# Alloyed – Technical Challenge 1

The Additive Manufacturing Machine requires the end user to be able to build a machine that is one of three different base types: High Power, Medium Power or Lower Power. This machine may then optionally be wrapped in specific features/add Ons by the customer and hence the decorator design pattern seems most viable. The decorator pattern allows features to be added or removed at runtime by wrapping the objects. This is critical for the system as the machine can dynamically gain or lose features such as adding a QuadLaser or removing a RecucedBuildVolume. Also, the decorator supports composition over inheritance, allowing features to be layered in any order without creating a combinatorial explosion of subclasses. The builder pattern was also an option that I considered however it does not allow the same runtime flexibility as the decorator pattern as it typically operates at object creation.

**2 Main principles applied**

1. Open/Closed Principle:
   1. Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification.
   2. The implementation of a MachineDecorator allows for the extension of the behaviour of the machines i.e. adding new features without having to modify the base classes such as IMachine, MediumPowerMachine etc.
2. Single Responsibility Principle:
   1. A class should handle one specific part of the systems behaviour or logic. If it tries to handle multiple responsibilities, it becomes harder to manage.
   2. The implementation of a MachineDecorator ensures each decorator class i.e. QuadLaser, Photodiodes is responsible for adding one specific feature to a machine. The base machine classes i.e. LowPowerMachine, MediumPowerMachine remain focused solely on providing the core functionality. This separation of concern means it’s very simple to add more decorators or more base classes without affecting other parts of the system.