Assignment 2
Business Blueprint Allocation Heuristics

Software Architecture
Barber

Heuristic	Description/Motivation
Reduce Data/Event Dependency	Reallocate functions and data to reduce the coupling between components that result from the domain-level input/output dependencies between functions.
Specify Overlapping Capabilities (performer hierarchy)	Increase degree of inheritance by abstracting functions and data into parent components. Moving functions and data from two different components into a common parent component is one method for eliminating duplicate definitions and reducing coupling.
Reduce Class Complexity – Size	Reduce the number of functions and data in a given component which will likely reduce the complexity of technologies implemented from that component specification.
Reduce Class Complexity – Weights	Break up components that contain complex functions and data (where complex functions have many inputs/outputs and complex data are composed of many fields). Reducing a component's complexity will likely reduce the complexity of technologies implemented from that component specification.
Group based on Task Similarity	Improve cohesion by collecting functions that (i) use similar combinations of data or (ii) have the same parent function in the function decomposition.
Group based on Implementation Reality	To increase the likelihood that existing technologies will match component boundaries, collect functions based on the functionality existing technologies can provide.
Isolate risk – Functions	Collect functions associated with technology implementations that tend to change often.
Isolate risk – Data	Collect data associated with technology implementations that tend to change often.
Group based on Similar Capabilities (performer roles)	Collect functions based on performer roles.
Reduce Blueprint Complexity – Size	Reduce the number of components in the blueprint.

Reduce Blueprint Complexity – Abstraction	Create more abstract components in the blueprint, thereby increasing the depth of component inheritance hierarchies. Such hierarchies help show the relationships between components and the functions they offer.
Group based on Spatial Locality (tasks executed in same location)	Collect functions executed in the same location.
Group based on Data Usage Frequency	Identify the data most frequently used as function inputs/outputs, and collect functions into components based on different frequency levels.
Group based on Task Usage Frequency	Collect functions into components based on frequency of execution.
Group based on Resource Demand	Collect functions based on similar resource dependencies and the availability of those resources (e.g., if a resource is scarce, collect all functions requiring that resource into the same component to facilitate managing access to that resource).
Group based on Architectural Style – Client/Server based on Functions	Apply the client/server architectural style by collecting functions specified as needing to be "highly available" into a single component.
Group based on Architectural Style – Client/Server based on Data	Apply the client/server architectural style by collecting functions using data specified as needing to be "highly available" into a single component.
Leverage Task Parallelism	Place functions capable of being performed in parallel into different components.
Group based on Temporal Locality	Collect functions into components based on how often they are executed in the same scenario.