Some Insights into Scrap and Map relationships with Trip Files and Surveys

Adventures with Therion Drawings, Surveys, Files and Namespaces

My need to try to unravel these secrets originates in years of confusion trying to understand the many subtly different ways contributors and helpful Therionisers have put their projects together, particularly with respect to how a survey trip centreline is related to it's associated (scrap) drawing. And thereafter the relationships of descendants of surveys and scraps (super-surveys and maps).

And mainly wondering if there is a 'best' way to do it, or at least what are the consequences of different approaches.

Recently, after another reading of FootLeg's Therion Tutorial, I did some intense pondering and checked out the various datasets I have collected via the Therion Forum over the last 10 years.

I realised that, at least for the entities mentioned in the title above, there seems to be a simple enough pattern that that describes all of the non-trivial examples I have come across.

To set the scene, here is a transcript of the start of FootLeg's (the emphasis is mine, as are the highlighted text)…

Lesson 8: Larger Project Data Structures

Real world surveys are rarely completed in one trip. In this lesson, we are going to reorganise our project so that we can expand it to cover **multiple surveying trips. It is important to have a good data structure for your files in large projects**.

There are many ways you could choose to do this, and different Therion users have their own ways for organising their data. The data structure we are going to use for this lesson is an example of one way to work. **It has the advantages of keeping your files organised by survey trip and allows you to process each trip separately when you are editing just that part of the cave.** Then you can process the whole cave to generate the complete map and models.

We are going to group our data and sketch files for each survey trip together. You can do this simply by naming the files with the same prefix. For larger cave system projects I also group trips together by creating a folder for each area or cave in the system (typically you start with separately named caves that over time get connected into one system, but continue to be referred to by the original name of each major cave entrance). The data organisation is also important in our naming and structure of the survey blocks in the project. The outline of the data structure we are going to be using is as follows:

```
survey EntireCave
  <map definitions for entire cave>
  <scrap joins for entire cave map>
  <equates for entire cave survey>
```

```
    survey Trip1
      <input scraps for trip 1>
      <map definitions for trip 1>
      centreline
            <survey trip 1 data>
      endcentreline
      endsurvey #end of Trip1

      survey Trip2
      <input scraps for trip 2>
      <map definitions for trip 2>
      centreline
            <survey trip 2 data>
      endcentreline
      endsurvey #end of Trip2

      <more trips ....>

endsurvey #end of EntireCave
```

We could enter this data structure into one big .th file, but it is generally **easier to split out each trip into its own file**. So**, each trip has one .th file** containing the centreline (numbers) data, **plus a set of .th2 files (one for each projection)** which go with it containing all the drawings.

Note that the scrap drawings, in .th2 files, are included inside the survey data block for each trip. This is important because it enables Therion to understand which numbered station in the survey centreline data corresponds to which numbered station point in the scraps. …

OK, enough of that (FootLeg's example). When I compared all of the other examples I have come across, I realised that most adhered to some sensible principles, and for those that did, there seemed to be patterns or categories that the datasets fitted into.

Principle One – Maintain granularity, store trip files independently

As FootLeg suggests, each trip has it's own **independent trip files for survey data and for scraps**.

Formatted: Highlight

Principle Two – Store Scrap and Map definitions (SAMD) consistently at all levels

Most examples I found store their scrap and map definitions (SAMD) **consistently at all hierarchical levels**. ie Trip level, Entire cave level, Collections of caves level. I acknowledge that scraps typically only occur at trip level, however maps can be defined at any level.

It seems only sensible that, even though Therion does not enforce such consistency, we should store data the same way at each hierarchical level of the project. I am talking about consistency within the survey project here, not consistency between projects (there is not necessarily much of that)!

Principle Three – Define all inside or all outside surveys

Scrap and map definitions (SAMD) can be located either **inside** the survey they are related to (as in Footleg's example above), or they can be located **outside** of the survey they are related to.
ie IS or OS (for define inside survey or define outside survey)
So we have SAMD-IS, or SAMD-OS

But I found this is not enough to categorise datasets…

Principle Four – Define all inside or all outside of trip files

Scrap and map definitions can be located either inside the Trip File or they can be located outside of the Trip Files,
ie ITF or OTF (for define inside trip file or define outside trip file)

So we have SAMD-ITF, or SAMD-OTF

So now we have two parameters that influence how we can organise our scraps and maps. Two x two gives four combinations of patterns, that (almost?) all of the files I have looked at fit.

Four Categories

SAMD-ITF-IS = Scrap and map definitions – inside trip file – inside survey = FootLeg, Rabbit Cave example Martin Sluka email 13Jul2017 part a

Formatted: Default Paragraph Font, Highlight

Formatted: Highlight

SAMD-ITF-OS = Scrap and map definitions – inside trip file – outside survey = This wiki page was built around what I have ended up doing, but I see I carefully avoided putting actual example files in there! [TO DO – add example]

SAMD-OTF-IS = Scrap and map definitions – outside trip file – inside survey = Padavka Chamber example, Simple Cave example, Martin Sluka email 13Jul2017 part b, wiki faq example description

SAMD-OTF-OS = Scrap and map definitions – outside trip file – outside survey = Danilo Magnani example [TO DO – seek permission to post Danilo example], Will Urbanski example [tidy Will's Marco example]

SAMD-ITF-IS	SAMD-ITF-OS	SAMD-OTF-IS	SAMD-OTF-OS
Scrap and map definitions – inside trip file – inside survey	Scrap and map definitions – inside trip file – outside survey	Scrap and map definitions – outside trip file – inside survey	Scrap and map definitions – outside trip file – outside survey
FootLeg~~FootLeg~~, Rabbit Cave example~~Rabbit example~~ Martin Sluka email 13Jul2017 part a	Bruce wiki description~~Bruce example~~	Padavka Chamber example~~demo padavka~~, Simple Cave example~~demo simple cave~~ Martin Sluka email 13Jul2017 part b wiki faq example description	Danilo Magnani example, Will Urbanski example

So, to clarify, these patterns describe relationships between scrap and map definitions, SAMD, with trip files and survey centreline. This concept should not be confused with the file and folder structure that you use to store the data. A particular SAMD pattern can be manifest in many folder arrangements, but some folder arrangements can only house some SAMD patterns.

To pan out a little to look at the bigger picture, I think a well designed Therion project should have...

1. **Consistent SAMD pattern/relationships** (pick any of the four that you like) on all branches and all levels of the project [the topic of this page/document]
2. A **survey namespace hierarchy** that is simple and not too nested (4 levels at most should accommodate 100's km of passage ie trip, passage series, cave, group of caves. My inclination is to reduce to 3 levels and ditch the passage series)
3. A **folder hierarchy** that matches exactly the survey namespace hierarchy. In addition the folder hierarchy may include other folder branches for say scans of sketches, maps, thconfigs. Whether you choose these extra folders may depend, in part, on the SAMD pattern you have chosen.

Formatted: Border: Box: (Single solid line, Auto, 0.5 pt Line width)

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Font: Bold

Formatted: Font: Bold

Back to SAMD, For the sample project patterns below, I have set the order of references in the files to;

Scrap definitions, map definitions, scrap joins, survey equates, trips

to make them easier to compare, however Therion does not care what order these parameters have. You can put these in any order that makes you happy. [I think] The examples referred to above do not necessarily use this order, but I think each can be reduced to one of these four general patterns.

The following patterns show the Therion project structure, for a hypothetical 'Entire Cave'. The shaded portions refer to data that is stored in another file, and is referenced by a single 'input xxx' line in the EntireCave.th file.

Remember, this is about the relationships between scrap and map definitions with trip files and survey centreline, and not about the survey namespace hierarchy or file and folder structure/hierarchy.

Formatted: Not Highlight

Formatted: Not Highlight

My preference is for smaller EntireCave files, and keeping drawing files close to and named same as Trip files, and so that points to one of the first two patterns following, -ITF. This seems to match many British cavers practice -ITF -IS. Scraps that span more than one survey are the exception, and can be accommodated with a minor variation of the pattern. After a while you learn to survey without causing these situations. I prefer to draw many small scraps, rather than a few large ones, and this helps avoid scraps spanning multiple surveys.

Others prefer to maximise the independence of scraps and maps from surveys, so this points to one of the later patterns, -OTF. This seems to match the practice of those close to the home of Therion, Europe, and possibly American cavers. Drawing files and scraps will tend not to be named after trip surveys, because there is no one to one relationship. Scraps can be larger and do not need to be clipped at survey trip boundaries (although big scraps are a bad idea).

One - SAMD-ITF-IS Scrap and map definitions – inside trip file – inside survey

```
FILE EntireCave.th
        survey EntireCave
        <map definitions for entire cave>
        <scrap joins for entire cave map>
        <equates for entire cave survey>
        input Trip1.th  (contents of FILE Trip1.th enumerated below for the sake of clarity, although
        not actually in the EntireCave file)
                  survey Trip1
                   Input Trip1Plan.th2 <input scraps for trip 1 >
                   Input Trip1Elev.th2 <input scraps for trip 1 >
                   <map definitions for trip 1>
                   centreline
                            <survey trip 1 data>
                   endcentreline
                   endsurvey #end of Trip1

        input Trip2.th  (contents of FILE Trip2.th enumerated below…)
                   survey Trip2
                   Input Trip2Plan.th2 <input scraps for trip 2>
                   Input Trip2Elev.th2 <input scraps for trip 2 >
                   <map definitions for trip 2>
                   centreline
                            <survey trip 2 data>
                   endcentreline
                   endsurvey #end of Trip2

        Input more Trip FILES…
                   <etc ….>

         endsurvey #end of EntireCave
```

Related survey and drawing definitions are defined or referenced in the same file, at trip level.

There tends to be a 1 to 1 relationship between a trip survey and its scraps. Each scrap has stations from only one trip survey.

It is the only ~~category~~ pattern where the scrap ‘point station’s are in the same survey namespace as the trip survey. ie In scraps you can refer to point station -name 23 without a suffix @Trip2

Formatted: Underline

Cannot create maps comprising ‘this centreline’ within the trip file, without violating the data structure

The size of the EntireCave.th file (often referred to as an INDEX file elsewhere) is minimised, and trip files are larger. (The lowest level maps are all defined in the Trip file with -ITF patterns).

~~The size of the EntireCave file is minimised, and trip files are larger.~~

Two - SAMD-ITF-OS Scrap and map definitions – inside trip file – outside survey

```
FILE EntireCave.th
        survey EntireCave
        <map definitions for entire cave>
        <scrap joins for entire cave map>
        <equates for entire cave survey>
        input Trip1.th  (contents of FILE Trip1.th enumerated below for the sake of clarity, although
        not actually in the EntireCave file)
                Input Trip1Plan.th2 <input scraps for trip 1 >
                Input Trip1Elev.th2 <input scraps for trip 1 >
                <map definitions for trip 1>
               survey Trip1
                centreline
                        <survey trip 1 data>
                endcentreline
                endsurvey #end of Trip1

       input Trip2.th  (contents of FILE Trip2.th enumerated below…)
                Input Trip2Plan.th2 <input scraps for trip 2 >
                Input Trip2Elev.th2 <input scraps for trip 2 >
                <map definitions for trip 2>
                survey Trip2
                centreline
                        <survey trip 2 data>
                endcentreline
                endsurvey #end of Trip2

        input more Trip FILES…
                <etc ….>

         endsurvey #end of EntireCave
```

Related survey and drawing definitions are defined or referenced in the same file, at trip level.

There tends to be a 1 to 1 relationship between a trip survey and its scraps. Each scrap has stations from only one trip survey.

Scrap ‘point station’s are not in the same survey namespace as the trip survey. In scraps you have to refer to the namespace for each point. ie point station -name 23@Trip2. A good work around where there is only trip per scrap is to set the parameter -station-names [] @Trip2 in each scrap definition ~~A good work around is to set the parameter -station-names [] @Trip2 in each scrap definition~~.

Can create maps comprising ‘this centreline’ within the trip file, without violating the data structure

The size of the EntireCave.th file (often referred to as an INDEX file elsewhere) is minimised, and trip files are larger. (The lowest level maps are all defined in the Trip file with -ITF patterns).

Three - SAMD-OTF-IS Scrap and map definitions – outside trip file – inside survey

```
FILE EntireCave.th
        survey EntireCave
        input Trip1Plan.th2 <Reference scraps for trip 1 >
        input Trip2Plan.th2 < Reference scraps for trip 2>

        input Trip1Elev.th2 <Reference scraps for trip 1 >
        input Trip2Elev.th2 < Reference scraps for trip 2>

        input more Trip drawing FILES…
         <or you put all drawings and joins in an EntireCaveMap.th(2) files and just input them here>
                   <etc ….>

        <map definitions for entire cave, all in one place, none in trip files>
        <scrap joins for entire cave map>
        <equates for entire cave survey>

        input Trip1.th  (contents of FILE Trip1.th enumerated below for the sake of clarity, although
        not actually in the EntireCave file)
                  survey Trip1
                   centreline
                            <survey trip 1 data>
                   endcentreline
                   endsurvey #end of Trip1

        input Trip2.th  (contents of FILE Trip2.th enumerated below…)
                   survey Trip2
                   centreline
                            <survey trip 2 data>
                   endcentreline
                   endsurvey #end of Trip2
        input more Trip FILES…
                   <etc ….>


        endsurvey #end of EntireCave
```

Formatted: Centered

Related survey and drawing definitions are defined in separate files and are tied together at the next hierarchical level up, ie EntireCave

There is no need to maintain a 1 to 1 relationship between a trip’s survey and its scraps. Each scrap may have stations from a number of trip surveys.

Scrap ‘point station’s are not in the same survey namespace as the trip survey. In scraps you have refer to the namespace for each point. ie point station -name 23@Trip2. A good work around where there is only trip per scrap is to set the parameter -station-names [] @Trip2 in each scrap definition ~~A good work around is to set the parameter -station-names [] @Trip2 in each scrap definition~~.

Can create maps comprising individual trip centrelines within the EntireCave file, without violating the data structure.

~~The~~ To simplify and minimise the size of the EntireCave.th file (often referred to as an INDEX file elsewhere) an EntireCaveMap.th(2) is usually created to contain maps of each projection ~~larger~~ (There are at least three times as many files to input to the EntireCave survey (assuming two projections drawn), more map definitions when compared to -ITF patterns).

Four - SAMD-OTF-OS Scrap and map definitions – outside trip file – outside survey

```
FILE EntireCave.th
        input Trip1Plan.th2 <Reference scraps for trip 1 >
        input Trip2Plan.th2 < Reference scraps for trip 2>

        input Trip1Elev.th2 <Reference scraps for trip 1 >
        input Trip2Elev.th2 < Reference scraps for trip 2>

        input more Trip drawing FILES…
        <or you put all drawings and joins in an EntireCaveMap.th(2) files and just input them here>
                <etc ....>

        <map definitions for entire cave, all in one place, none in trip files>
        <scrap joins for entire cave map>

        survey EntireCave
        <equates for entire cave survey>

        input Trip1.th  (contents of FILE Trip1.th enumerated below for the sake of clarity, although
        not actually in the EntireCave file)
               survey Trip1
                centreline
                        <survey trip 1 data>
                endcentreline
                endsurvey #end of Trip1

        input Trip2.th  (contents of FILE Trip2.th enumerated below…)
                survey Trip2
                centreline
                        <survey trip 2 data>
                endcentreline
                endsurvey #end of Trip2
        input more Trip FILES…
                <etc ....>


        endsurvey #end of EntireCave
```

Related survey and drawing definitions are defined in separate files and are tied together at the next hierarchical level up, ie EntireCave

There is no need to maintain a 1 to 1 relationship between a trip's survey and its scraps. Each scrap may have stations from a number of trip surveys.

Scrap 'point station's are not in the same namespace as the survey. In scraps you have refer to the namespace for each point. ie point station -name 23@Trip2. A good work around where there is only trip per scrap is to set the parameter -station-names [] @Trip2 in each scrap definition.

Can create maps comprising individual trip centrelines within the EntireCave file, without violating the data structure.

To simplify and minimise the size of the EntireCave.th file (often referred to as an INDEX file elsewhere) an EntireCaveMap.th(2) is usually created to contain maps of each projection (There are at least three times as many files to input to the EntireCave survey (assuming two projections drawn), more map definitions when compared to -ITF patterns).

~~The size of the EntireCave file is larger (at least three times as many files to input (assuming two projections drawn), more map definitions).~~

Martin Sluka folder structure example:

[This folder structure says nothing about SAMD pattern that Martin uses]

```
my_cave                                          – folder
      my_cave.thconfig
      my_cave.th
      data                                       – folder
         north_branch                            – folder
              north_branch_170321                – folder
                    north_branch.thconfig
                    north_branch_170321.th
                    north_branch_170321_scrap1.th2
                    north_branch_170321_scrap2.th2
                    north_branch_170321_scrap3.th2
                    ...
                    scans                         – folder
                    output                        – folder
              north_branch_170514                 – folder
                    ...
         south_branch                             – folder
              ...
      index                                       – folder
      output                                      – folder
```

Vasily Suhachev folder structure example:

[This folder structure must be SAMD-OTF-OS, I think]

Collections of caves must end up with a very nested and complex folder structure

```
Cave
    +-- SurveyNotes                # scanned notebooks
        +-- trip1
            +-- 1.jpg
            +-- 2.jpg
            +-- 3.jpg
        +-- trip2
        +-- trip3
        +-- trip4
    +-- Centerline                  # centerline
        +-- caveindex.th            # index file with equates
        +-- trip1.th
        +-- trip2.th
        +-- trip3.th
        +-- trip4.th
    +-- Model
        +-- thconfig
    +-- Map
        +-- Plan                     # plan
            +-- subsystem1
                +-- 1.th2
                +-- 2.th2
                +-- maps.th
                +-- thconfig
            +-- subsystem2
                +-- 3.th2
                +-- maps.th
                +-- thconfig
            +-- maps.th
            +-- thconfig
        +-- ElevEXT                   # extended elevation
        +-- Elev240                   # elevation at 240
```