SENG5802: Software Engineering II - Software Design
Domain Modeling and Ubiquitous Language

Outline

Outline

Layered Architecture

- Focus on separation of concerns
- Why is the application separate from the domain?
- Why is domain separate from infrastructure?

Common Layered Archs.

What is the typical layering among common architectures?

- Client-server
- Query-mostly web apps
- Java EE/ROR/Grails
- AJAX apps
- Service-oriented architecture

Buschmann on Layered Architecture

- Large-scale systems, decomposed
- Coupling is <u>strictly top–down</u>
- Lower level replaceable
- Multiple upper layers can use common lower layer
- Process to define

Layers of Abstraction

- A model has classes and subclasses
- An architecture has (among other things) layers
- A design/architecture groups layer elements into modules
- Modules usually map to packages

Connecting layers

- The general rule: elements in layer n (with 1 on the bottom) can depend only on layers <= n
- So, how do you communicate “upward”?
- If the application requires system initiative, then a reactive design may be a poor choice.

Outline

Using language (1/2)

- Nouns:
  - Things, places, parties
  - Roles
  - Services, operations
  - Relationships
  - Rules
  - Events

Using language (2/2)

- Verbs
  - Steps in a process
  - Operations on or by objects
- Modifiers & qualifiers
  - May indicate rules or options
  - May constrain relationships: ordering, cardinality, quality, mapping
- How do you define a term?
  - Diagrams, explanatory text, domain artifacts
  - Relationships to other items in the ontology

Ontology – formalized meaning

"A systematic arrangement of all of the important categories of objects or concepts which exist in some field of discourse, showing the relations between them."

– GNU version of the Collaborative International Dictionary of English

Ontology – no really, meaning

- It's about language and meaning. An ontology is a fancy glossary.
- To a philosopher, ontology is the study of existence, its categories and the relationships among them.
- To a linguist, ontology connects words with mental concepts and their referents in the world.

Components of an Ontology

Two primary components

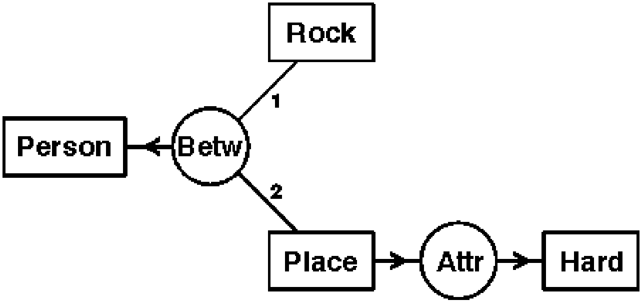
- Names for important concepts in the domain
- Background knowledge/constraints on the domain

Categories

- Description logic
- Taxonomy
- Inheritance
- Disjoint subclasses
- Exhaustive decomposition
- Partition

Conceptual Graphs

Organize knowledge into concepts and relations.



Outline

MDD: Requirements

- Starts at BEGINNING of requirements-gathering
- Requirements are necessarily expressed with respect to a model.
- Model tested and improved through usage scenarios
- Result is initial domain model and vocabulary

MDD: Design & development

- Architecture determines application structure
- Initial domain model is refined, integrated into architecture
- Iteration planning driven by model and priorities
- Incremental model detailing, development, and testing

Agile modeling principles (1/2)

- Model with purpose
- Maximize stakeholder investment
- Embrace change
- Refactor as you go
- Use multiple models
- Travel light
- Goals
  - Working software
  - Enabling the next iteration

Agile modeling principles (2/2)

- Focus on quality work
- Organize for rapid feedback
- Assume simplicity
- Content over representation
- Everyone can learn
- Use the simplest tools
- Model with others
- Prove it with code

– Scott Ambler

Building a domain model

- Understand and document user's goals and success criteria
- As you develop ubiquitous language, revise and refactor
- Requirements, model, and code must use the same language.
    - If it's wrong, change it.

Modeling process

- CRC cards are helpful in the early stages
- Test the model
- Don't let inconsistency and fuzziness slide
- Know when to stop

Outline

Battleship

- With your project groups
- Begin the process of building a domain model for a program to play the game of “Battleship”.

Outline

Domain model details

Good domain models have transparent, predictable structure

- Entities, Values, and Services — Evans
- Archetypes mapped in well-understood way — Coad

Entities

If it's a persistent object with possibly varying attributes,

it's an Entity (or reference object).

Values

If it's a description of an Entity, or if it's a data holder or message,

it's a Value.

Services

If it’s a set of operations not obviously associated with any particular Entity or Value and (ideally) has no state,

it’s a Service

Domain Services

What sorts of services do we expect to see in the domain model?

Coad's archetypes

- Party, Place, Thing – things (often entities) that can play a role in the domain or system
- Moment-interval – activities or events that are located in time or extend over an interval.
- Role – a way that something (usually a PPT) participates in a M–I.
- Description – a value or set of values that label a thing or set of things (non-identifying)

Coad Discussion

Coad claims that the addition of color "increases the amount of content we can express."

Does color make the diagrams more expressive?

What if you could add or remove colors, or change their meaning?

Modeling associations

Bidirectional associations

- Don't work so well in code
- Make sure you fully understand domain and application requirements
- Use implicit associations for the less-traveled route

Assigning Responsibility

- Begin with the clear operations on Entities and Values
- Operation on multiple objects? Sounds like a service.

Outline

Exercise

Work out a more detailed domain model for Battleship

Goals:

- Understand the game
- Agree on vocabulary
- Find Entity and Value objects
- Find Service objects

Outline

Jack Reeves

Software Design

Martin Fowler

Domain Model and the Service Layer