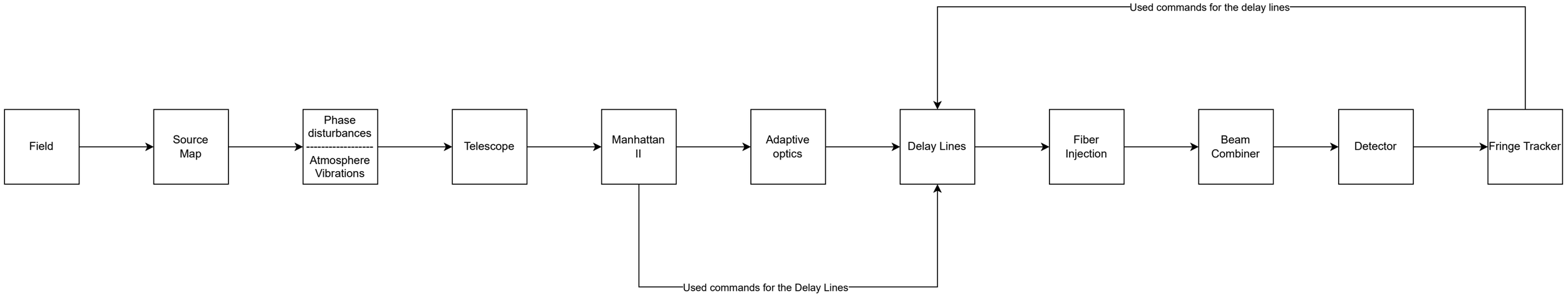

Motivation of the structure – top-down requirements:

- The system is built from class objects that modify a major class; the Field class – class contains the EMF of light as it propagates from system to system.
- The field is modified by the Objects to reflect the underlying physical process – e.g., the atmosphere object receives a Field class and adds the turbulent phase to the EMF.
- To simplify the code structure, the object needs to be de-coupled. The input of a code must be only the Field object, and the modification should operate only on the Field object.
- Each of the object classes is built from an abstract class that must include the following:
    - **Initialize function** – initializes the object and loads the parameters from the preamble section.
    - **Propagation function** – the function that picks up the field from the previous object and operates the field to reflect the physical process of the current object – e.g., a beam combiner class splits the introduced four telescopes into 24 outputs in an ABCD configuration (6 baselines x 4 ABCD inputs).
    - **The state function** yields the current state of the object – e.g., this will include the mirror commands in an AO class; the delay line positions in the Delay lines class.

Structure – basic code lay down:

Preamble – parameter loading. Reads parameter files using a reader function

1. Telescope/atmospheric/source parameters; AO/FT/MAH2 configuration; Beam combiner matrices; source parameters; DL/Detector/Spectrometer static parameters; …

Initialization of the field - The initialized field in a first call is empty and need information on the source to populate the class. Every object propagates the field and modifies it to reflect the effect of the object (e.g., a deformable mirror modifies the wavefront phase with the deformable mirror).

1. field = field.initialize();

Propagation of the field across the other Digital Twin components:

1. source = source.initialize(source parameters);
2. source.propagate(field);
3. atmosphere = atmosphere.initialize(atmospheric parameters);
4. atmosphere.propagate(field);

Note on control strategies: The control strategies applied to each of the control elements is fixed for a loop – this implies that the current strategy (e.g., FT using an integrator) needs to be included in the initialize function and loaded from the parameters.

Structure – basic pseudo-code loop:

```
## Preamble

objects = [Source, Atmosphere, Phase_Disturbances, Telescope, Manhattan, Adaptive_Optics, Delay_Lines, Fiber_injection, Beam_combiner, Detector, Fringe_tracker]
param_objects = load_parameters( parameter_file.txt )

# Load parameters into objects
[current_object.initialize(current_parameters) for current_object, current_parameters in zip(objects, param_objects) ]

# Initialize field

emf = Field.initialize()

# Loop

for frame in range(number_of_iterations ):

    [current_object.propagate(emf) for current_object in objects]

# Analysis Loop

(...)
```