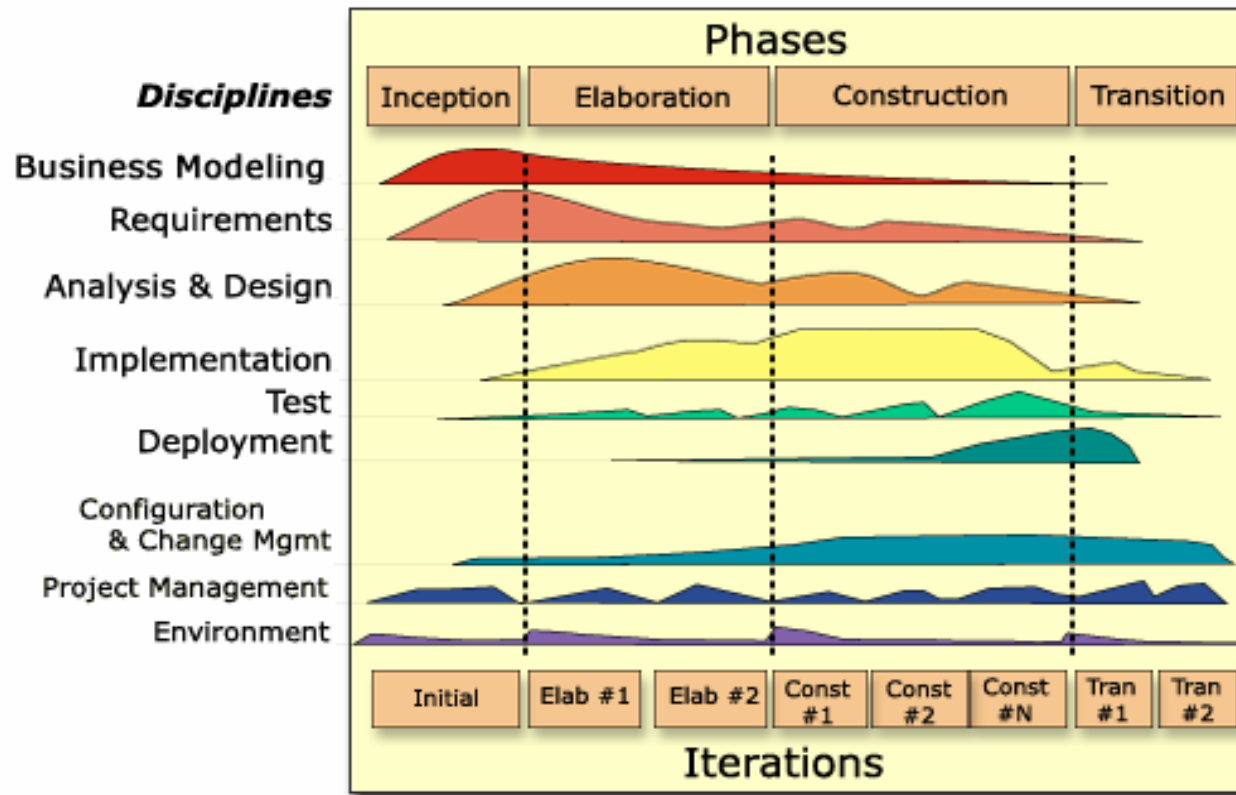


Lecture 2: Elaboration Tasks and Domain Modeling

Rational Unified Process



Elaboration Tasks

- Explore concepts from requirements/use cases
- Domain models: basic concepts and their inter-relationships
- Basic architectural concepts for system
- System sequence interactions – system events, UI basics

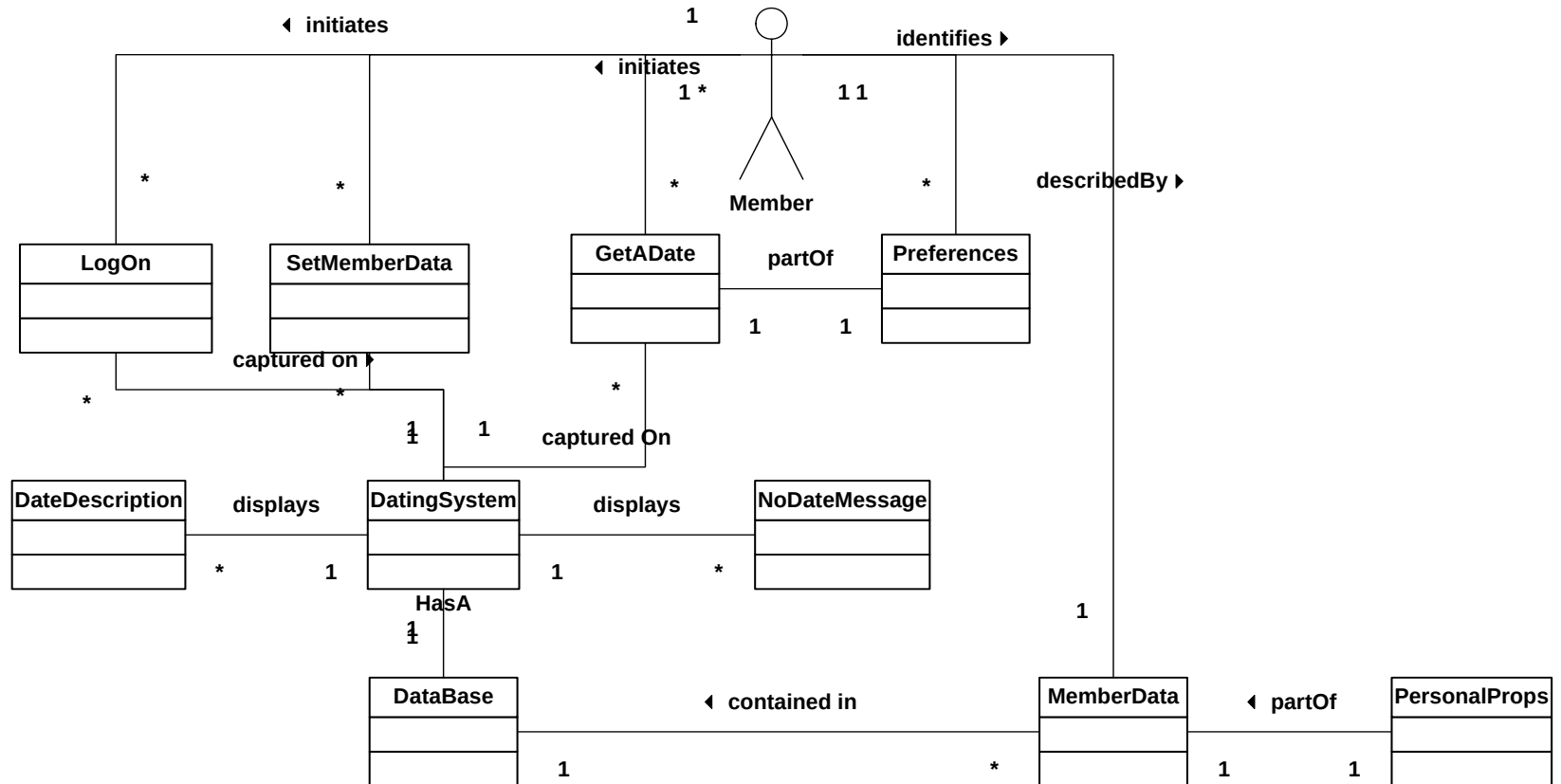
Stable System Design and Domain Models

- Systems change
 - Incremental development
 - Changing requirements
 - Post deployment enhancement
- Design for change
 - Simulation of problem domain
 - Functionality added to simulation

Domain Models

- Graphical Model
- Nodes:
 - Concepts/conceptual classes
 - Attributes: properties of classes
- Arcs: Relationships between concepts
- Particular kind of UML Class Model

Sample Partial Domain Model



Domain Model Does and Don'ts

- ✓ Static Model
 - E.g. Parts explosion diagram
- ✓ Real World Concepts
 - E.g. Gender Preference in Dating System Application
- ✓ Overdo concept list
- ✗ Dynamic Flow Chart
 - One entity sends a message to another
- ✗ Software Entities
 - E.g. GenderPreferenceTextField in DS
- ✗ Leave out concepts

Domain Models and Data Base Models

- Entity Relationship Diagrams
 - Similar to domain modes
- Used to identify:
 - Tables = Concepts
 - Columns/fields = Concept attributes
 - Table links = Concept relationships

Finding Concepts

- Define the boundary of the system
- Noun and noun phrase identification from prose descriptions and use cases
- Concept category checklists
- Special cases

Nouns and Noun Phrases for DS

- User
- Dating System
- LogOn
- Name
- Member
- Member Option
Choice
- Error Message
- GetADate
- DataBase
- SetMemberData
- Preferences
- DateDescription
- NoDateMessage
- MemberData
- PersonalProps

Concept Categories and the DS

- Physical Objects: user, member
- Descriptions: personal props, preferences
- Roles: member, dater, administrator
- Containers: DS Data Base
- Organizations: DS Accounting Dept.
- Events: logon, getADate, SetMemberData
- Abstract nouns: session

Descriptions

- Descriptions of other concepts that have a lifetime of their own
 - E.g. dater properties in Dating System
 - May be stored in data base
 - May be changed
- Alternative is to make the information an attribute of a concept
 - primitive class attributes vs class variables

Events

- Things to which system will have to respond
 - E.g. logon, date-request
- Object oriented approach to events: create an instance of an event object which “knows how to process such events”
 - E.g. Java GUI:
 - create event listener objects which have a method for processing the associated event
 - add the listeners to the object that is the source of the event
 - when event occurs, listener objects are “executed”

Associations

- Describe semantically meaningful relationships between concepts
- Give the structure of the domain model
- May be annotated with roles and multiplicity designators
- Important to not miss concepts
- Important to not over-do associations

Finding Associations

- Need to know criteria
 - Needed for operation of system
 - Have a lifetime of importance
 - E.g. link between DS Member and Account is necessary.
 - E.g. link between Administrator and GetDate request is not necessary.
- Association category checklists

Association Category Checklists

- A is a physical or logical part of B
 - E.g. Physical Characteristics and Personal Characteristics are part of Dater Properties
- A is physically or logically contained in B
 - E.g. MembershipData contained in MemberDB
- A is recorded in/by B
 - DatingSystem records occurrence of User Asking for a Date event

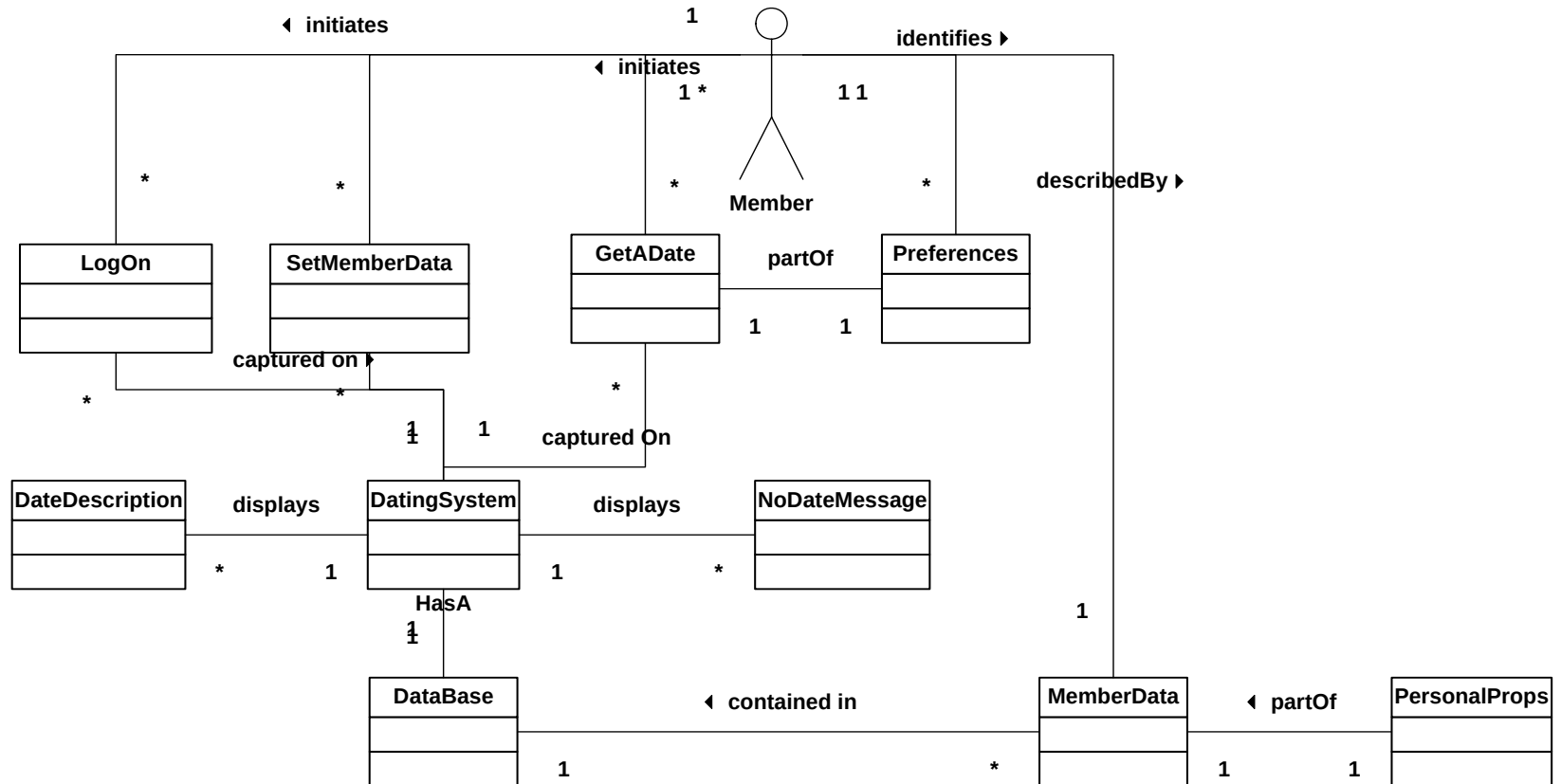
Associations and Class Design

- Logical Parts
 - Class Variables: the objects in one class consist of an aggregation of objects from other classes
- Contained in
 - A vector or other container class contains objects of one or more other classes

Association Details

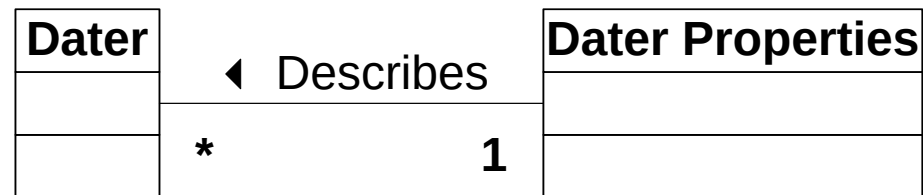
- Names
- Multiplicity
 - One to one, many to many, one to many
- Direction of association

Domain Model Example



Attributes

- Problem domain properties
- Should have primitive data type values
- If property is a (complex) object, should be a separate concept with a linking association
 - E.g. Dater has Dater Properties. This is a complex object and should be its own concept. Linking association:



Associations vs Attributes –

- Has Parts to it
- Special operations associated with it
- Has its own attributes
- Quantity with a unit
- Abstraction of an entity with above properties
- Standard data types (integer, string, ...)
- Quantity with a unit
- Note: use of foreign keys (primitive key value used to identify a complex object) not acceptable

Associations vs Attributes – Why Associations?

- Allows postponing of details
- Leads to a design that facilitates change
- Recognizes that some kinds of properties/descriptions may have a life of their own

E.g. In a complex DS could have a catalog of personality profiles, which we use to classify a member/user

Domain Models and Attributes

- Not enough room in graphical models
- Give separate documents showing domain model concepts with their attributes
- E.g.

Logon
-Name: : string
-UserType : char

SetMemberData
-Gender : bool
-Religion : string
-Occupation : string

UML and Domain Models

- UML does not have Domain Models
- UML class diagrams
 - can be used for several purposes
 - Conceptual classes as in Domain Model
 - Attributes but no methods
 - Design classes from design activities
 - Graphical representation of PL classes

Domain Models and System Increments

- Construct for current iteration concepts, associations and attributes
- Will augment with later increments
- May include concepts not in current increment as part of “extra is better than missing” approach