



MSc in Official Statistics
Statistical Computing:
UML Modelling

Andrew Westlake

Survey & Statistical Computing

63 Ridge Road, London N8 9NP, UK

+44 (0) 20 8374 4723

AJW@SaSC.co.uk (E-Mail)

www.SaSC.co.uk

Why do we model?

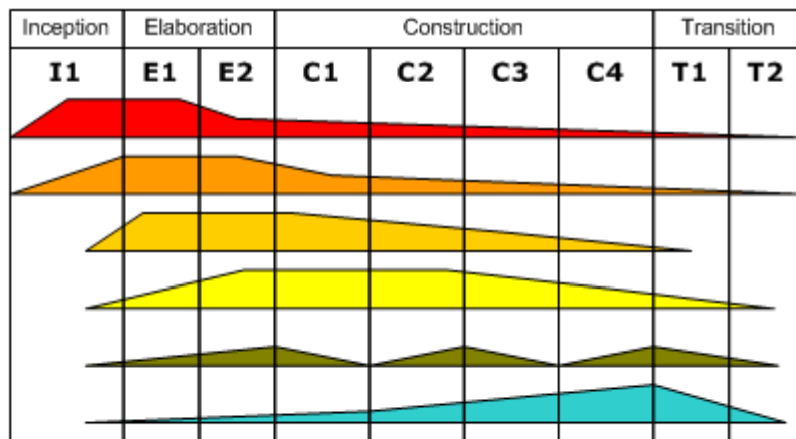
- Provide structure for problem solving
- Experiment to explore multiple solutions
- Furnish abstractions to manage complexity
- Reduce time-to-market for business problem solutions
- Decrease development costs
- Manage the risk of mistakes

- Important to be as formal as possible
 - » So that model can contribute directly to implementation
- Must retain flexibility and expressiveness

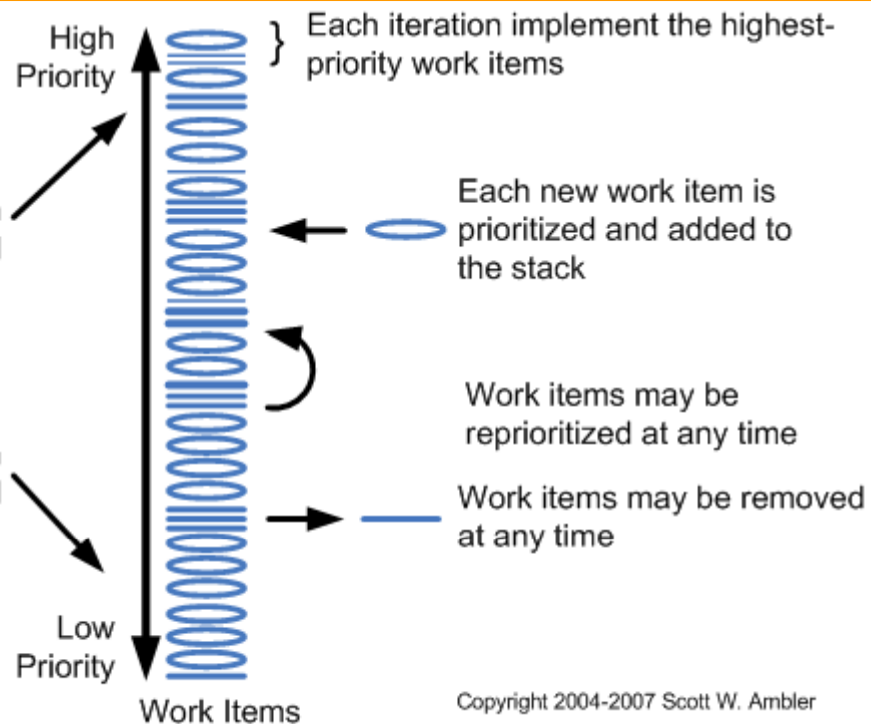
Incremental Development

Iterative Development

Business value is delivered incrementally in time-boxed cross-discipline iterations.



Time →



Quick Tour

- The UML is a graphical language for
 - » specifying
 - » visualizing
 - » constructing
 - » documentingthe artifacts of systems, particularly software
- Added to the list of OMG adopted technologies in November 1997 as UML 1.1
- Current version is UML 2, stable in 2005, some refinements now in progress (on 2.3)

UML for Statisticians

- Useful set of tools and notations for specifying requirements and communicating with IT staff
- Formal definitions
 - » Gives precision if understood and used correctly
- Not trivial to learn, but based on sound and familiar principles
 - » Don't need to learn it all
 - » See Practical UML and SDMX UML Tutorial in Handouts folder

UML Approach

- Based on Object (O-O) approach to design
 - » Built from components and objects
 - » Time aspects for long-lasting actions
 - » Communication between components
 - » Sequence and state
 - » More than Entity-Relationship plus Flowcharts
- Example:
 - » Database system for PHLS HIV and AIDS Reporting section - Use Case and Activity

UML Software

- Rational Rose (plus tools for analysts, developers, etc)
 - » Market leader, bought by IBM
 - » Code generation facilities, integrates with programming environments
 - » Repository extensions, e.g. for management of requirements
- Together
 - » Very strong on round-trip engineering in Java, merged with Borland
- MS Visio
 - » General diagramming tool (very good), with extensions for UML and ER modelling
 - » Some code generation in Enterprise version
- Poseidon
 - » Based on OpenSource Argo project from UCLA
 - » Has free Community Edition
- Objecteering
 - » Commercial system, but with free unlimited evaluation version
- Other Open Source initiatives, focussed around the Eclipse IDE

UML Goals

- Define an easy-to-learn but semantically rich visual modeling language
- Unify the Booch, OMT, and Objectory modeling languages
- Include ideas from other modeling languages
- Incorporate industry best practices
- Address contemporary software development issues
 - » scale, distribution, concurrency, executability, etc.
- Provide flexibility for applying different processes
- Enable model interchange and define repository interfaces



Building Blocks

- The basic building blocks of UML are:
 - » model elements (classes, interfaces, components, use cases, etc.)
 - » relationships (associations, generalization, dependencies, etc.)
 - » diagrams (class diagrams, use case diagrams, interaction diagrams, etc.)
- Simple building blocks are used to create large, complex structures
 - » cf. elements, bonds and molecules in chemistry
 - » cf. components, connectors and circuit boards in hardware

Well-Formedness Rules

- Well-formed:
 - » Indicates that a model or model fragment adheres to all semantic and syntactic rules that apply to it
- UML specifies rules for:
 - » naming
 - » scoping
 - » visibility
 - » integrity
 - » execution (limited)
- However, during iterative, incremental development it is expected that models will be incomplete and inconsistent

UML Diagrams (1)

- At the centre of the UML are three types of modelling diagram
 - » Structure Diagrams
 - Class, Object, Component, Composite Structure, Deployment diagrams
 - » Behaviour Diagrams
 - Activity, Use case, State Machine Diagrams
 - » Interaction Diagrams
 - Sequence, Interaction Overview, Communication, Timing Diagrams

UML Diagrams (2)

- Structure diagrams: the static structure of the objects in a system
 - » That is, they depict those elements in a specification that are fixed over time
 - » The elements in a structure diagram represent the meaningful concepts of an application, and may include abstract, real-world and implementation concepts
 - » For example, a structure diagram for an airline reservation system might include classifiers that represent seat assignment algorithms, tickets, and a credit authorization service
 - » Structure diagrams do not show the details of dynamic behaviour (covered by behavioural diagrams). However, they may show relationships to the behaviours of the classifiers exhibited in the structure diagrams
 - » Can be seen as a generalisation of Entity-Relationship diagrams
- Behaviour diagrams: the dynamic behaviour of the objects in a system
 - » Including their methods, collaborations, activities, and state histories
 - » The dynamic behaviour of a system can be described as a series of changes to the system over time

What is structural modeling?

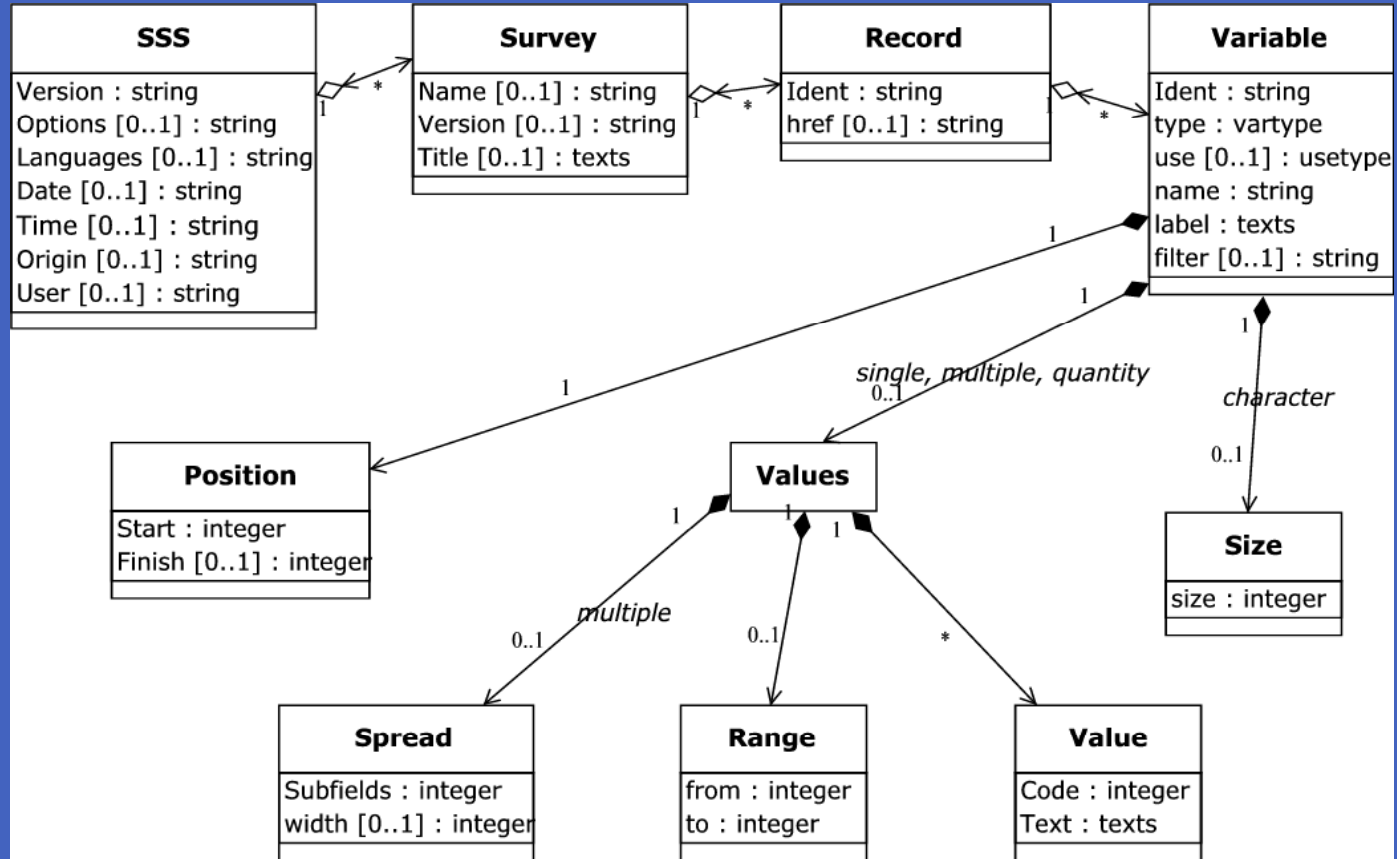
- Structural model:
 - » a view of an system that emphasizes the structure of the objects, including their classifiers, relationships, attributes and operations

Class Diagrams

- Show static structure of Classes
 - » Abstractions of Object instances
- Similar to Entity - Relationship
 - » Presentation of structure
- Classes are richer
 - » Attributes and Methods (behaviour)
- Relationships are richer
 - » Many types
- Not so close to implementation

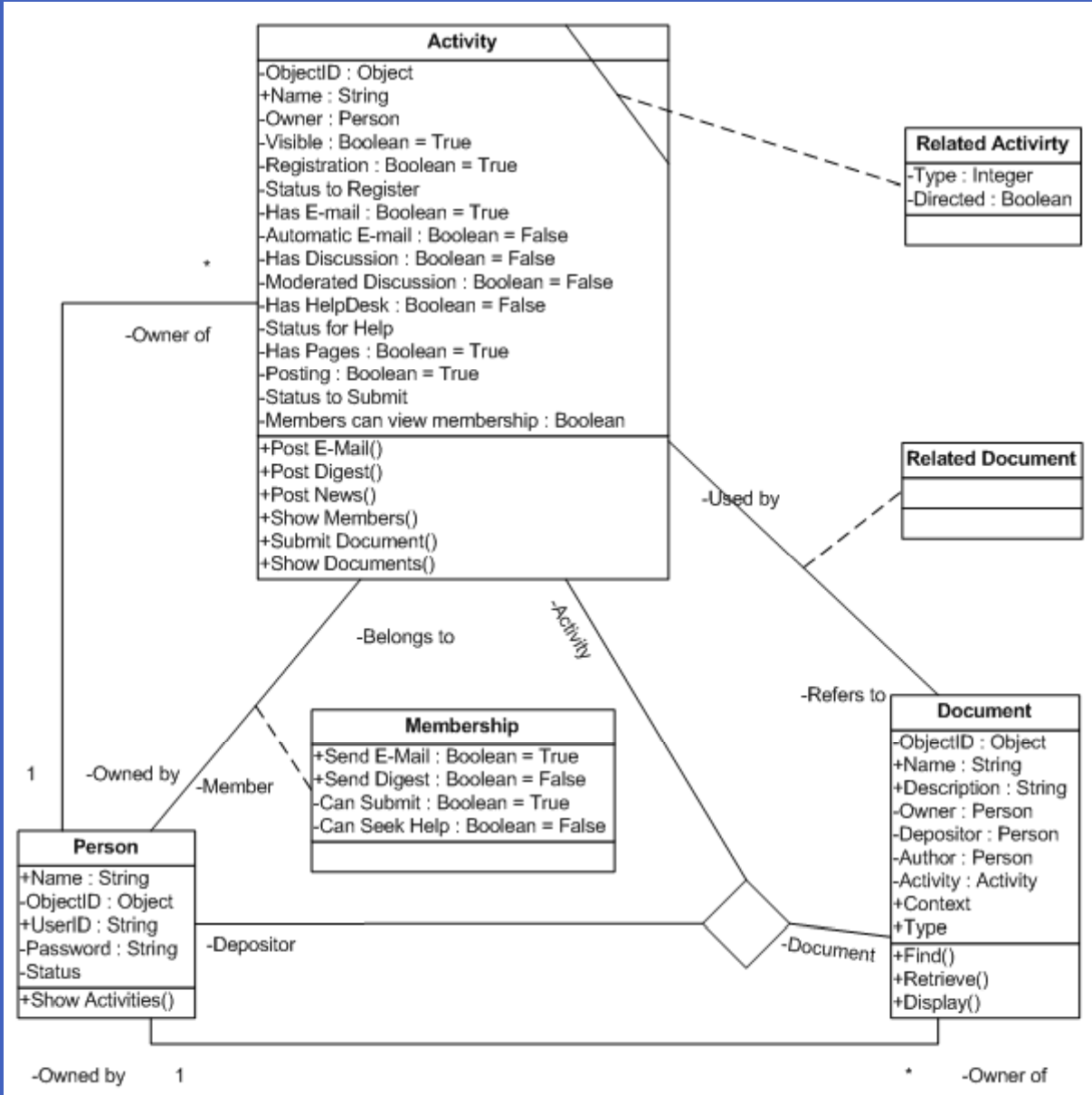
Class Diagram of Triple-S

- Only composition associations used
- Constraint semantics included
- No methods in this example







Class Diagram for AMRADS Repository

- Includes methods and cardinality
- Associations can be classes and have attributes
- Do not bother to elaborate many to many associations
 - » Would be implementation dependent



Structural Modeling: Core Relationships

Construct	Description	Syntax
association	a relationship between two or more classifiers that involves connections among their instances.	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
generalization	a taxonomic relationship between a more general and a more specific element.	
dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	

Structural Modeling Tips

- Define a “skeleton” (or “backbone”) that can be extended and refined as you learn more about your domain
- Focus on using basic constructs well; add advanced constructs and/or notation only as required
- Defer implementation concerns until late in the modeling process
- Structural diagrams should
 - » emphasize a particular aspect of the structural model
 - » contain classifiers at the same level of abstraction
 - » Same classes can appear in different diagrams to show different aspects (views) of the system
- Large numbers of classes should be organized into packages



When to model structure

- Adopt an opportunistic top-down + bottom-up approach to modeling structure
 - » Specify the top-level structure using “architecturally significant” classes and model management constructs (packages, models, subsystems)
 - » Specify lower-level structure as you discover detail about classes and relationships
- If you understand your domain well you can frequently start with structural modeling
- Otherwise
 - » If you start with use case modeling (as with a use-case driven method) make sure that your structural model is consistent with your use cases
 - » If you start with role modeling (as with a collaboration-driven method) make sure that your structural model is consistent with your collaborations

Use Case Modeling

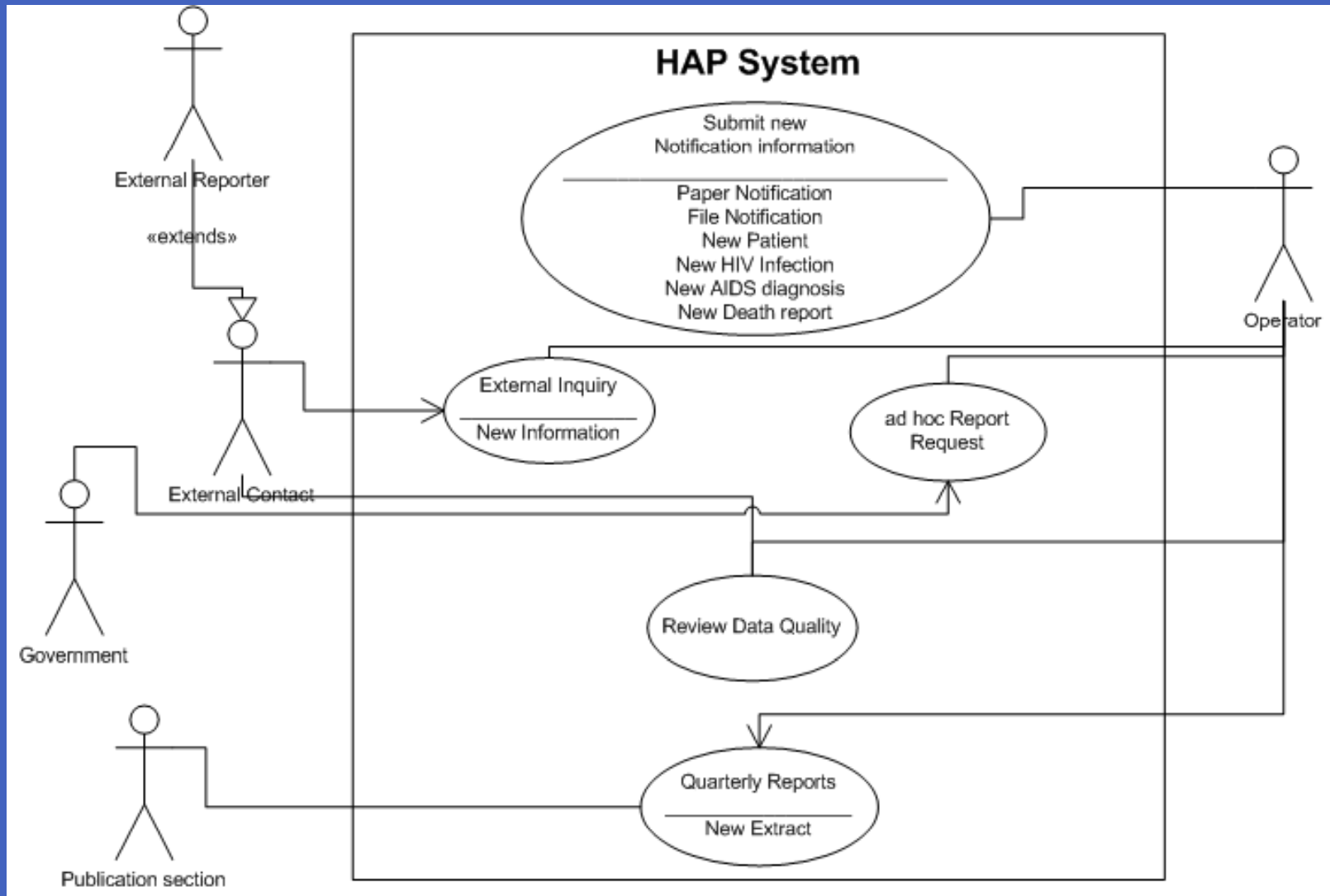
- What is Use Case modeling?
- Core concepts
- Diagram tour
- When to model Use Cases
- Modeling tips
- Example:
 - » HAP system

What is Use Case modeling?

- The word Use is a noun
- Use Case model:
 - » A view of a system that emphasizes the behavior as it appears to outside users
 - » Partitions system functionality into transactions ('use cases') that are meaningful to users ('actors')
 - » Combination of diagram for structure and text for description of processes
- Important tool for capturing user requirements
 - » Diagrams communicate understanding
 - » Form a basis for testing
- As with all of UML, can be used in various ways

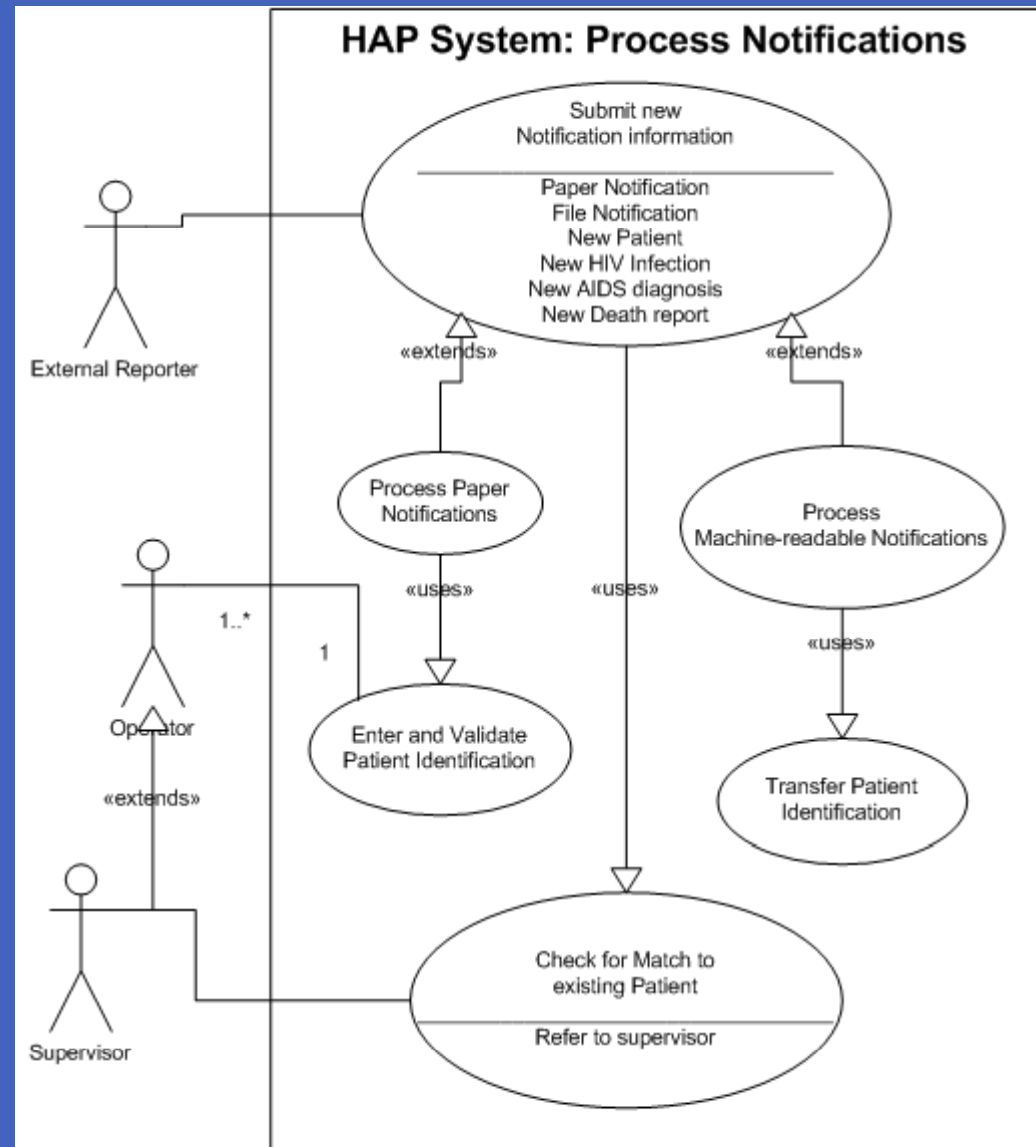


Use Cases for HAP



Notifications for HAP

- Reporting organisation can use paper or send files
 - » Information on paper must be entered and verified (separate operators)
 - » Files need to be transformed before import
- Patient Identification information compared with existing Patients and matches scored
 - » If no match, create a new Patient and transfer the information
 - » If match, compare new with existing information
 - » If unclear, refer to supervisor
- Enter information about the notification
 - » May require new record for the Notification type
- Needs to be backed up with details of the processes - see in Visio



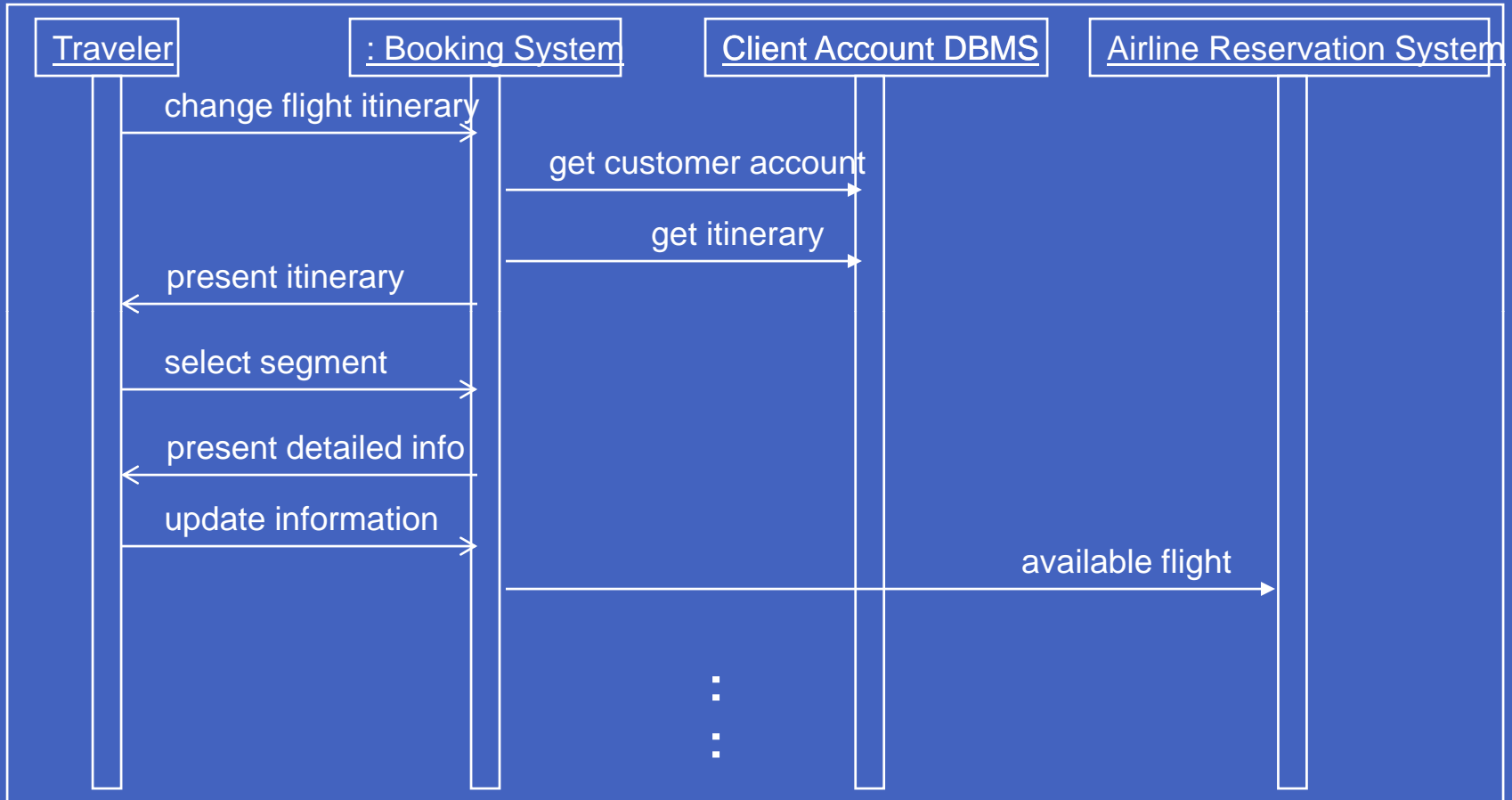
Alternative representations

- Use Case Diagram
- Use Case Description
- Sequence Diagram
 - » Emphasises temporal aspects
- Collaboration Diagram
 - » Flow of information between components
- Activity Diagram
 - » Emphasis on flow of control through activities

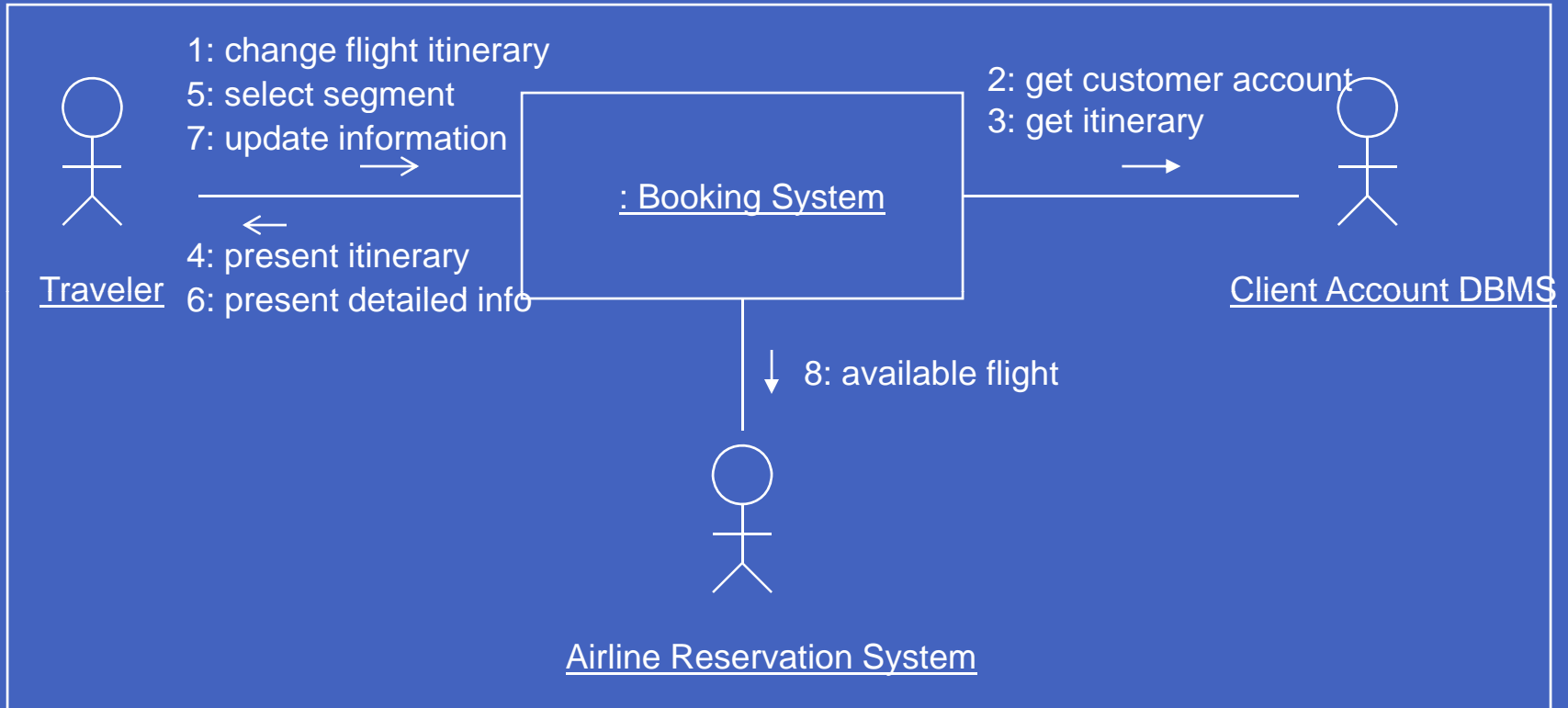
Use Case Description: Change Flight

- Actors: traveller, client account db, airline reservation system
- Preconditions:
 - » Traveller has logged on to the system and selected 'change flight itinerary' option
- Basic course
 - » System retrieves traveller's account and flight itinerary from client account database
 - » System asks traveller to select itinerary segment she wants to change; traveller selects itinerary segment.
 - » System asks traveller for new departure and destination information; traveller provides information.
 - » If flights are available then
 - » ...
 - » System displays transaction summary.
- Alternative courses
 - » If no flights are available then ...

Sequence Diagram: Change Flight Itinerary

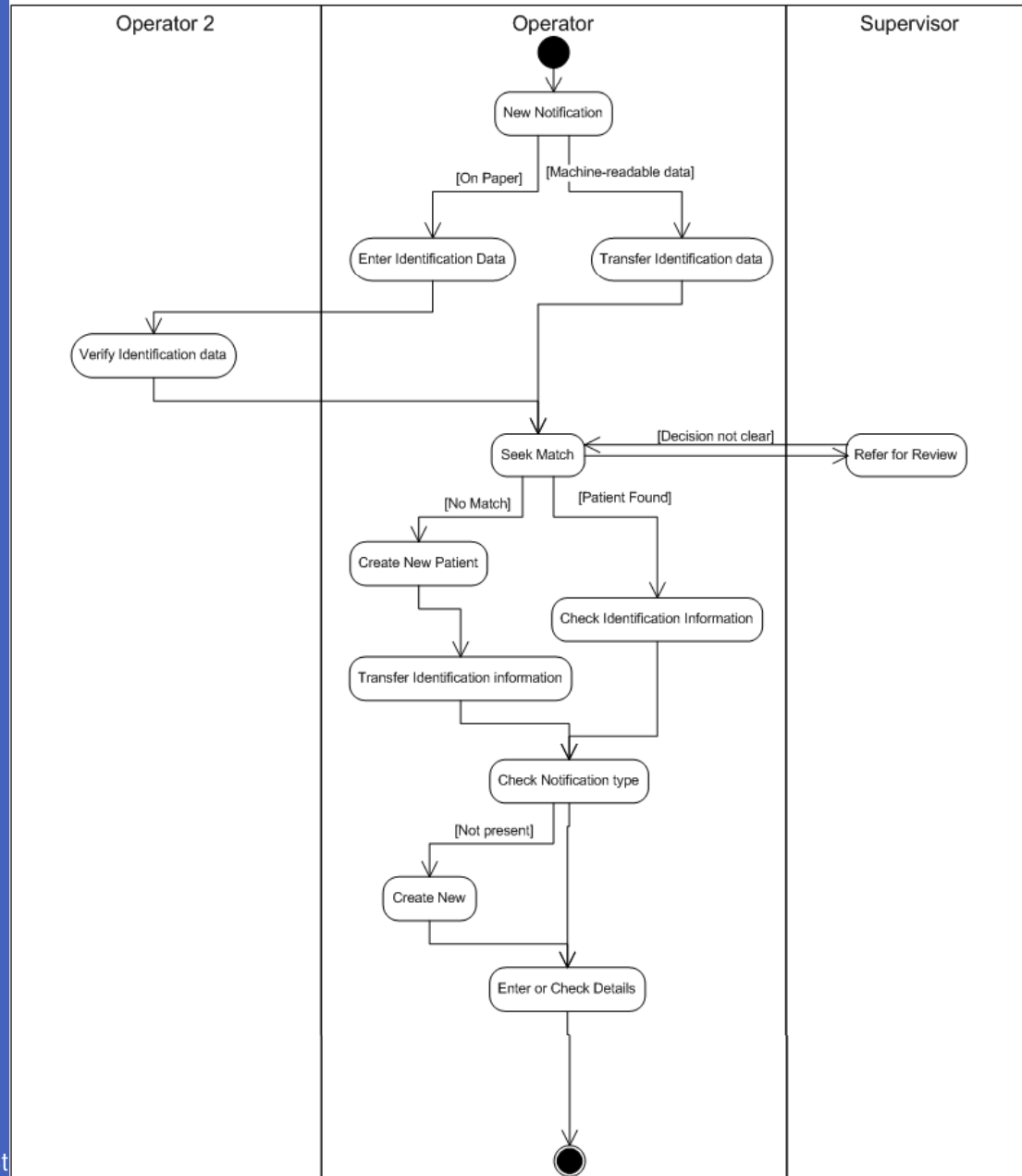


Collaboration Diagram: Change Flight Itinerary



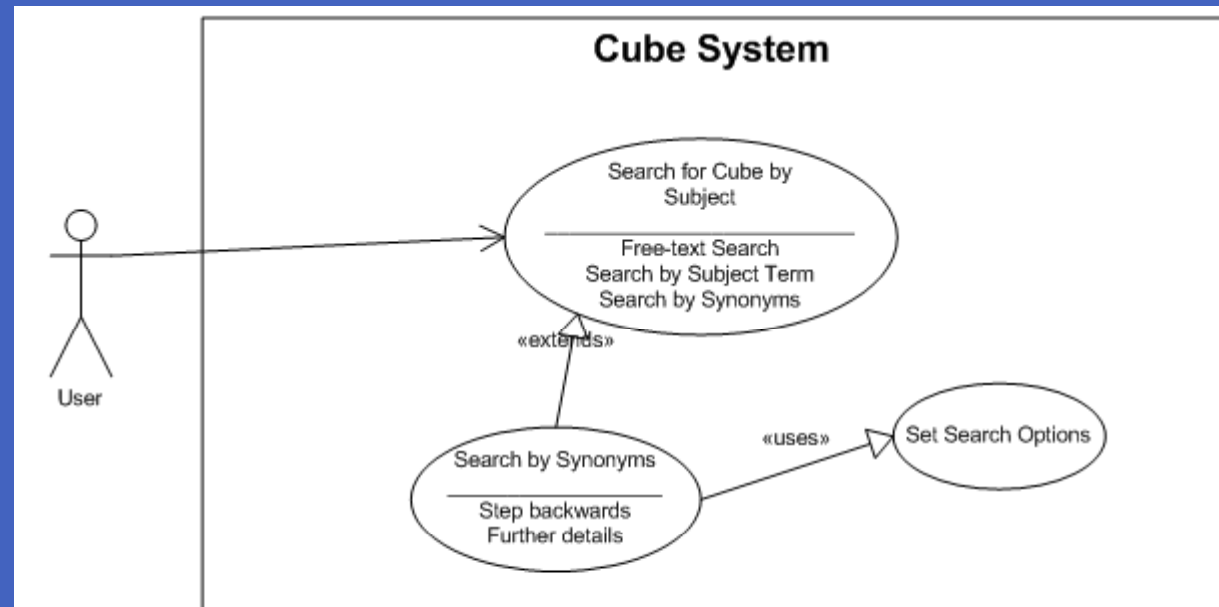
Activity Diagram: HAP Notification

Enter New Notification



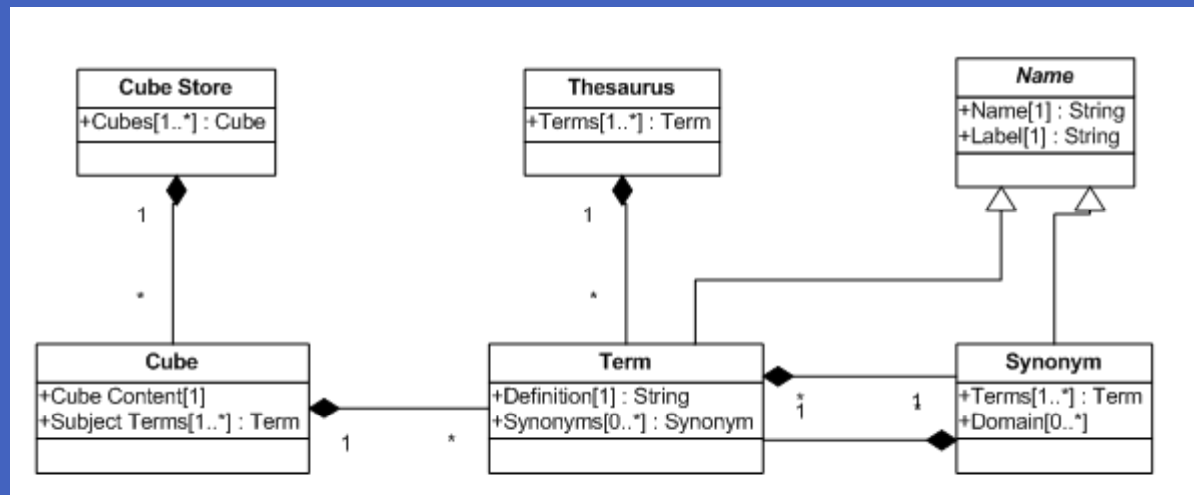
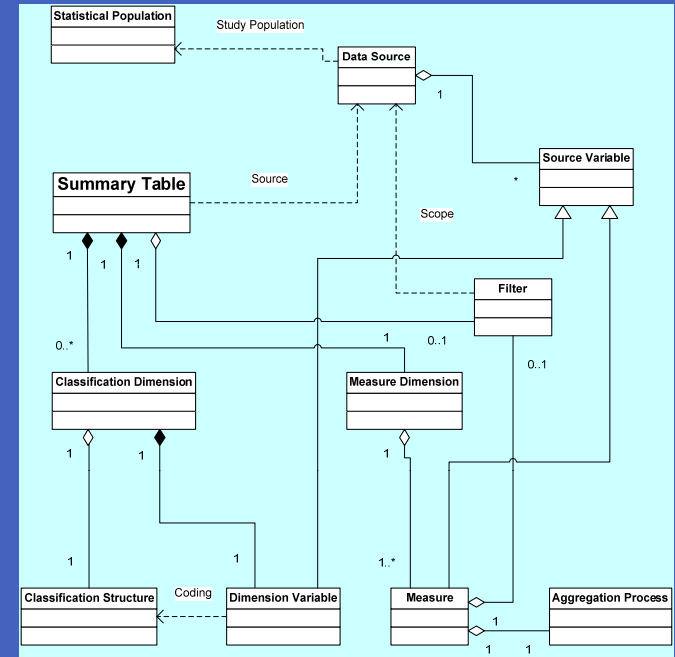
Searching for Cubes - requirement

- Search over Cubes for Subjects of interest
 - Requirement is to be able to use terminology from the user's own domain
 - In addition to standard Terms and content
 - Introduce Synonyms



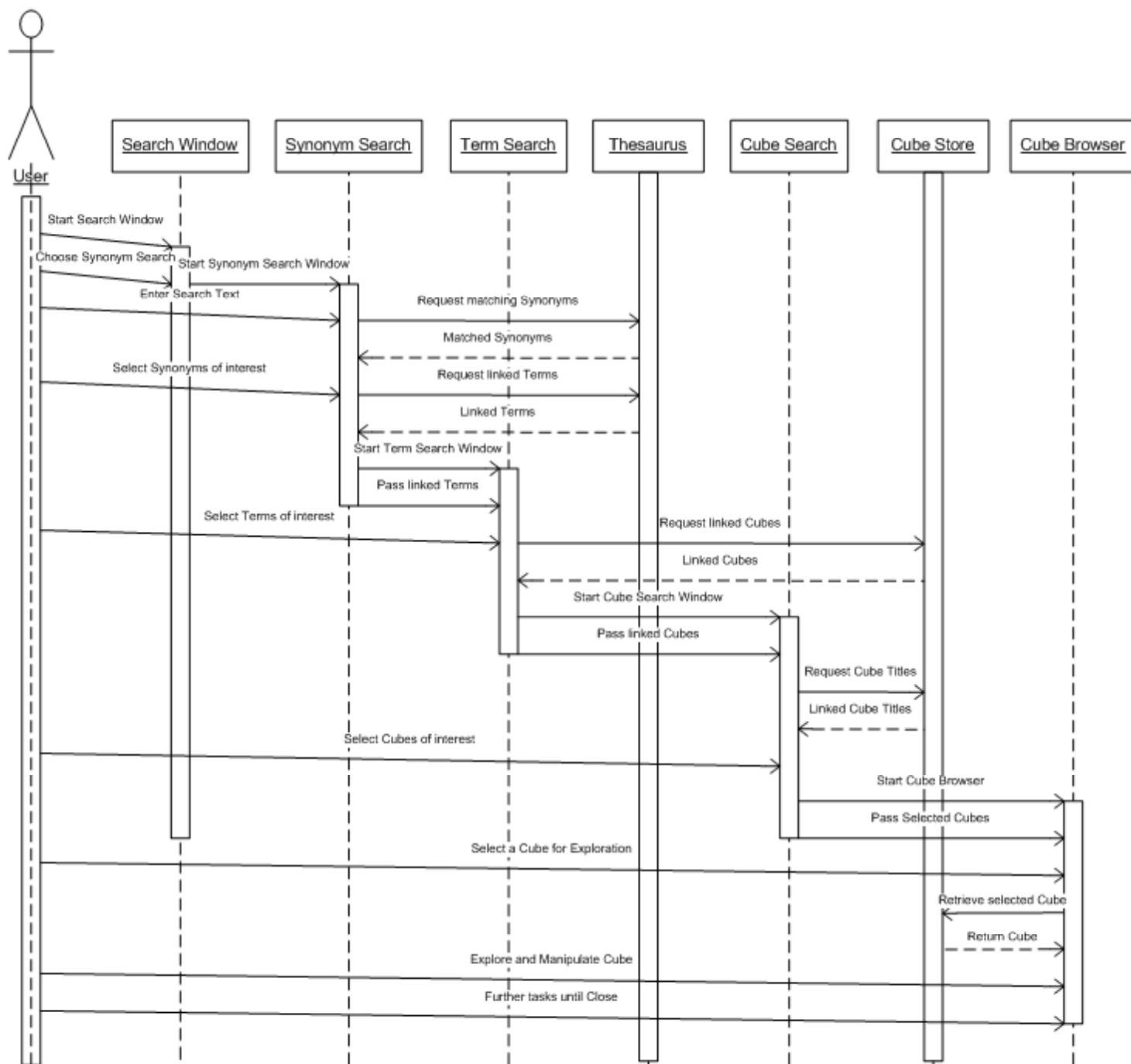
Searching for Cubes - Structure

- Extend structure to introduce Synonyms for Terms
 - Terms from a Controlled Vocabulary
 - Different Terms can have the same Synonym (in different Domains)
 - Terms and Synonyms are specialisations of a more general Name class, so can share methods



Searching

- Messages between components of the system
- Periods when components are active



Use Case Modeling Tips

- Make sure that each use case
 - » describes a significant chunk of system usage
 - » and is understandable by both domain experts and programmers
- When defining use cases in text
 - » use nouns and verbs accurately and consistently
 - » helps derive objects and messages for interaction diagrams
- Factor out common usages that are required by multiple use cases
 - » If the usage is required use <<uses>>
 - » If the base use case is complete and the usage may be optional, consider using <<extend>>
- A use case diagram should
 - » contain only use cases at the same level of abstraction
 - » include only actors who are required
- Large numbers of use cases should be organized into packages

When to model use cases

- Model user requirements with use cases
 - » Use to support a requirements methodology
- Model test scenarios with use cases
- If you are using a use-case driven method
 - » start with use cases and derive your structural and behavioral models from it
- If you are not using a use-case driven method
 - » make sure that your use cases are consistent with your structural and behavioral models

Activity Diagram Applications

- Intended for applications that need to control flow or object/data flow models ...
 - » Similar to flow charts
 - » Can involve external inputs, but in a controlled sequence
- ... rather than event-driven models like state machines
 - » Which respond to asynchronous external events
- For example: business process modeling and workflow
- Useful for elaborating Use Cases

Ideas to Take Away

- UML is effective for modeling large, complex software systems
- Not just for software, can be used for any system
- It is simple to learn for most developers, but provides advanced features for expert analysts, designers and architects
- It can specify systems in an implementation-independent manner
- 10-20% of the constructs are used 80-90% of the time
- Structural modeling specifies a skeleton that can be refined and extended with additional structure and behavior
- Use case modeling specifies the functional requirements of system in an object-oriented manner
- Can be tightly coupled with program code

