Criteria for modularity in concept design

criterion name	criterion description	evidence	example failing
Independence	Concepts are fully independent of each other, and can therefore be understood and used independently of one another.	Concept description limits any references to context of use to notes.	Concept purpose mentions the way in which the concept is intended to be used (eg, a payment concept whose purpose says "enables payment for magazine subscription").
		Concept does not refer to another concept by name.	Concept mentions working in concert with another (eg, session concept says "works with authentication concept to provide authenticated sessions").
		Concept does not rely on any properties of other concepts.	Concept action "calls" an action of another concept or queries the state of another concept.
		All external datatypes are either generic parameters or built-in types (such as String).	Concept treats arguments as objects that have been constructed elsewhere (eg, takes in a user object that is assumed to have a name field).
Completeness	Each concept provides a complete and coherent unit of functionality that delivers the value described in the purpose without the help of other concepts.	Concept functionality covers entire lifecycle of the purpose.	Concept doesn't include actions for set up (eg, defining available slots for reservations), or for closing down (eg, no deletion for an account).
		Concept embodies real functionality that fulfills a compelling purpose.	Concept is a data structure with CRUD actions when purpose calls for richer behavior (eg, concept holds contact info for a user but doesn't include any notification behaviors).
		Concept state is rich enough to support all the concept actions.	Concept state is expressed as the instance variables of a single object (eg, password auth concept that declares state as username and password, failing to support lookup by username needed to check password).
		Concept actions are sufficient to provide essential functionality to users.	No action to allow users to undo the effects of prior actions (eg, to cancel a reservation).
Separation of concerns	Concept does not conflate two concerns that could be broken into separate concepts that could be reused independently of one another.	All components of the state work together for a single purpose.	The state admits a factoring into two or more independent parts (eg, a user concept mixes preferences and profile fields).
		No state component can be dropped without compromising essential functionality.	The concept gratuitously includes state that is not needed to support actions (eg, a password authentication concept that stores, in addition to username and password, the date on which the user first joined).
		The concept does not include state components that could be easily expanded into much richer, self-contained structures.	The concept contains references to external objects and stores properties of them that are not needed for this concept (eg, references to users along with their names, which would better be stored in a separate profile concept).
		Concept represents at most one reusable and ideally familiar units of functionality.	The concept does not include a subpart that could easily stand by itself, and may even be familiar in its own right (eg, user concept includes karma points).
		The concept is balanced in the attention to behavioral detail.	The concept does not include a fragment of functionality that would in practice grow into a full and complex concept of its own (eg, a restaurant reservation concept including some details of table sizes, which would in practice belong to a concept that managed table layouts).