sure exactly what they do.*

In this amended form of gridspec, the data variables which together compose a mosaic of data have different names (`zwlleft` and `zwlright` in the example). They are known to belong together because of their `standard_names` and perhaps other attributes, just as in ugrid.

The amended form of gridspec proposed here could easily be extended to permit any given tile to belong to more than one mosaic, by allowing the `mosaic` attribute to be a blank-separated list of mosaic container variables names, each of which has its own list of constituent tiles in its `tile_names` attribute. Thus we could at the same time describe both the indepedent mosaics belonging to different submodels and the joint mosaic constructed by tile-to-tile contacts between the submodel mosaics, the latter being the arrangement which Balaji talked about. However, we think it is not in the scope of the current version of gridspec to record how the geophysical variables in one submodel are to be computed from those in the other.

22 March 2012, revised 27 March 2012 following webex meeting

Jonathan Gregory