Overview of Concept Design

What does concept design offer?

- Shared language across roles (esp. UX/eng)
- Way to organize evolving design knowledge
- Strategy for aligning and integrating products
- Clarity and flexibility in user experience
- Structure for more modular implementation
- Reduced costs through greater reuse

What classic ideas do concepts build on?

- Separation of concerns and modularity
- Data models and actions for abstract behavior
- Conceptual models and user-centric computing
- Event-based invocation and mediators
- Type polymorphism and generics
- Christopher Alexander's design patterns

What new ideas do concepts bring?

- Concepts as behavior patterns not data entities
- Structuring function by user-facing behaviors
- Decoupling for full independent development
- Emphasis on reuse of familiar concepts
- Action sync as a composition mechanism

What related approaches are there?

- Domain-driven design: less modularity & reuse
- Microservices: often coupled and less granular
- Aspect-oriented: low level code
- Feature-oriented: features don't apply across apps
- Entity-component, role-oriented, subject-oriented: other attempts to fix OOP

Defining Concepts

What are the elements of a concept?

- *Name*: pithy and mnemonic, one word if possible
- *Purpose*: what motivated the concept, what value it brings
- *Actions*: performed by user or concept, read/write state
- *State*: a data model, what the running concept remembers
- *Principle*: archetypal scenario showing how it's used

What other parts of a concept catalog entry?

- Related: concepts similar or often used with this one
- Variants: common variants in design for different usages
- Challenges: what makes design hard, esp. human context
- Implementation: notes on implementation tricks
- Notes: comments, discussions, etc of evolving design

How to spot a good concept definition?

- Compelling purpose: reflected in name and principle
- Strong actions: meaning and effect of each action clear
- Rich state: non-trivial data model for valuable function
- Free standing: easily understood without other concepts

Common Concept Design Flaws

Concept design flaw #1: Conflation

- Issue: A concept should not conflate separable functions
- Example: User concept that conflates authentication, profile info, preferences
- How to fix: Separate into multiple concepts (eg, Password, Profile, Preference)

Concept design flaw #2: Fragmentation

- Issue: A concept includes only part of a function, doesn't deliver full value
- Example: Notification concept can send messages but doesn't support user's selection of channel
- How to fix: Expand the concept so it supports the function completely, end to end

Concept design flaw #3: Coupling

- Issue: A concept depends on details of another concept so it can't be used without it
- Example: Team concept aggregates users (OK), but assumes users have names (bad)
- How to fix: Ensure that all references to external objects are fully generic; expand state if necessary

Concept design flaw #4: Not user facing

- Issue: A concept contains details not relevant to users
- Example: Replication concept models how system replicates data for availability
- How to fix: Remove concept and put details into architectural or implementation notes

Concept design flaw #5: User interface coupling

- Issue: A concept contains details too tied to user interface
- Example: Password concept has separate actions for entering username and password
- How to fix: Define actions that abstract over multiple user interface steps

Concept design flaw #6: Missing actions

- Issue: Some actions are omitted that are essential to the full concept behavior
- Example: Reservation concept has actions for reserving but not for creating reservable slots
- How to fix: Expand the operational principle and add the necessary actions

Concept design flaw #7: Inadequate state

- Issue: State not rich enough to support specified actions
- Example: Upvote fails to track which user upvoted for an item, so double voting can't be stopped
- How to fix: Write careful specs of each actions and expand the state accordingly

Concept design flaw #8: Not invented here

- Issue: Concept invents new way of doing something familiar, for no good reason
- Example: Powerpoint's Section concept, a complex mechanism for a familiar purpose
- How to fix: Use a familiar and well-known concept instead

Concept design flaw #9: Crossing boundary

- Issue: Concept includes actions that should belong to another concept (minor conflation)
- Example: Reservation concept includes sending notifications to users
- How to fix: Include additional concept and synchronize

Concept design flaw #10: Observer action

- Issue: Concept includes actions that just query the state and aren't necessary
- Example: Comment concept includes action getAllCommentsByUser
- How to fix: Just drop such actions; synchronizations can query the concept state

Bad Smells

Bad smell #1: Single object

- Symptom: State is defined like fields of an object
- Example: User concept defines state as username, password, email
- Indicates: Not understanding what concepts are, and how they differ from objects
- Consequences: Doesn't handle set of objects; often OOP-induced conflation and fragmentation

Bad smell #2: Vague purpose

- Symptom: Purpose is vague and expansive
- Example: Printing concept says purpose is to handle "printing of documents"
- Indicates: Not separating concerns enough
- Consequences: Conflation, needless complexity, lack of clarity and reusability

Bad smell #3: UI component

- Symptom: Defines a user interface component
- Example: Settings concept, with features of many concepts (Password, Profile, Preference, Session)
- Indicates: Thinking about UI, not underlying function
- Consequences: Concept design is needless complicated and too tied to layout design

Bad smell #4: Entity

- Symptom: Define a concept for each entity in an ontology
- Example: Banking system has Account, Transaction, Withdrawal, Deposit
- Indicates: Not understanding concepts are behavioral
- Consequences: Proliferation of many concepts with little value and high coupling

Bad smell #5: Phase

- Symptom: Concept defines a phase of behavior that brings no value in itself
- Example: LoadParameters concept that loads some parameters but doesn't use them
- Indicates: Thinking of behaviors as steps in a computation rather than projections
- Consequences: Resulting concepts will be fragmented and coupled to others