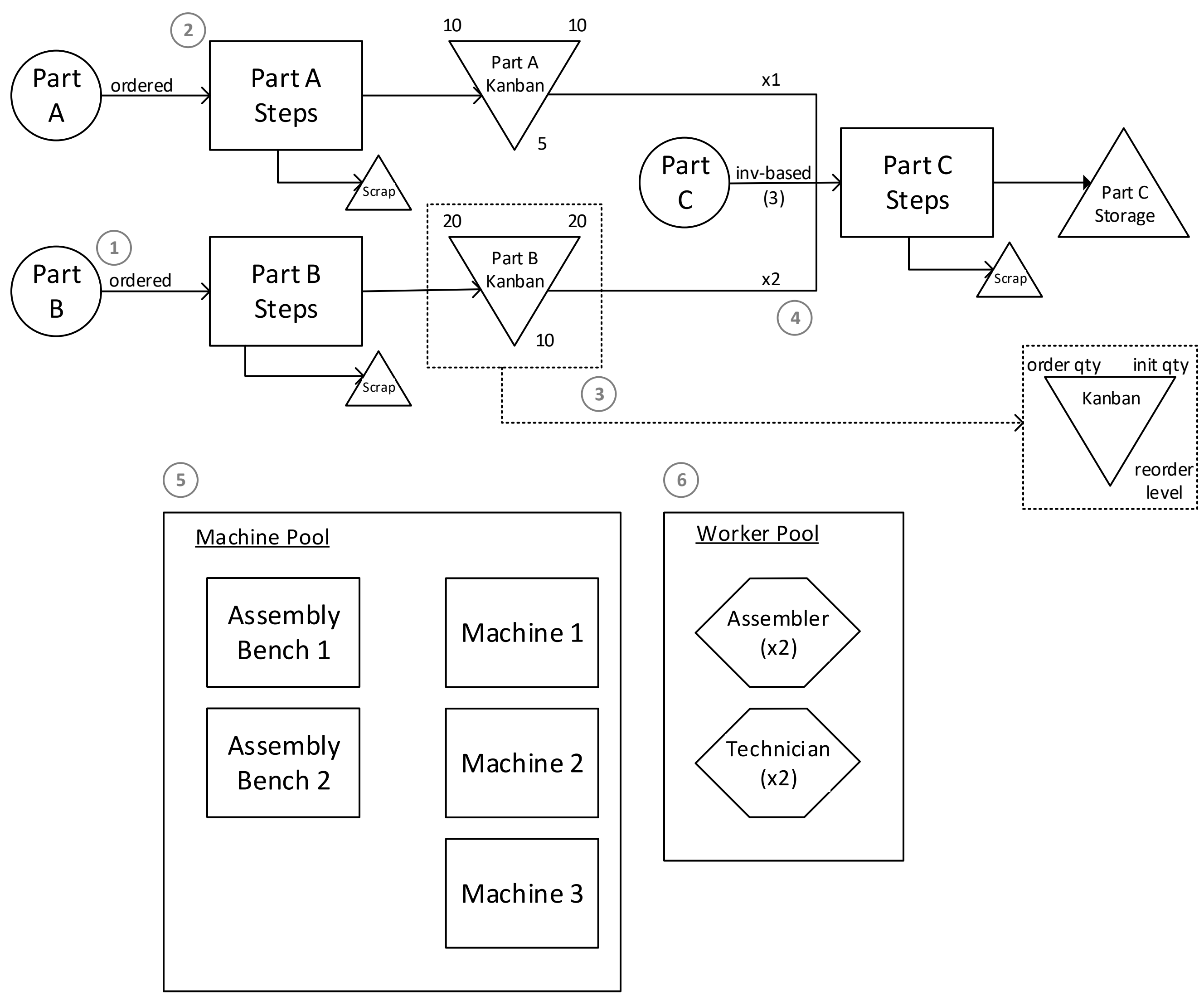
Overall Process Flow: Simple Example



1 Entities ("Parts") can be generated through a variety of methods. They are as follows…
Continuous: Parts are made continuously every 1 time epoch
Periodic: Parts are made every x time epochs, defined by the user
Ordered: A Kanban orders more parts to be made at a certain inventory threshold
Inv-based: Parts are made to maintain x number of parts being processed at all times, defined by the user

2 Entity steps are the detailed set of operations an entity takes before completion or exiting the system. Steps are predefined before entering the system by using a function which determines the steps taken for that particular entity instance. (More detail provided below)

3 The Kanban acts as an ordering system for downstream processes. They can be initialized with a given quantity, "init qty". And then they will order entities at quantity "order qty" when the Kanban contents falls below its "reorder level". Entities must arrived based on an order to use.

4 Upstream processes can have build of materials in which downstream entities can be taken from their storage location and used for the upstream entity. The "bom" variable indicates these dependencies. Here's how Part C would be formatted for the given process:

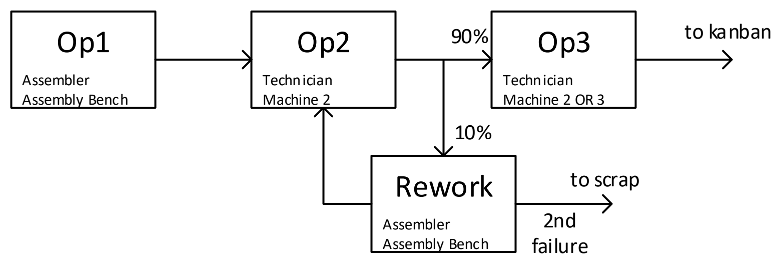
```
{
    "Part A": {
        "location": Part A Kanban,
        "qty": 1
    },
    "Part B": {
        "location": Part B Kanban,
        "qty": 2
    }
}
```

5 Machines should be thought of as physical locations an entity has to pass through in order to move on to the next step in its process.

6 Workers should be thought of as floating requirements that must be present at the physical Machine location in order for an entity to be processed and move to its next step. Workers can have designated shift schedules where they will be unable to fulfill requests at certain times.

NOTE:
Outside of the dotted legend box for Kanban explanation, this would be the expected graphic to be generated if animation were to be added to the simulation model.

7 Part A Steps Flow

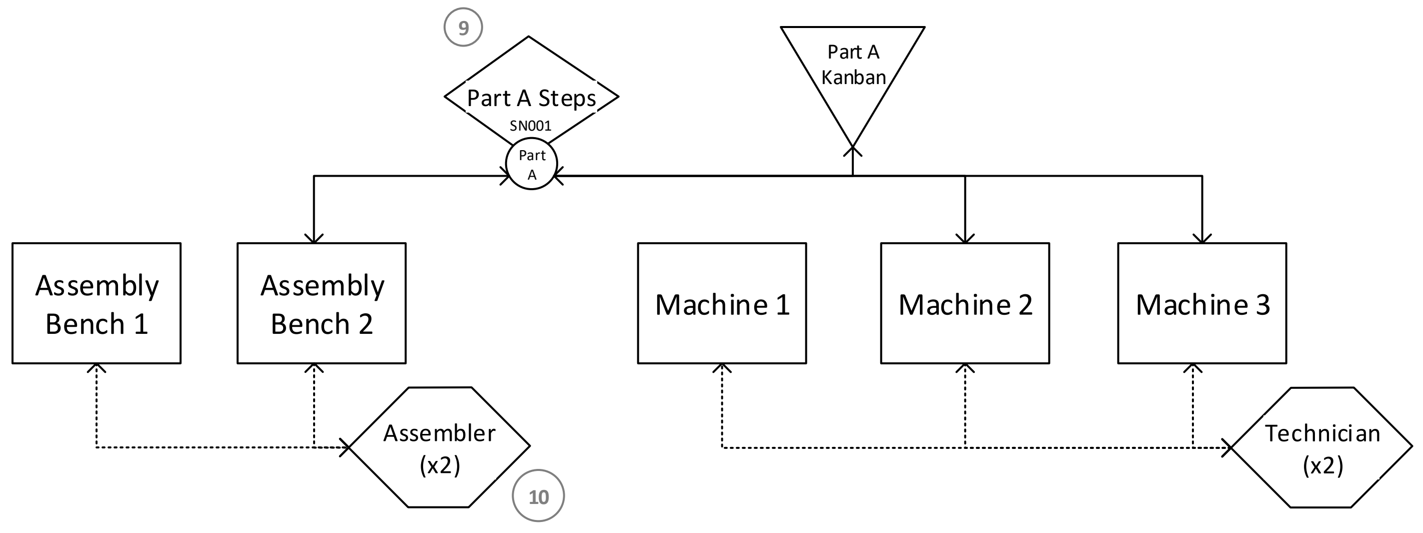


7 Part A Steps Flow depicts the decision tree used in determining the steps that are taken for an entity. A function executes the decision tree upon an entity being created where it determines all the steps to be taken for that entity before entering the system. Each "operation" is a formatted JSON object which indicates what is required for the operation as well as the properties that determine what happens next. JSON objects for Op1 and Op2 are shown on the right.

8 The JSON objects are overwritten in some cases during function execution (i.e. making decisions based on percentages, sampling values from distributions, etc.) to comprise a list on explicit steps the entity will take when it enters the system

```
{
    "Op1": {
        "location": Assembly Bench,
        "worker": Assembler,
        "setup_time": 0,
        "run_time": 15,
        "teardown_time": 0,
        "transit_time": 1,
        "route_to": Op2
    },
    "Op2": {
        "location": Machine 2,
        "worker": Technician,
        "manned": False,
        "setup_time": 5,
        "run_time": 10,
        "teardown_time": 2,
        "transit_time": 1,
        "yield": 0.96,
    8   "route_to_pass": Op3,
        "route_to_fail": rework
    },
    ...
}
```

Part A Entity Process within salabim_plus



9 Part A steps can be thought of as a list of instructions telling that entity what Machine to go to, which Worker to request once it gets there, and how long it will take to process. The figure shows a more literal sense of what is happening when salabim_plus executes its simulation code.

10 Worker availability can also be controlled by the ShiftController which indicates when a Worker is actually working. During unavailable times, any machine using a worker will be interrupted and any requests for that worker will wait in a hold pattern until a worker returns to work and everything resumes. Shifts are formatted in a variety of methods. They are as follows…
Continuous: A defined shift for one day repeating everyday.
Pattern: A defined list of shifts repeating at the end of the list.
Custom: A defined list of shifts that occur and a worker no longer works after the last shift.