Digital Twin Identifiers

Wednesday May 4th, 2022

Martin Hardwick

Convenor WG15

Universally Unique Identifiers (UUID)

- UUID is a large number that is unlikely to ever be duplicated
- There are four types of UUID
  - UUID.4 is generally preferred
  - UUID.5 retains some history
- Different opinions on how to use the UUID
  - UUID used for one value/entity
  - UUID links several values/entities
- This project is about using UUID’s to identify digital twins
  - The digital twin is described in many places / data formats
  - The UUID identifies the twin in all the places.

Use cases for manufacturing

1. The digital twin is explicitly modeled
   - The design of a bolt
2. The digital twin is implicitly modeled
   - The six bolts in the AS1 assembly
3. The digital twin will exist at a future time
   - The holes that are going to be drilled and filled on a wing

Explicitly modeled digital twins

- Identify using a header entity
  - Without categorization

    ```
    <f1f98802-2f74-4100-9048-2909bf732679>=#765
    ```

  - With categorization / classification / description

    ```
    <REQUIREMENT::e00a534e-4a44-4fd1-a530-e4f0abc34037>=#2693;
    ```

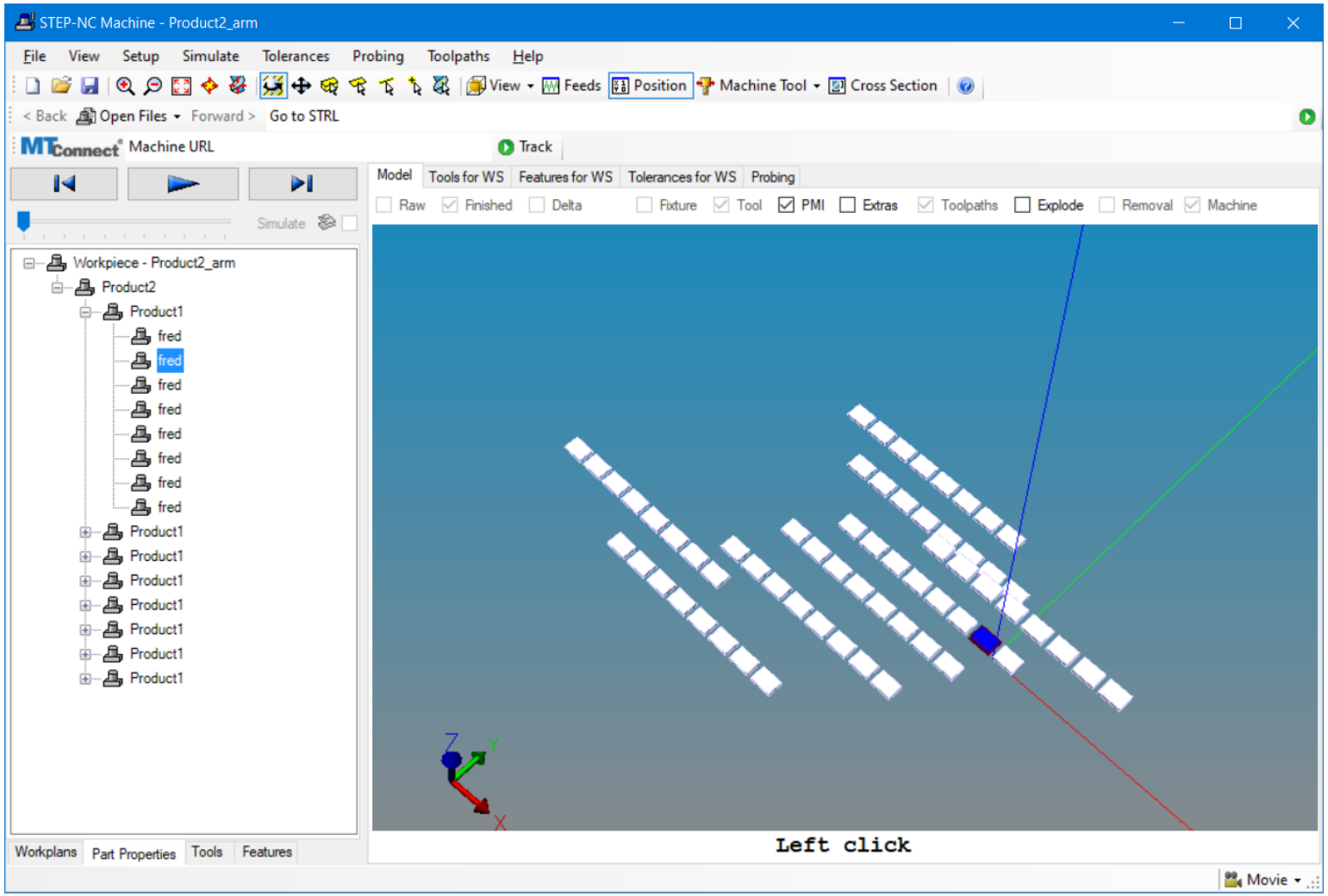
- Identify using a data entity
  - Entity 30246 assigns f1f988... to entity #765

    ```
    #30246=GUID('f1f98802-2f74-4100-9048-2909bf732679', #765);
    ```

Implicitly modeled digital twins

- One design and one assembly creates an array of 64 pins
- Three solutions

1. We can invent a notation to identify the implicit twin
2. We can add data to identify the implicit twin
3. We can copy data to make the twin explicit



Invent a notation to find the twin (case 1)

- CAx systems can read the data and find implicit digital twins.
- We can define a notation to identify these twins and give them UUIDs
  - For example, we can number the assemblies using a defined sequence
    - A1. A2, A3 etc
    - `<f1f98802-2f74-4100-9048-2909bf732679>=A1`
  - Or, we can define an implicit path in the data
    - STEP assemblies are required to have unique product id's and NAUO id's
    - `<f1f98802-2f74-4100-9048-2909bf732679> = Product.id = "Product.1" => NAUO.id="fred.2"`

Add data to identify the twin (case 2)

- We can add entities to designate the twin
  - For example, the STEP multi_level_reference_designator entity
  - First define the MLRD
  - Then define a GUID to reference the MLRD
  - Then add properties to the MLRD
- However, our “tangled spaghetti” becomes even more tangled
  - Harder for low level CAD systems to read and write the data
  - Maintenance issues when GUID data contradicts traditional data

Copy data to make the twin (case 3)

- The holes and fasteners that will be drilled onto the wing assembly
  - An AP238 program defines how to drill the holes
  - An AP238 program defines how to fill the holes with fasteners
  - Result is an AP242 wing assembly containing holes and fasteners

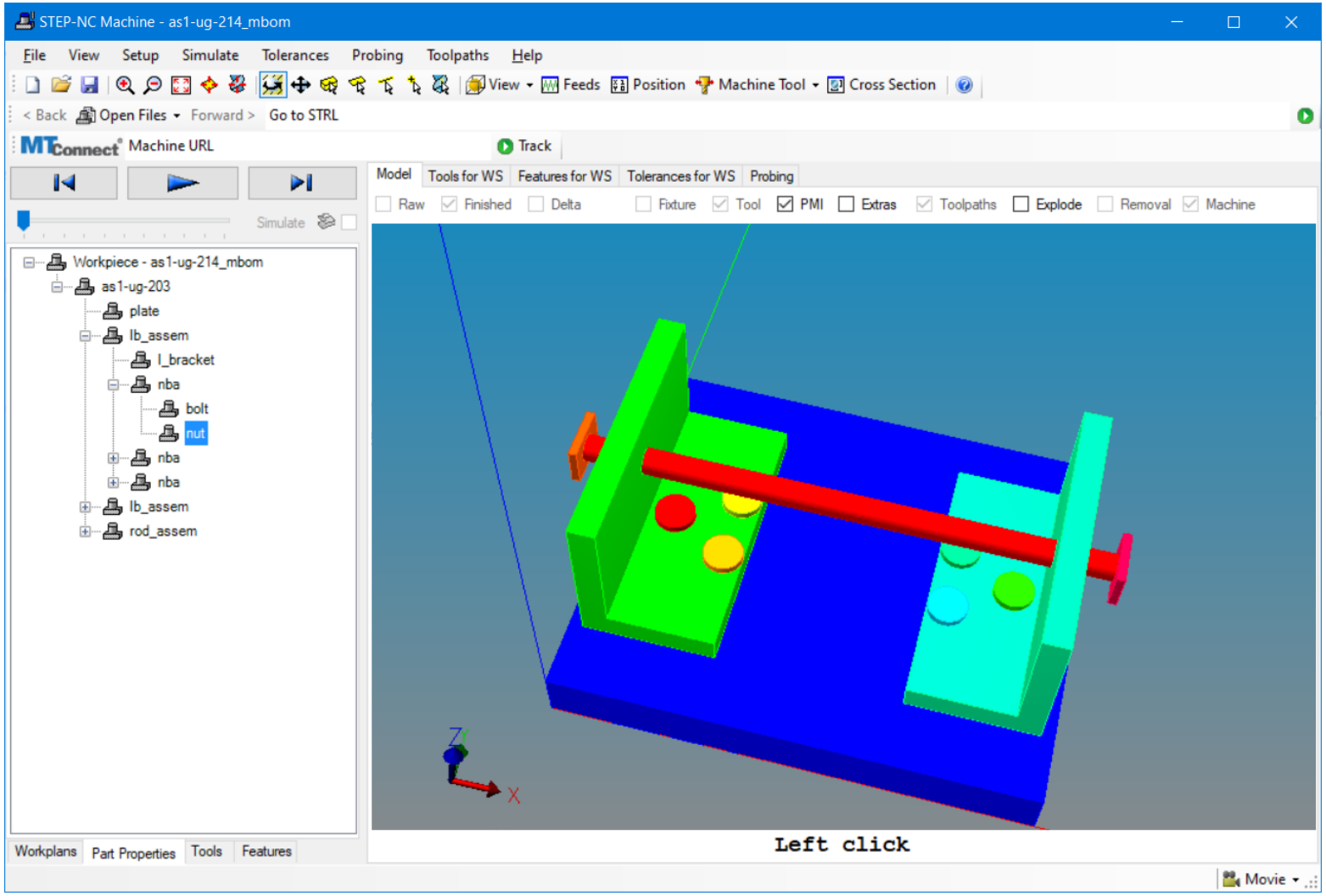
- AP238 Solution
  1. Store UUID in the workingstep (WS) that does the drill and fill
  2. Transfer UUID from the WS to the hole when it is drilled
  3. Transfer UUID from the WS to the fastener when it is filled

Need new AP238 operations to give a new product / feature a new UUID (and name and serial#)

Summary

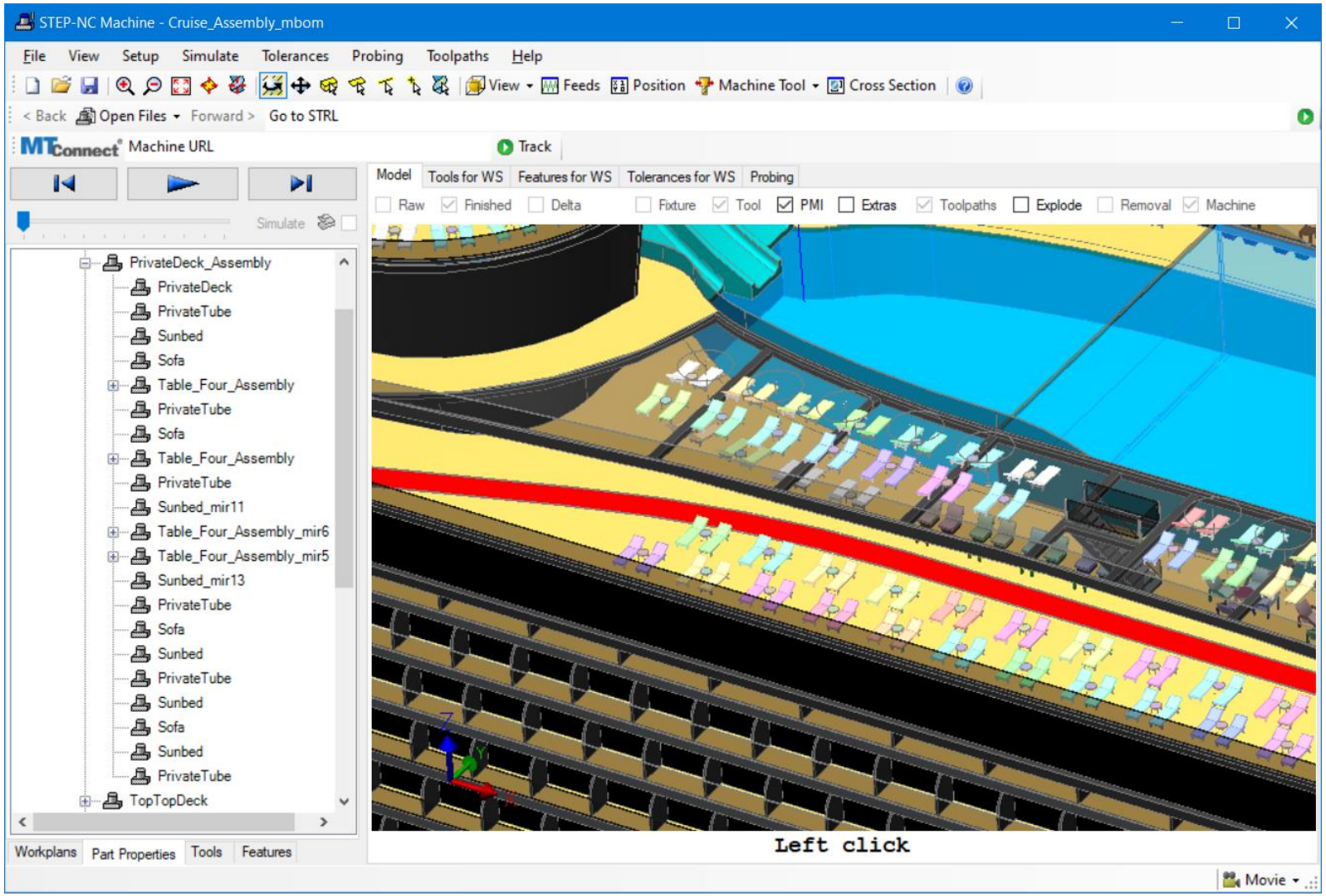
- Solution 1 – create new notation to identify implicit digital twins
  - WG 11 effort
  - Considerable effort to define the notation
- Solution 2 – add data to identify implicit twins in AP242
  - WG 12 effort
  - Definitions already made by CAX-IF and PLCS in 2014
- Solution 3 – copy data to make the digital twins using AP238
  - WG 15 effort
  - Operations to assign new identity to new digital twins as they are created

Digital Twin example – AS1



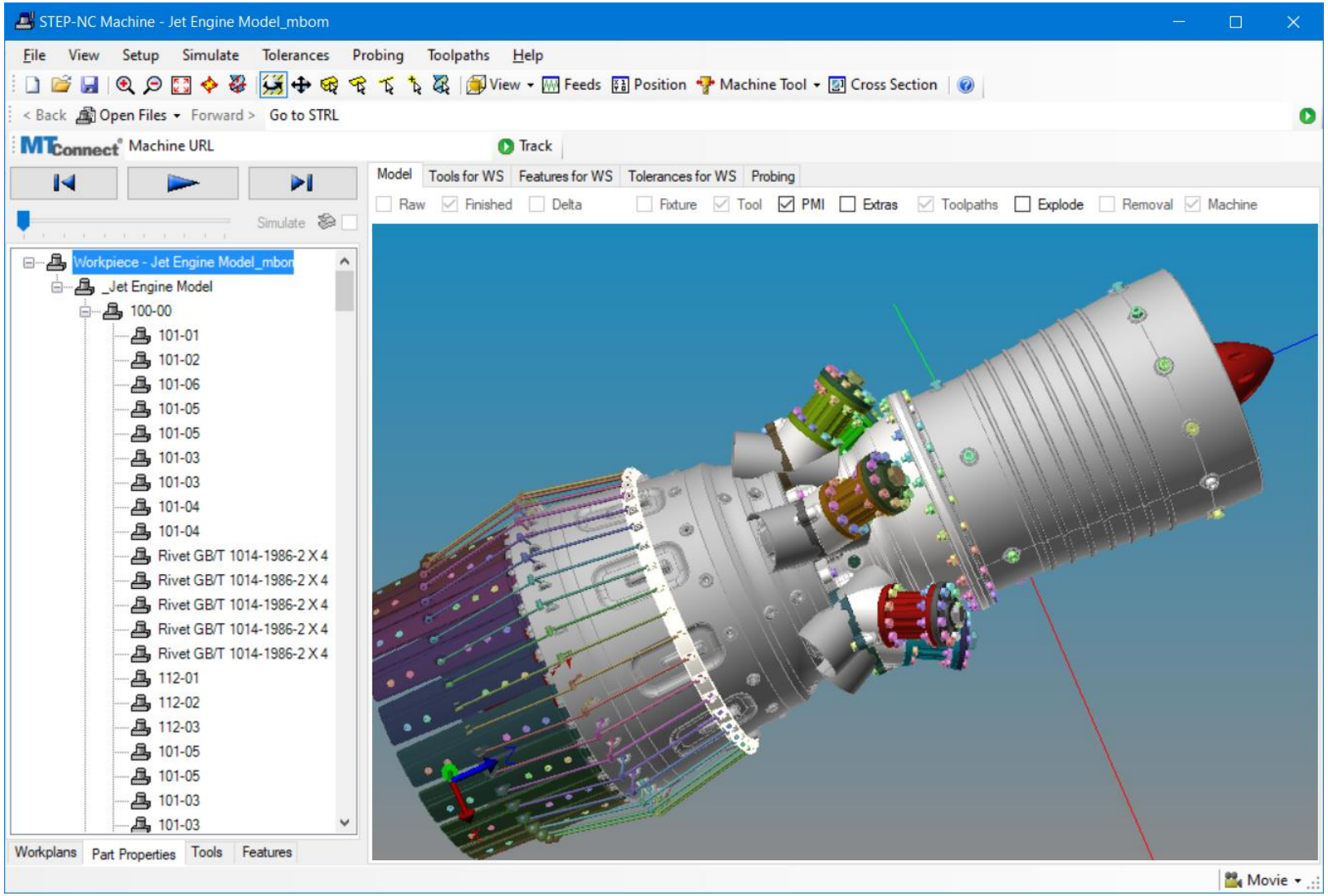
Twins are colored

Digital Twin example – Cruise Ship



Files sizes are doubled

Digital Twin example – Jet Engine



Examples are solution 3