Lecture 5: Introduction to Analysis

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2003

Goals for this Lecture

- Introduce the concept of analysis
  - Discuss requirements
  - Discuss requirements engineering
  - Discuss requirements analysis
  - Discuss requirements/design gap
  - Discuss the problem context of software engineering
  - Discuss domains

IEEE definition of requirement

1. A condition or capacity needed by a user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents
3. A documented representation of a condition or capability as in 1 or 2

Requirements Engineering

"The systematic process of developing requirements through an iterative cooperative process of analyzing the problem, documenting the resulting observations in a variety of representation formats, and checking the accuracy of the understanding gained."

- K. Pohl, 1993

Questions to consider

- Can one be systematic in the face of vaguely understood requirements?
- Can one know whether the requirements are complete in the context of iteration?
- How do you define cooperation among agents?
- What representation formalisms can be used?
- How can a genuine shared understanding be reached?

Two Sides to Requirements Engineering

- Requirements Elicitation
  - The process whereby a development agency discovers what is needed and why
  - Uses knowledge elicitation techniques
    - ethnomethodology, human factors, ergonomics, etc.
- Requirements Analysis
  - The process of understanding the requirements
  - Asks questions about completeness and consistency
  - Uses formal methods of systems analysis

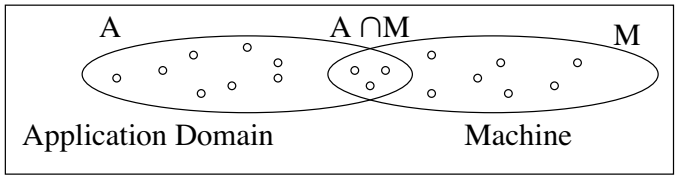
Requirements Analysis

- Understanding the phenomena of the application domain
- Describing the required relationships among the phenomena
- Example: Elevator Controller
  - Phenomena concern the application domain, not the (software) machine that controls it
    - buttons being pressed, buttons lighting up, cars moving in directions, doors opening and closing, people entering and leaving

Design

- Creating a machine that satisfies the requirements
  - Machine ensures satisfaction by sharing phenomena with application domain
    - shared events occur in both domains
    - shared states visible in both domains
- Example: Elevator Controller
  - “Press up button on floor 3” ≈ “Signal on line 3U”
  - “Car at floor 3” ≈ “Floor_Sensor_State[3] = 1”

Application versus Machine Phenomena



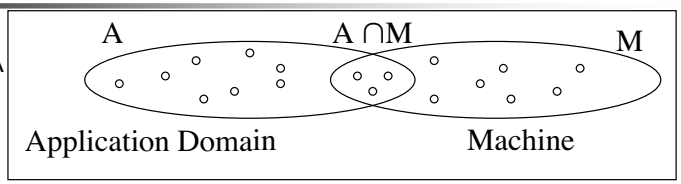
- Not all phenomena are shared
- Creates requirements/design gap
- Example: Elevator Controller
  - Car movement while between sensors
  - Correspondence of person pushing button to person exiting

Does System Satisfy Requirements?

R(equirements) ⇒ A

P(rogram) ⇒ M

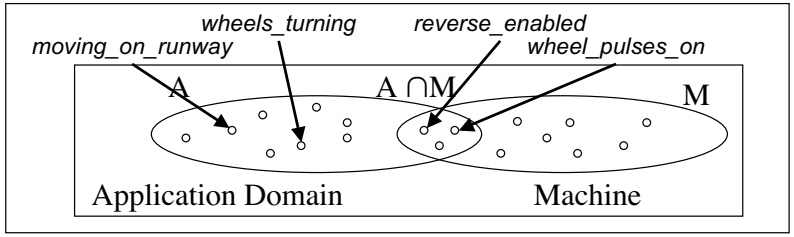
S(pec.) ⇒ A ∩ M



1. If computer behaves as P, then S satisfied
   - C,P ⇒ S, where C is the properties of the computer
2. If S satisfied, then R must be satisfied
   - D,S ⇒ R, where D is the properties of the application domain

Understanding Domain is Critical

- Example: Automated Thrust Reverser
  - Requirement
    - *reverse_enabled* IFF *moving_on_runway*
  - Domain Properties Assumed by Developers
    - *wheel_pulses_on* IFF *wheels_turning*
    - *wheels_turning* IFF *moving_on_runway*



Domain Misunderstandings ➞ Errors

- Example: Automated Thrust Reverser
  - Derived Interface Specification
    - *reverse_enabled* IFF *wheel_pulses_on*
  - Domain Properties Assumed by Developers
    - ✓ *wheel_pulses_on* IFF *wheels_turning*
    - ✗ *wheels_turning* IFF *moving_on_runway*
  - Aquaplaning Wheels
    - *moving_on_runway* is TRUE
    - *wheels_turning* is FALSE

Delving Deeper

- The requirements/design gap is a significant challenge to software development
  - Lets dig deeper and examine
    - The problem context of software development
    - Domains
      - And return to the concept of identity

Problem Context vs. Problem

- One step in making requirements easier is understanding the difference between the problem context and the problem

What's the problem?

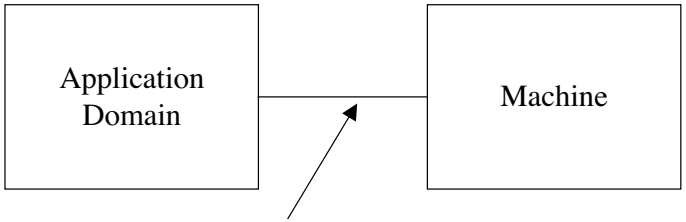
- There is a river. On one side of the river there is a farmer, with a fox, a rabbit, and a prize cabbage. There is a rowing boat, complete with oars, moored on that side of the river. On the other side of the river is a market. There is room in the boat for any two of the four: farmer, fox, rabbit, and cabbage. The fox is hungry, and so is the rabbit. Foxes like to eat rabbits and rabbits like to eat cabbages.

Problem Context vs. Problem

- There is typically one problem context, but there may be multiple possible problems
- In software development, our job is to build a machine. The problem context is
  - the part of the world in which the machine will be installed
  - the part of the world in which the effects and benefits of the installed machine will be felt and evaluated

Software Development Problem Context

We want to build a machine that will solve a particular problem; so to determine the problem we must understand the (application) domain



This line is important; it represents shared phenomena

Domains

- What is a domain?
  - "In building a typical large software system, the analyst generally has to deal with a number of distinctly different subject matters, or domains. Each domain can be thought of as a separate world inhabited by its own conceptual entities, or objects"
    - From "Object Lifecycles" by Shlaer and Mellor
      - both the application domain and the machine are domains

Subdomains: How Many?

- Patients in an intensive-care ward in a hospital are monitored by electronic analog devices attached to their bodies by sensors of various kinds.
- Through the sensors the devices measure the patients' vital factors: one device measures pulse rate, another temperature, another blood pressure, and so on. A program is needed to read the factors, at a frequency specified for each patient, and store them in a database.
- The factors read are to be compared with safe ranges specified for each patient, and readings that exceed the safe ranges are to be reported by alarm messages displayed on the screen of the nurse's station. An alarm message is also to be displayed if any analog device fails.

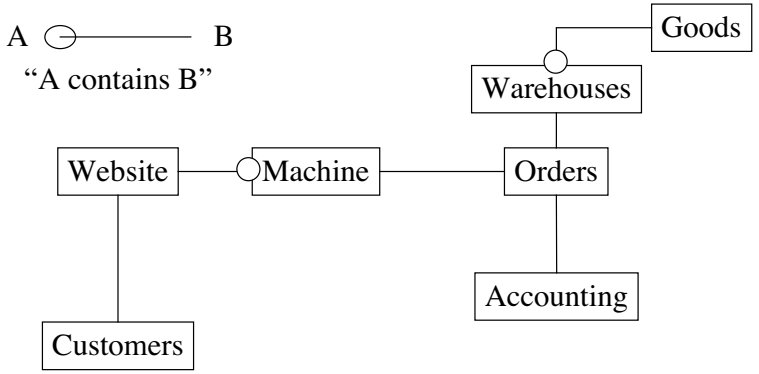
The Principle of Domain Relevance

- "Everything that's relevant to the requirements must appear in some part of the application domain"
  - If you identify a sub-domain that seems to have no relevance to the requirements, then you have picked a domain outside of the application domain
- Important consequence: the application domain is not limited to the parts of the world directly connected to the machine

The Context Diagram

- Useful in tracking the relationships between domains
  - identify those domains that are directly connected to the machine
    - draw them as nodes connected to the machine node
  - then attach the remaining domains

Example



Domain Interactions

- Domains interact via Shared Phenomena
  - Take the intensive care unit example
    1. A patient's temperature increases one degree
    2. A sensor detects this change and updates its internal state
    3. The machine detects this change and updates the database
    4. The machine may later notify the nurses station

More on Domain Interactions

- Shared Phenomena are important but internal properties of a domain are important too
  - a device updating its register based on a change reported by a sensor is an internal property
- So, how do we determine if our breakdown of an application domain is good? (whether each domain can be considered separately?)
  - the answer is that the internal properties and behaviors of each domain must be largely independent and only interact minimally via shared phenomena
  - similar to the software engineering terms of coupling and cohesion; want high cohesion and low coupling of domains

Why are domains important?

- Domains contain Phenomena
  - And shared phenomena between the application domain and the machine can lead to requirements
- Phenomena are often modeled as
  - entities and relations
  - events involving processes

Entities and Relations

- Managing Courses at a College
- Entities
  - Courses, Subjects, Lecturers, Students
- Relations
  - Attends, Covered-By, Taught-By, etc.
- Is this enough to model a domain, to capture all relevant phenomena?

Events involving Processes

- Automatic Turnstile (say for a Subway)
- Events
  - InsertTicket, UpdateTicket, ReturnTicket, Lock, Unlock, Enter
- Processes
  - Turnstile, Customer
- Is this enough to model a domain?

Clearly Not!

- In an entity-relation view, you are still going to need events and processes
  - and vice versa
- This is the result of the fact that the “real world” is too complex and varied to be modeled by a single phenomenology
  - You need multiple ones to do it right
  - However, there is a “lowest common denominator” which can serve as a useful starting point

Facts about Individuals

- A fact is a simple truth about the world
  - 23 is prime; 6 is between 4 and 9;
  - Ann is a manager
- A fact is, thus, the smallest unit of observation about a domain, the smallest phenomenon
- Larger and more complex observations can be broken down into facts
  - "All employees are people" is not a fact; it's a complex assertion about many facts

Facts and Propositions

- It is important to distinguish facts from propositions
  - facts are phenomena in the world
  - propositions are statements of what may be facts
- "Ann is a manager" is a proposition
  - If Ann is a manager, this statement is true (and is a fact)

Facts Involve Individuals

- Dr. Anderson wrote this lecture
  - is a fact about two individuals
- In a domain, an individual can be anything at all…
  - a person, a number, an event, a date
- …as long as you can distinguish one from the other, that is, they must have distinct identities

Choosing Individuals

- Picking the individuals of a domain is key to shaping how you view the world and what you can accomplish
- But it is often not easy!
  - "Nine teachers and 23 alumni are coming to a school meeting. Each wants a cup of coffee. How many cups do you need?"
  - Two mechanics start with two cars of the same make and model; they begin to swap parts of the car (first wheels, then doors, etc.) At what point have they swapped cars?

More examples

- An airline flight may be regarded as an individual. But two flights may be merged into one journey made by one airplane. Or one flight may have intermediate stops between its starting and ending airports, with different planes used for different sections of the same flight
- A phone call may be regarded as individual. But suppose A calls B, B establishes a conference call with C, B then drops out leaving A talking to C. How many calls is that? How many calls are on a "chat line" where an ever changing population of people dial into an unending conversation?

What's the problem? Identity...

- If you are forced to take a view of the world
  - where you cannot reliably distinguish one flight from another, or one call from another, or one car from another
    - then cars, flights, and calls cannot be individuals
- To say that X is identical to Y is to say that they are one and the same individual
- A related notion is "similarity"; often two individuals are similar because they share a common trait
  - B's birthday is the same as C's birthday
  - B and C are similar; their birthdays are identical

Summing Up

- In software development,
  - we have a problem context that includes an application domain and a machine
  - since the machine is a solution, we need to understand the application domain to understand the problem we are facing
  - a domain consists of phenomena, which we can model (e.g. understand) as facts concerning individuals
  - if we can pick the right individuals within a domain, we can identify relevant facts or phenomena about the domain
  - In particular, we increase our chance of finding shared phenomena between the application domain and the machine; this shared phenomena can serve as a specification for a program

What's Next?

- Descriptions
  - designations and definitions
    - a structured approach to requirements analysis
  - refutable and rough sketch
    - states a description can be in
- Events and Intervals
  - representing time in our descriptions
- We will then explore the tutorial that appears in Section 2.2 of your textbook and then look at object-oriented analysis in more detail