# Tree

*How to set up a good project tree and why we need it*



---

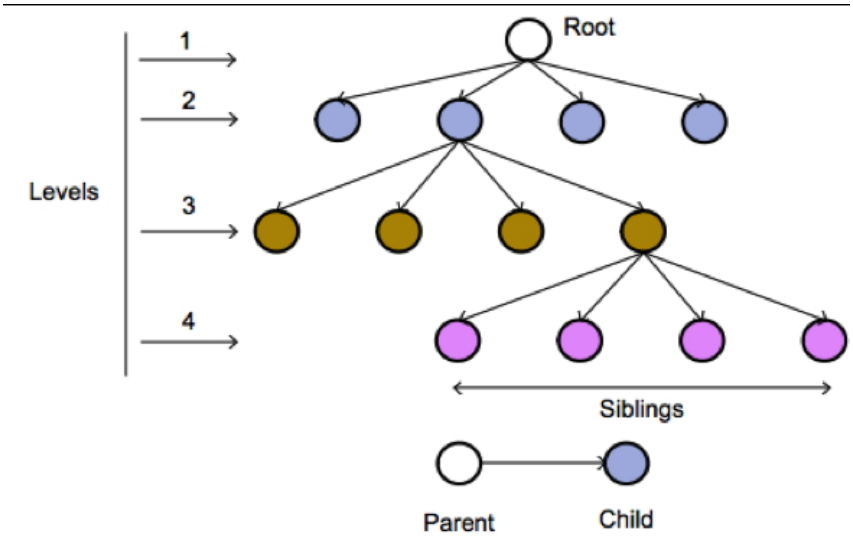
## Tree structure

The concept of tree structure is generally well know to planners. Unfortunately, often monster trees are constructed.
The good health of the project tree is important, because a healthy tree will help us understand the project process and sub-processes.
A monster tree will cause confusion.

The project tree is set up when we construct a planning with our favorite scheduler software.
Constructing the project tree consists in

- grouping atomic task into summaries,
- grouping summaries into higher level summaries,
- and so on, until the root of the project is reached.

This is the graph representation of a tree:




The mathematical definition of a tree - a special type of graph - is :
A tree is a directed graph whereby these rules apply:

- There is only one root item
- The relationship between items is "parent to child" or "sibling to sibling".
- Any given parent can have many children
- Any given child has only one parent.

# Why tree ?

Why should we care?
Setting up a healthy project tree will help us put order and structure into what would else be a chaos of tasks.
It will help us read the schedule and it will help us explain and communicate the schedule.

But there is more to it.
Often will the data of the schedule be exported and used by some third party analytical tool with the aim of producing reports:

- risk analysis
- progress reports
- resources usage
- cost analysis

The quality of the reports depends on the quality of the tree structure of the original schedule. The readability and the significance of the reported items will be degraded if the tree is not healthy.

# Good tree - Bad tree

The health of a project tree is defined by two aspects:

- coherence
- being free of hybrids

## Coherence

When grouping tasks into summaries we will try to setup coherent sets. A coherent set with contain task of the same nature.
Coherence can be defined as

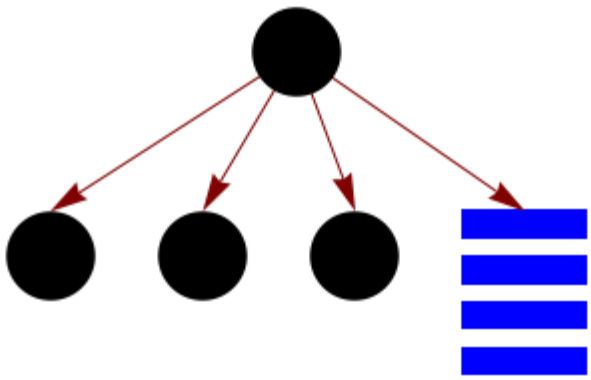
- belonging to the same project phase
    - engineering
    - procurement
    - construction
    - commissioning
- belonging to the same discipline
    - civil
    - steel construction
    - mechanical
    - piping
    - electrical
    - software
- belonging to the same area
    - unit 1
    - unit 2
    - …

## Hybrid structures

*A non-hybrid is a summary that contains only sub-summaries **or** only tasks.*

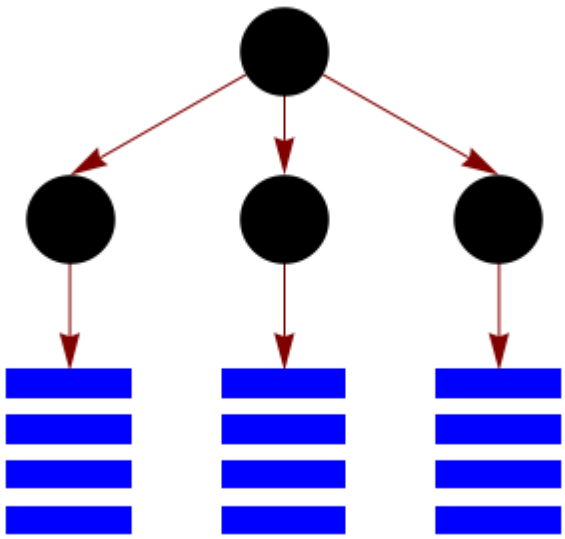
A hybrid is a summary that contains a mixture or tasks and sub-summaries.
Voila!

This is a hybrid:



Black disk: summary, or sub-summary, or sub-sub-summary, or …
Blue bar: atomic task

This tree has no hybrids:



The tasks in a hybrid do not have a parent at the same level as their siblings. The analytical tools that use the tree structure as a framework to structure their reports, will then produce “blurred” data.
The behaviour of the top node (black disk) can be understood from the behaviour of it’s child nodes when there is no hybrid.
This is not the case when there is a hybrid: only part of the behaviour of the parent node can be understood from the behaviour of it’s child nodes.The “hybrid” tasks are invisible and can overshadow the behaviour of the sibling nodes. The leads to confusing reports.

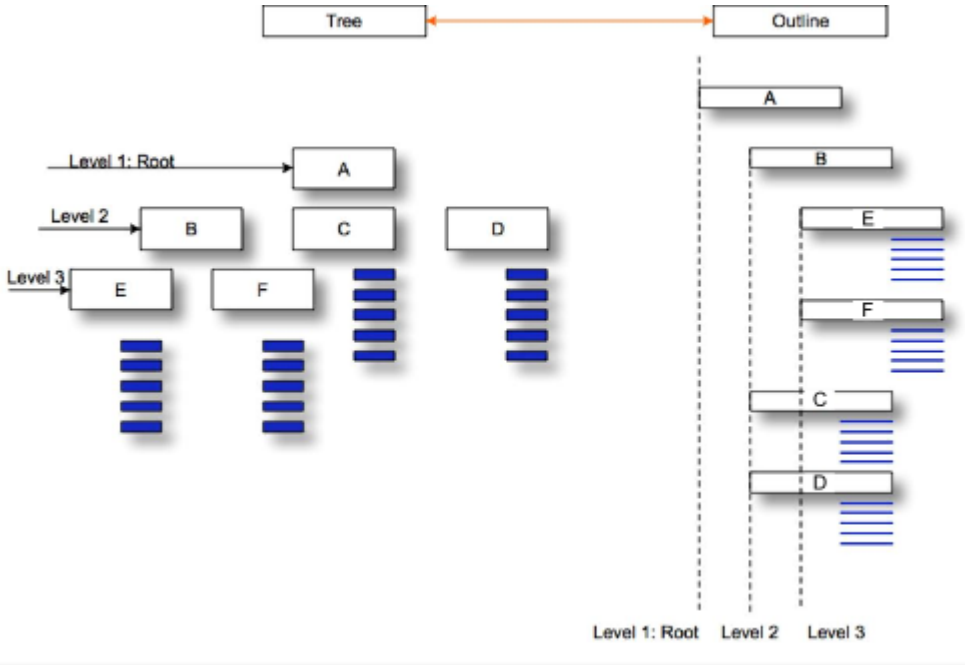
*Hybrids degrade the readability of structured reports.*

## How ?

There are two mechanisms to construct the project tree:

- the outlining
- the WBS code

The outlining is the most frequent system: outlining the tasks effectively defines the tree:




The use of directly defined WBS code (work breakdown code) is less frequent and often prone to create monster trees.
This is the WBS coding of the above example:

A → 1
B → 1.1
C → 1.2
D → 1.3
E → 1.1.1
F → 1.1.2
The tasks under E would become resp. 1.1.1.1, 1.1.1.2, 1.1.1.3, etc.

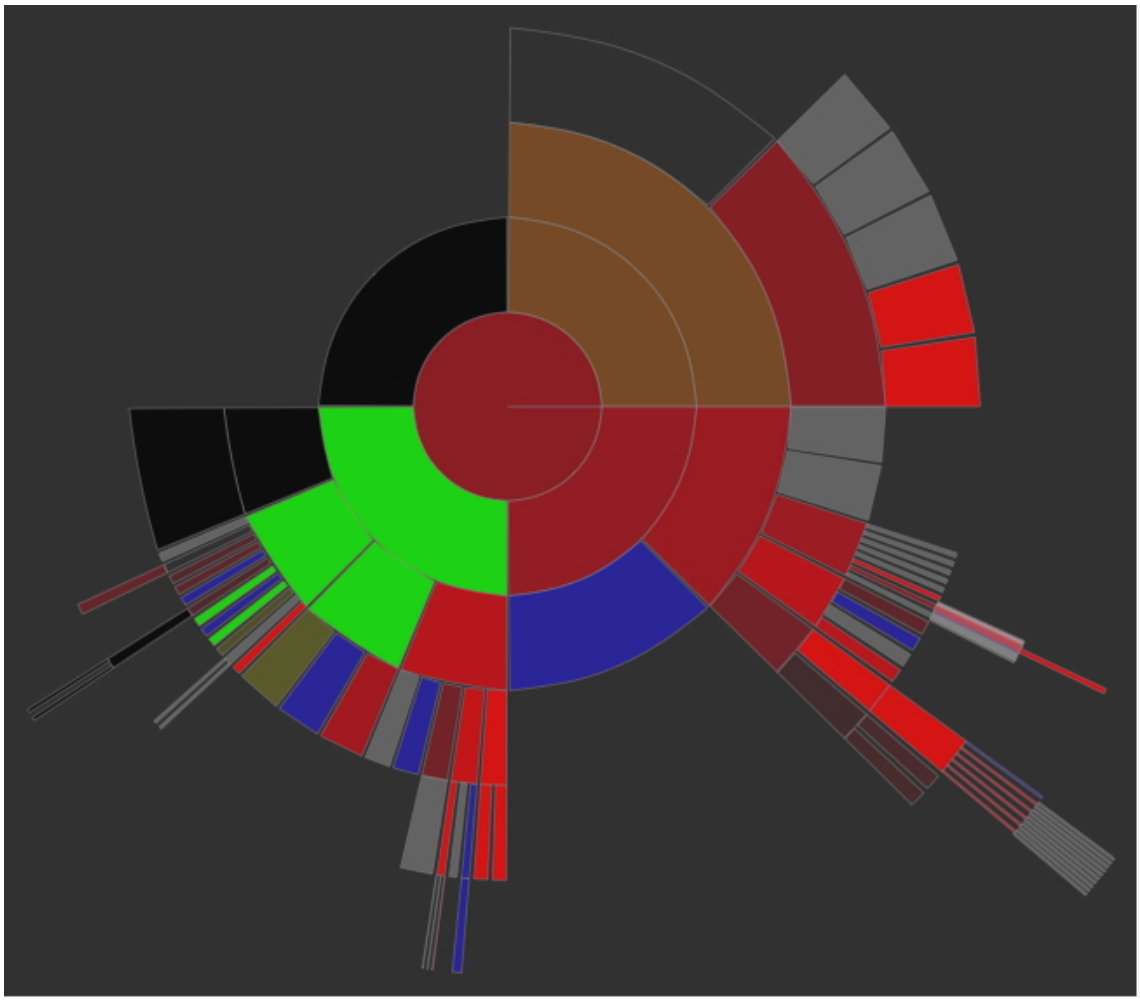
**Warning**
Some schedulers generate inconsistent wbs codes, eg. when lots of changes have been applied to the schedule. In such case, a renumbering of the wbs codes in necessary. The renumbering facility is generally available as a recovery procedure.

## How many levels?

As in all things: not too few, not too many.

### Best practice

According to our experience it is a good idea to stay in the range 4 to 6 for the nodes.
This means that the deepest tasks are then on level 5 to 7.
Here is an example of a very deep - 8 deep - and very asymmetric tree shown as a daisy chart, aka Daisy Tree. The asymmetry results in the whiskers. Such structure is difficult to handle and should be avoided.



---

For further information you can contact me at
jp@tollenboom.be and @JPToll,

or simply post a comment on my blog www.jptollenboom.org