Brewery Problem Metrics and Peer Review Analysis

1. Metrics Analysis

This report analyzes the *Brewery Problem* project based on five structural metrics: **Weighted Methods per Class (WMC)**, **Depth of Inheritance Tree (DIT)**, **Number of Children (NOC)**, **Coupling Between Objects (CBO)**, and **Response for a Class (RFC)**. These values reflect the system's overall complexity, cohesion, and design balance.

Package	WMC	DIT	NOC	CBO	RFC
brewery.app	21	2	0	15	20
brewery.inventory	31	2	0	9	18
brewery.plant	63	5	5	16	34
brewery.production	47	3	1	13	29
brewery.recipes	28	2	0	7	16
brewery.services	25	2	0	11	21
Total	**215**	**16**	**6**	**71**	**138**

Complexity (WMC): Overall complexity is moderate (215). Nearly half originates from `plant` and `production`, confirming that manufacturing and orchestration dominate system logic.

Inheritance (DIT/NOC): The hierarchy remains shallow (DIT ≤ 5, NOC = 6), mainly from `Vat` and `Sensor` abstractions. This is structurally acceptable but could be simplified by merging minor subclasses.

Coupling (CBO): Coupling peaks in `app` (15) and `plant` (16). These results reflect expected dependencies from orchestration but suggest opportunities to reduce cross-module reliance.

Responsiveness (RFC): RFC values are well-distributed. Keeping most classes below 12 ensures maintainable interfaces and testability.

2. Summary of Group Discussion and Surprises

During group review, they presented simpler architectures with fewer subpackages. Their total complexity values were slightly lower, but they achieved this through **cleaner grouping and fewer class boundaries**.
In contrast, my design favored strict modular separation, resulting in more detailed—but heavier—package interactions. Despite higher complexity, it provided flexibility for future scaling and clearer domain ownership.

The most surprising finding was that **my architecture was more granular than others**. While I aimed for modular precision, excessive subdivision inflated WMC and CBO values.
They adopted broader modules achieved cleaner dependency graphs and easier-to-read structures, even with similar functionality. This revealed that **clarity sometimes outweighs granularity** in maintainable design.

3. Planned Changes

1. **Merge small modules** in `plant` and `production` to reduce WMC and RFC totals.
2. **Delegate orchestration** from `BrewerySystem` to separate service layers.
3. **Introduce domain ports** (e.g., `MonitoringPort`, `SchedulerPort`) to lower coupling between layers.
4. **Simplify inheritance**—replace trivial subclasses with composition.

These refinements will preserve modularity while improving readability and reducing class-level dependencies.

4. Conclusion

The *Brewery Problem* design currently shows **balanced complexity and strong layering**, but the metrics highlight over-segmentation as the main inefficiency. Simplifying module boundaries and tightening orchestration will reduce complexity without sacrificing clarity. Group feedback emphasized that **simplicity and cohesion** are as critical as strict modular correctness in achieving sustainable design quality.