Introduction to Software Engineering

Requirement engineering – part II

Philippe Lalanda

Philippe.lalanda@imag.fr

http://membres-liglab.imag.fr/lalanda/

Requirement engineering - reminder

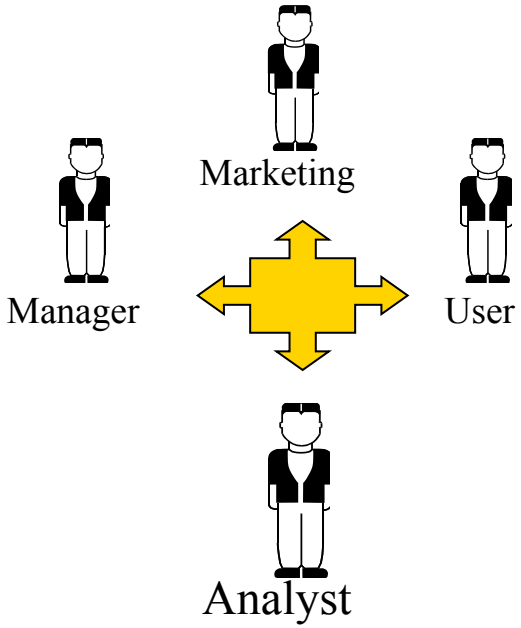
Analysis	Identifying new requirements and stakeholder conflicts (can include a feasibility study)
Definition and negotiation	Checking requirements and resolving stakeholder conflicts
Specification	Documenting the requirements in a requirements document (SRS)
Validation	Checking that the documented requirements and models are consistent and meet stakeholder needs
Management	Managing changes to the requirements as the system is developed and put into use

Outline

- Analysis phase
- Analysis techniques
- Analysis and prototypes
- Specification in natural language
- Specification with analysis models
- Conclusion

Analysis

- Identifying the requirements and the conflicts is a complex activity
- A hard-to-find combination of skills is necessary
  - Communication
  - Politics
  - Synthesis
  - Quick understanding
- Fortunately, analysis techniques have been proposed



Analysis issues

- Using multiple sources of information
  - How to identify and use them?
  - How to communicate with stakeholders?
  - How to identify and solve conflicts?
- Dealing with a great deal of information
  - How to store and structure the information?
  - How to present the information?
- Getting all the requirements
- Staying away from design

Multiple sources of information

- All available sources have to be exploited
  - Stakeholders
  - Existing systems (legacy)
    - Documentation and code!
  - Standards
  - Regulation authority, certification authority, …
  - Domain decriptions

Multiple sources - example

- Library example
  - Library employees (direct users)
  - Clients (indirect users)
  - Library director (indirect user)
  - System administrators
  - Book editors
  - Regulation (law about book borrowing)

Multiple stakeholders – various interests

Uses

Pays for

Develops

User
- Functions
- usability
- response time
- reliability

Client
- Development time
- costs
- Functions
- maintainability
- evolvability
- reliability

Development contractor
- profits
- reuse of legacy
- capitalization
- maintainability : depends on the contract.
- recurring product or one shot?

Multiple stakeholders - politics

- Politics play a major role
  - Many decisions about requirements are taken to satisfy personal goals
  - Some people may be against a given development and adopt aggressive manners or be passive
  - These are human attitudes (protective, most of the time)
- Conflict management
  - Personal goals or oppositions are hard to perceive
  - They are harder to solve !

Some fun

- Study of 100 projects (Index Group Cambridge, 90)

- Conclusion
  - « Systems developers are operating amid turf battles, historical bickering, low credibility and the difficulty in pinning down ever-changing systems requirement »

Communication issues

- Getting users/clients needs and goals is difficult
  - At times, they are not aware of what they can get
  - At times, there is confusion between their needs and what they have with existing systems
  - At times, there is a gap between what they want and what they need
  - At times, they do not want to work hard
    - « Show me what you can do … »
    - « You are the specialists »

Not working hard enough

Communication issues - example

- Users ask the analyst to change an algorithm which does not work correctly on the existing system
  - Analyst: how often is it used?
  - Users (together): never!
  - So the analyst ignores the demand

- Of course, the reason why the algorithm is never used is because it … does not work !
  - Be curious
  - Be careful not to ask biased questions
  - Be careful not to decide for others

Communication issues

- Use of slang
  - Domain specialists use their own language (with hard to get subtleties)
- Undesired omission of information
  - Domain specialist do not tell “obvious” things
- Desired omission of information
  - Requirements related to the organization for instance

Different types of knowledge

- Tacit knowledge
  - Knowledge we are not aware of
- Semi-tacit knowledge
  - Knowledge we are vaguely aware of (we can express it if we are questioned)
- Non tacit knowledge
  - Knowledge we can express directly

Big quantity of information

- Analysis techniques are needed to
  - Structure information
  - View information
  - Improve systematic search (of information)
  - Verify coherence, completeness, …
- Related to the way we specify requirements
  - Sentences, models, interaction diagrams, …

Outline

- Analysis phase
- Analysis techniques
- Analysis and prototypes
- Specification in natural language
- Specification with analysis models
- Conclusion

Data mining

- Use of existing systems to infer requirements
  - Requires access to documents, code, …
- Advantages
  - Autonomy
  - Reuse (of previous analysis)
- Limits
  - Reuse of good and bad stuff
  - Constraints cannot be identified
  - Clean re-design is not possible

Ethnography

- Immersion in the user work environment
  - Requires access to working environment
- Advantages
  - Discovery of functional requirements and constraints
  - Successful use in air traffic, subway control, office work (big difference between presumed and real work, Suchman 87)
- Limits
  - Collection of lots of useless information
  - Hard to find valuable information
  - Domain/organizational requirements are difficult to get
  - Actions are not always understood

Protocol analysis

- A user performs a task, constantly explaining what he is doing
  - Requires identification of meaningful tasks
- Advantages
  - Discovery of functional requirements and constraints
  - Working activities (actions) are understood
- Limits
  - Collection of lots of useless information
  - Long and meticulous approach
  - Domain/organizational requirements are difficult to get

Brainstorming

- Group people to generate as many ideas as possible
  - Creativity, no evaluation
  - Requires a well balanced group
- Advantages
  - Tackle aspects that would not be treated with a rational approach
- Limits
  - Depends heavily on the group
  - Unstructured, uncontrolled, non systematic
  - Can be completely out of scope and useless
  - Conflicts not revealed

Non structured interviews

- Interview stakeholders without prepared framework
- Advantages
  - No preparation (or very light)
  - Important aspects are directly identified with stakeholders
- Limits
  - Interviews are hard to conduct (skills and experience needed)
  - A lot of time can be spent on minor questions
  - Information can be hard to analyze / structure
  - Conflicts are generally not revealed

Structured interviews

- Interview stakeholders with predefined questions
  - Requires preparation and rapid analysis of problem
- Advantages
  - Interviews are easier to conduct (still hard!)
  - Systematic (same questions for everyone)
  - Interviews are under control
- Limits
  - The predefined framework may omit important points
  - Conflicts are generally not revealed

RAD workshop

- A 8 to 20 persons workshop
  - With a facilitator and creation of working groups
  - Workshop has to be prepared (heavy task)
- Advantages
  - Bring together major actors
  - Conflicts are revealed
  - Quality and efficiency
    - Joint Application Design : 10% of missed requirements against 30% in général (Caper Jones, 1989)
- Limits
  - Depends heavily on the facilitator and on the group
  - Important persons must be brought together a few days
  - Appropriate for major projects, not too big

Use cases

- Analysis can be partly oriented towards the elicitation of use cases.
  - Focus on interactions
  - Built with stakeholders, but mainly users
- Advantages
  - Facilitates discussion (very concrete)
  - Strong user involvement
- Limits
  - Domain/organization requirements are not revealed
  - Conflicts and constraints are not revealed

Used techniques

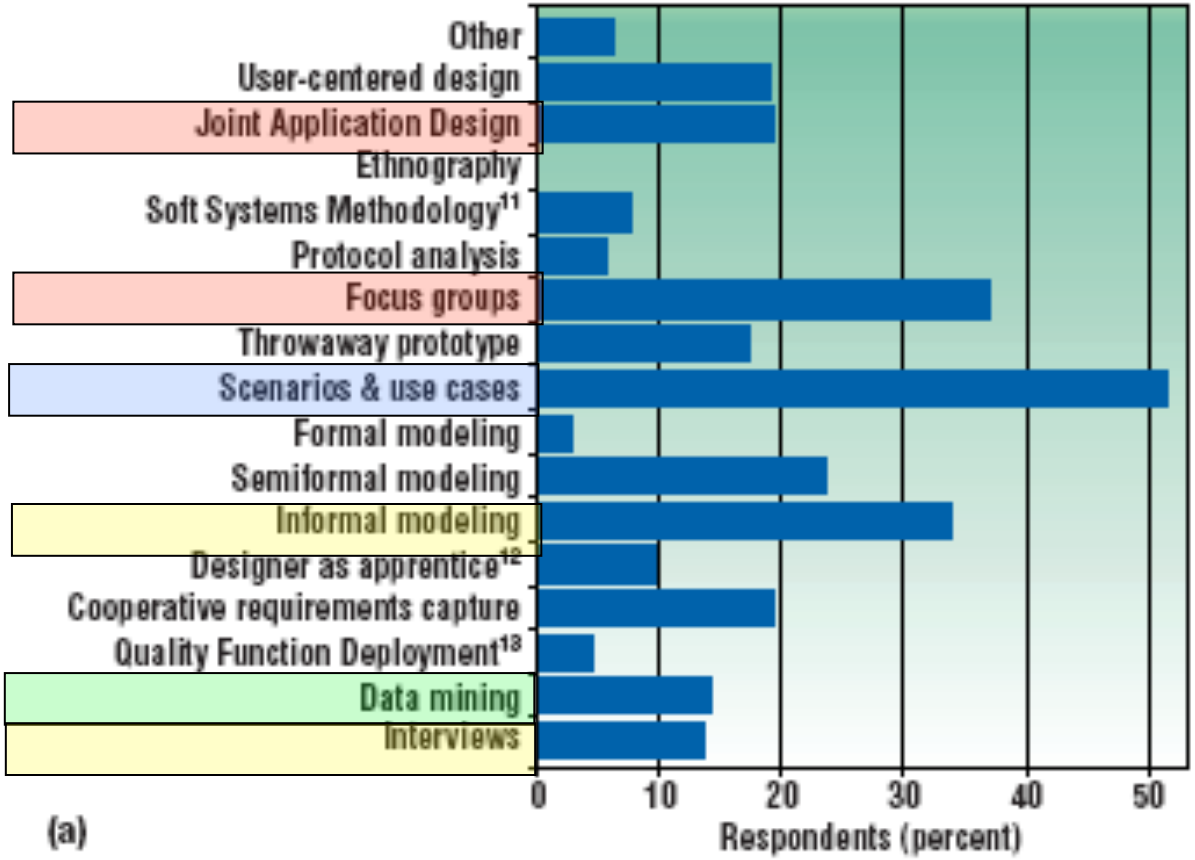
- Requirements engineering: The state of the practice
  Neill et Laplante, IEEE Software, 2003

workshops

Use cases

Informal

Mining



Comparison: Function and data acquisition

	Data mining	Observation	Protocol analysis	Non str. Interviews	Interview	Brainstorming	RAD
Functions	+	++	++	+	+	+	++
Data	+	-	-	+	+	+	++

From Maiden and Rugg, 96

Comparison: knowledge acquisition

	Data mining	Observation	Protocol analysis	Ns Interview	Interview	Brainstorming	RAD
Non tacit	-	-	++	++	++	++	++
Tacit	++	++	-	-	-	-	-

From Maiden and Rugg, 96

Comparison: preparation and acquisition time

	Data mining	Observation	Protocol analysis	Ns Interview	Interview	Brainstorming	RAD
Preparation time	-	-	++	+	++	-	++
Time needed to get req.	++	+	-	-	-	++	-

From Maiden and Rugg, 96

Comparison: actors interest

	Data mining	Observation	Protocol analysis	Ns Interview	Interview	Brainstorming	RAD
Actors interests	++	-	-	++	+	++	+

From Maiden and Rugg, 96

Outline

- Analysis phase
- Analysis techniques
- Analysis and prototypes
- Specification in natural language
- Specification with analysis models
- Conclusion

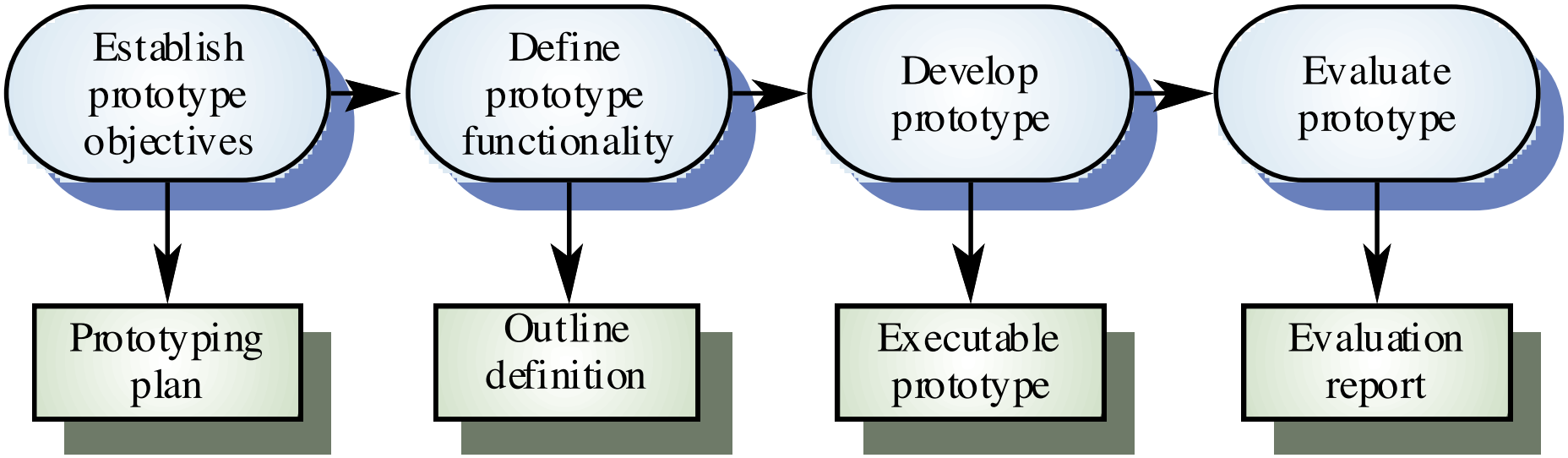
Prototyping

- The purpose is to build a mockup of the system to be developed and present it to stakeholders
  - help customers and developers understand the requirements for the system
- Useful for software with
  - many interactions
  - dynamic interfaces
  - unclear algorithms
  - unclear expected functions

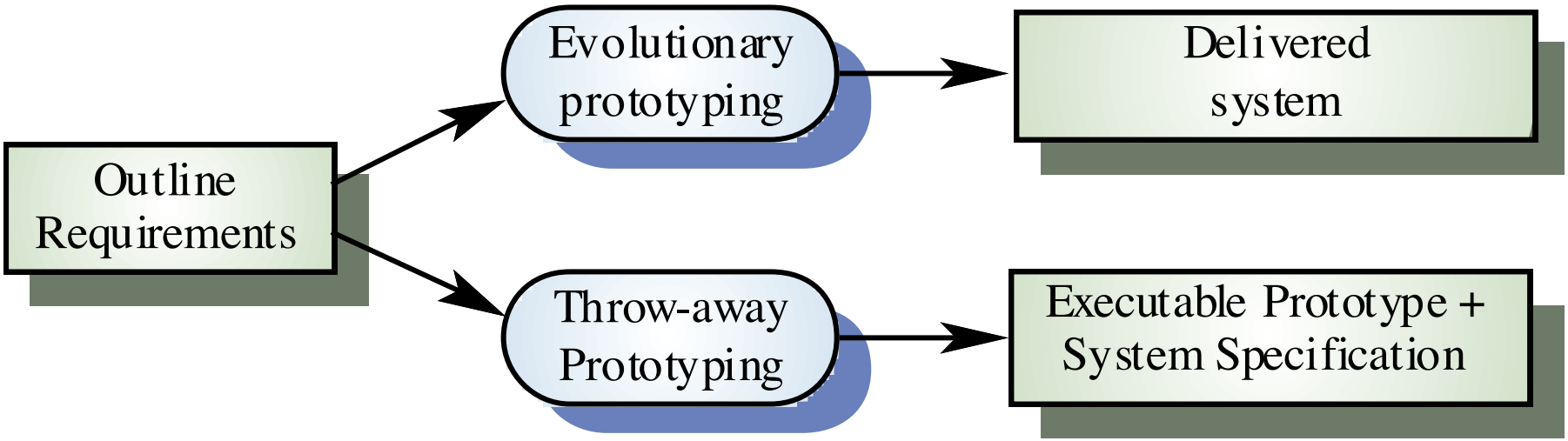
Rapid prototyping - continued

- Advantages
  - Facilitates discussion (very concrete)
  - Misunderstandings between software users and developers are exposed
  - Missing services may be detected
  - Confusing services may be identified
- Limits
  - Cost may be badly understood by managers
  - Temptation to build on top of the prototype
  - Functions to be prototyped must be identified anyhow!

Prototyping process – from Sommerville



Prototyping approaches – from Sommerville



Throw away prototyping – from Sommerville

- Use to reduce requirements risk
  - The prototype is developed from an initial specification, delivered for experiment then discarded
- The throw-away prototype should NOT be seen as a final system
  - Some system characteristics may have been left out - shortcuts
  - There is no specification for long-term maintenance
  - The system will be poorly structured and difficult to maintain

Evolutionary prototype – from Sommerville

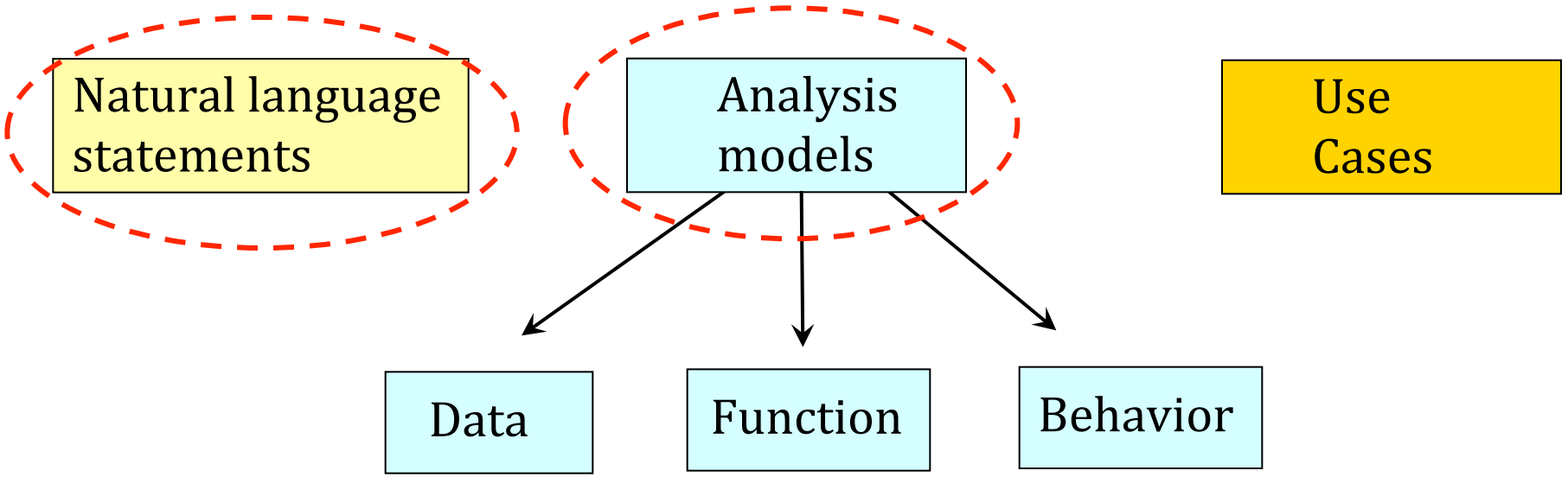
- Must be used for systems where the specification cannot be developed in advance
  - Based on techniques which allow rapid system iterations
  - Verification is impossible as there is no specification. Validation means demonstrating system adequacy
- Issues
  - Continual change tends to corrupt system structure
  - Specialist skills are required
  - Organizations must accept that the lifetime of systems developed this way will inevitably be short

Outline

- Analysis phase
- Analysis techniques
- Analysis and prototypes
- Specification in natural language
- Specification with analysis models
- Conclusion

How to write specifications?

- Different notations are used to write down the SRS
  - Different types of requirements
  - The sum of them form the requirements document



Natural language statements

- Natural language is an easy, natural way to express requirements
- But possibly
  - Ambiguous
  - Imprecise
  - Inconsistent
- Also, there is a gap with the design/coding phases

Examples

- The system must provide appropriate viewers to read documents
  - Imprecise -> which ones? what is appropriate?
- A subscriber cannot borrow more than 5 books
  - Ambiguous -> at a time?
- A book can be available or not. A book turns unavailable when it is spoiled.
  - Incomplete -> available again?

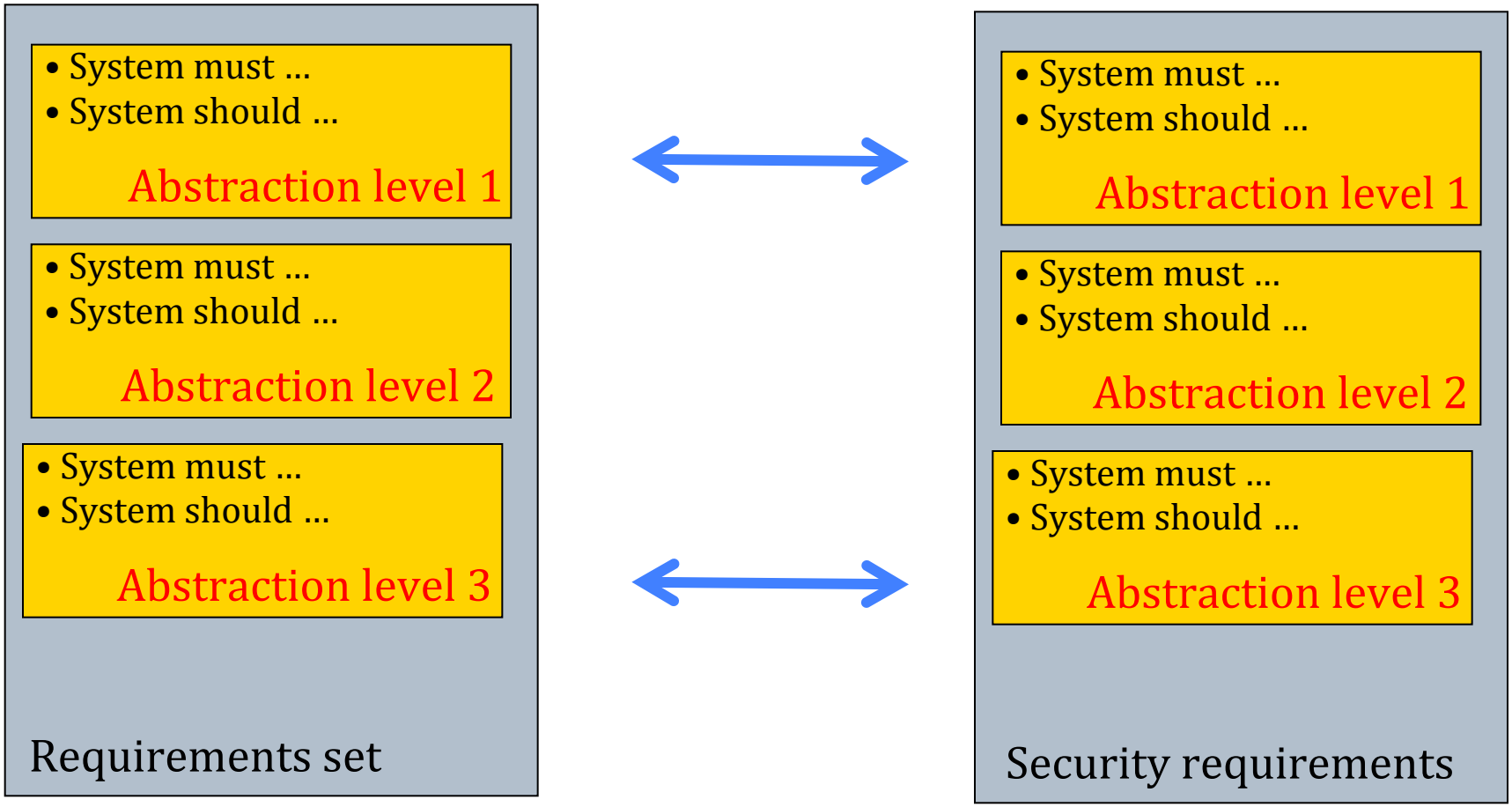
Natural language - conventions

- Use of conventions to improve consistency
  - Consistent use of important words
    - must, should, may …
  - Use bold to indicate important parts
  - Avoid computing jargon

Natural language - structuration

- Grouping requirements
  - Data
  - Functions
  - qualities
  - Constraints
  - Business-related aspects
- Abstraction
- Traceability

Natural language - structuration



Structured languages

- Requirements are expressed in a common format
  - A form specifies a format that has to be used for every requirement
  - A form is made of fields
  - Fields can be typed
  - Some fields may be optional
  - Form types can be defined for requirements types

Structured languages - examples

- **Function** Calculate-rights
- **Description** This requirement shows how authors rights must be calculated
- **Input** A paper with a list of author
- **Output** Due amount of money
- **Action** Due money is null is the paper appeared before 1900.
  If the paper is posterior to 1900, the amount of due money is the number of pages multiplied by 1 cent.
  (note: this is grossly invented!)
- **Pre-cond.** The user can pay
- **Post-cond.** Add the due money to the user account
- **Other** None

Structured languages

- Advantages
  - It is natural language …
  - … with an improvement in structure and uniformity
- Limits
  - Same flaws as natural language (ambiguity, …)
  - Still general
    - not always detailed enough for designers/programmers

Outline

- Analysis phase
- Analysis techniques
- Analysis and prototypes
- Specification in natural language
- Specification with analysis models
- Conclusion

Analysis model

- The purpose of an analysis model is to express requirements in a semi formal way
- Combination of text and diagrams to represent software requirements
  - Data, function, behaviour
- First « technical » representation of a system
  - May be the foundation of the design model since it describes the logical structure of the system

Analysis model

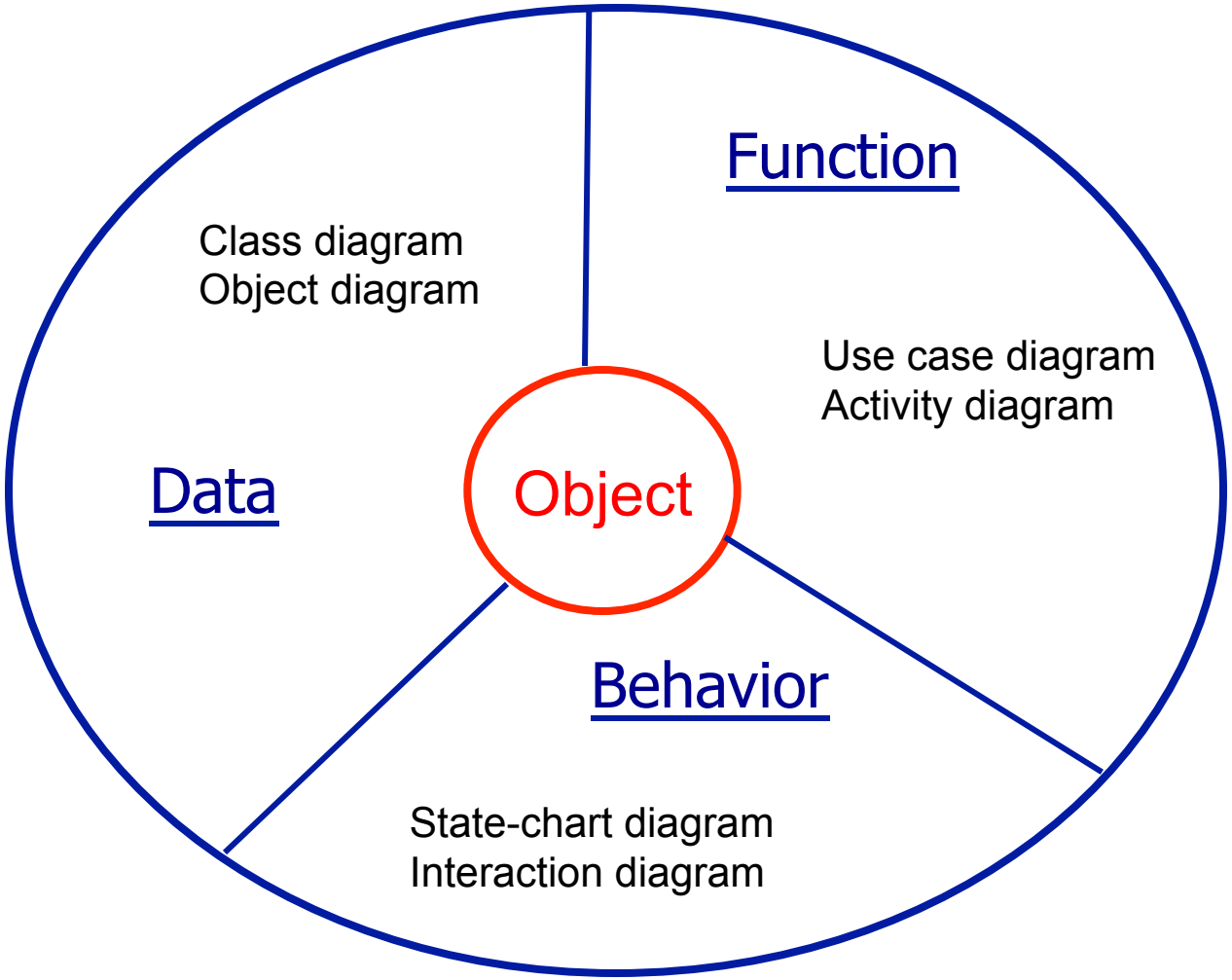
- Two types of analysis modelling
  - structured analysis
  - object-oriented analysis

- Same goal, different focus

Analysis models

- Data model
  - ER model
  - Object model
- Functional model
  - Data flow model
- Behavioral model
  - State transition model
  - System behavior modeling with sequence diagram

Object analysis models



Object oriented analysis models

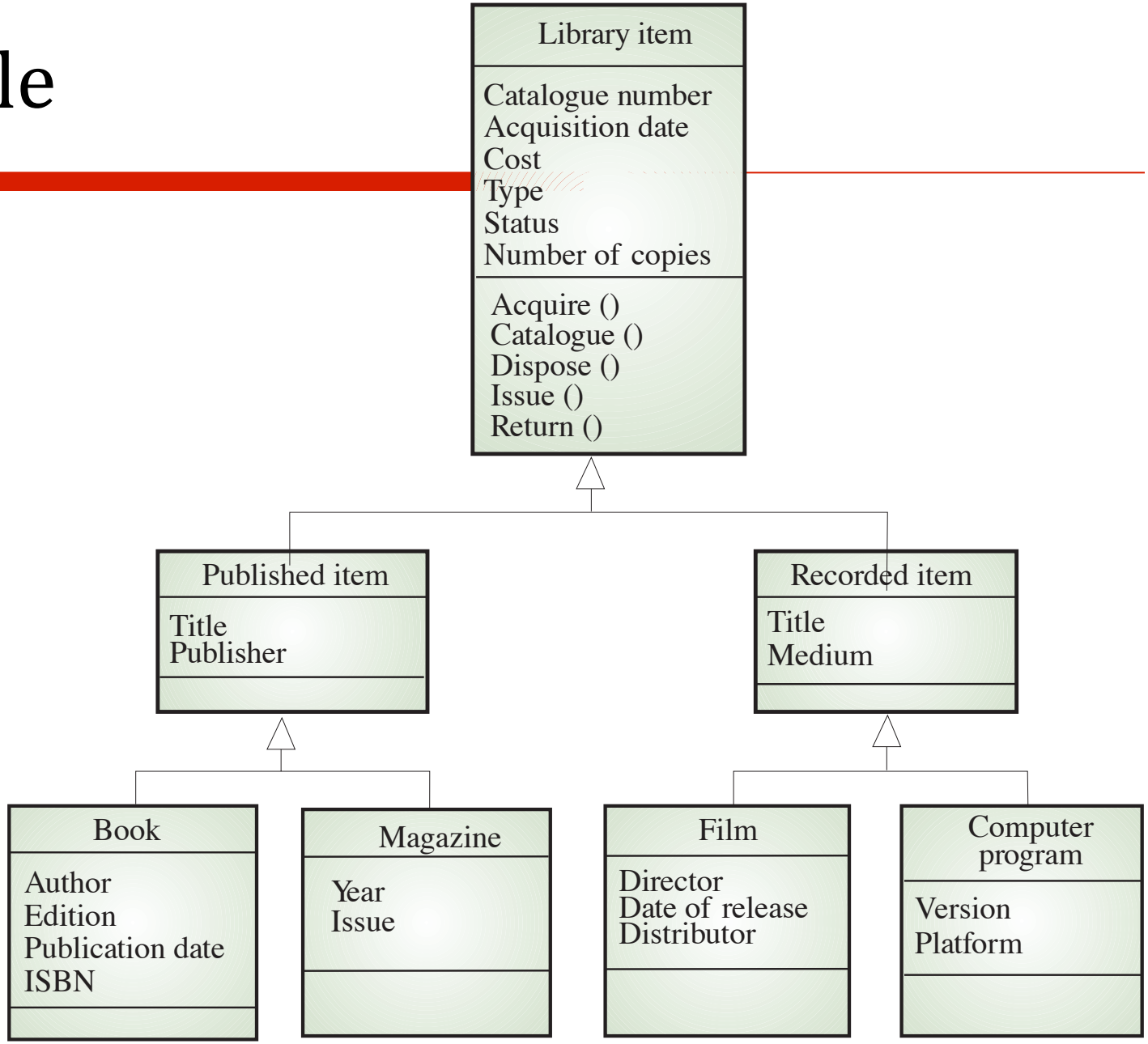
- Object = entity + operations
- Models describe the system in terms of classes
- An object class is an abstraction over a set of objects with common attributes and services

- Various object models may be produced
  - Class models (with inheritance, aggregation, ...)
  - Interaction models (use cases)
  - Dynamics models (sequence diagram ...)

OO analysis – class diagram

- Identify objects and relationships
  - Identify classes (no implementation details)
  - Eliminate irrelevant classes (redundancies, lack of interest, vague, …)
  - Identify relations
  - Eliminate irrelevant relations (redundancies, irrelevant classes, implementation level, …)
  - Identify attributes …

Example

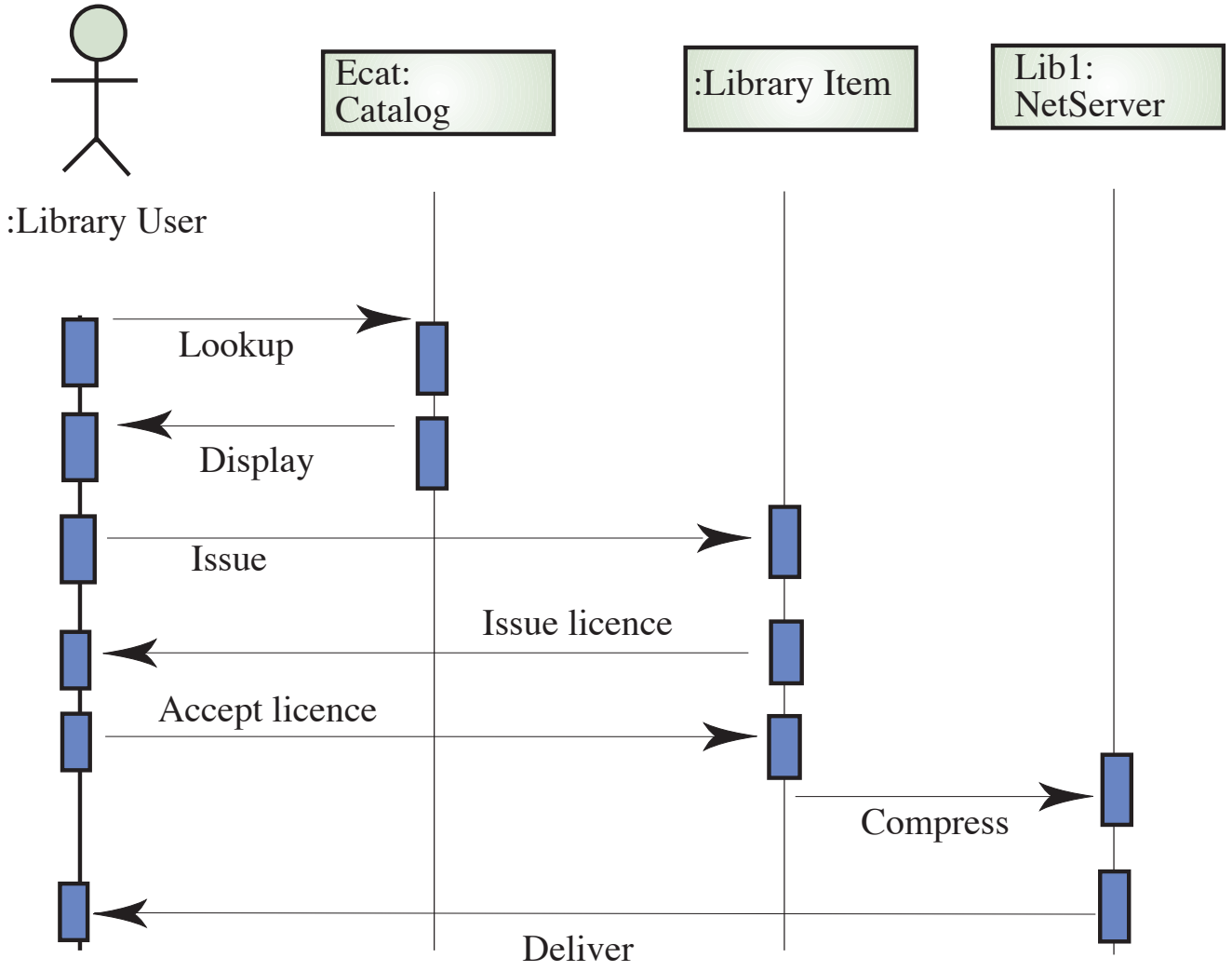

Library item
Catalogue number
Acquisition date
Cost
Type
Status
Number of copies
Acquire ()
Catalogue ()
Dispose ()
Issue ()
Return ()
Published item
Title
Publisher
Recorded item
Title
Medium
Book
Author
Edition
Publication date
ISBN
Magazine
Year
Issue
Film
Director
Date of release
Distributor
Computer program
Version
Platform

OO analysis – behavioural diagrams

- A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case

- Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects

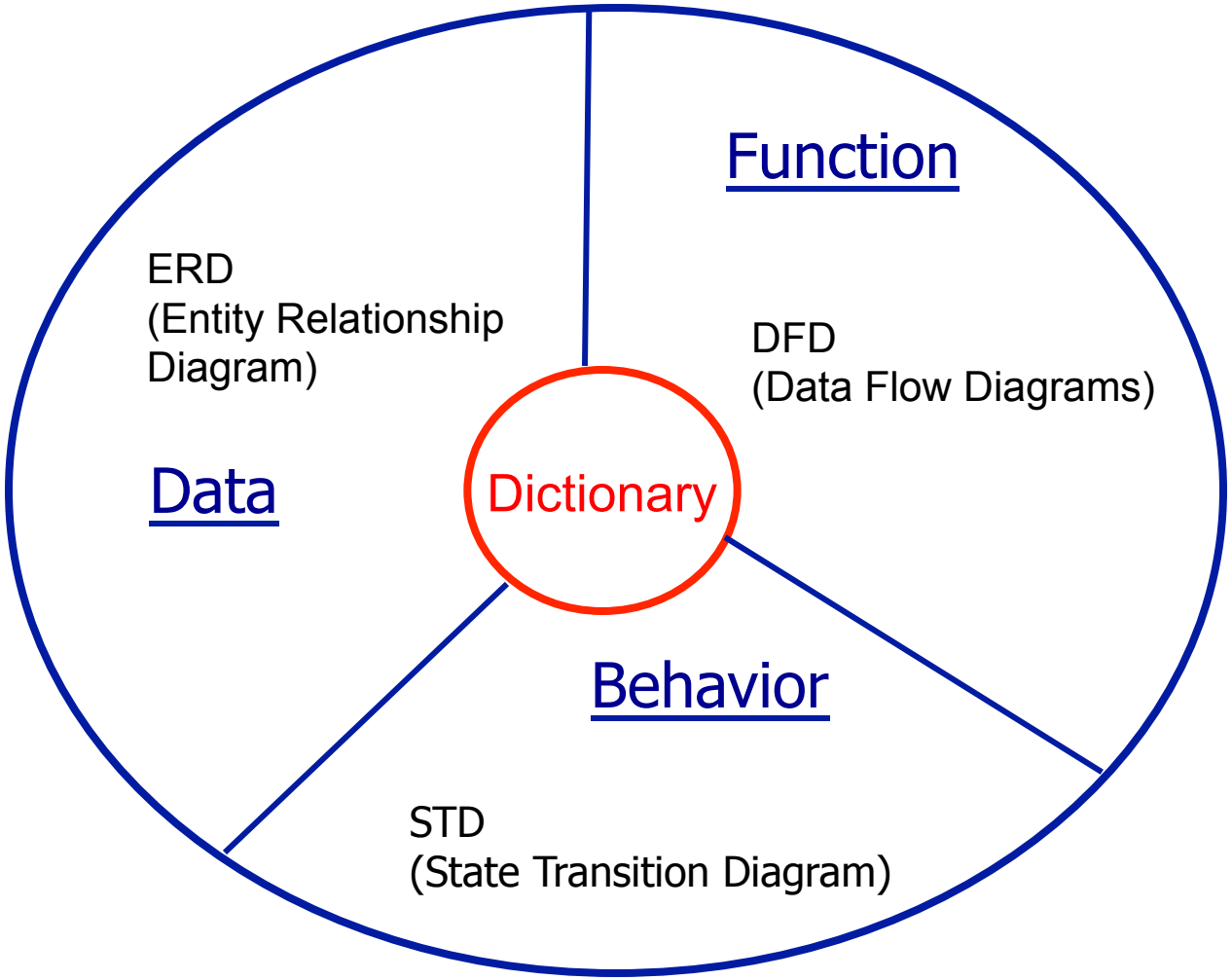
Example of sequence diagram



OO analysis – conclusion

- Salient points
  - Careful not to be naïve
    - Domain entities do not appear like magic!
    - Requirement elicitation not a domain study
  - Functions are not easily discovered
  - Passing to the design phase is not seamless
  - Very popular today

Structured Analysis models



Entity-relationships diagram

- Examine data objects independently of processing
- Focus attention on the data domain
- Create a model at the customer level of abstraction

Book
Title Author ISBN

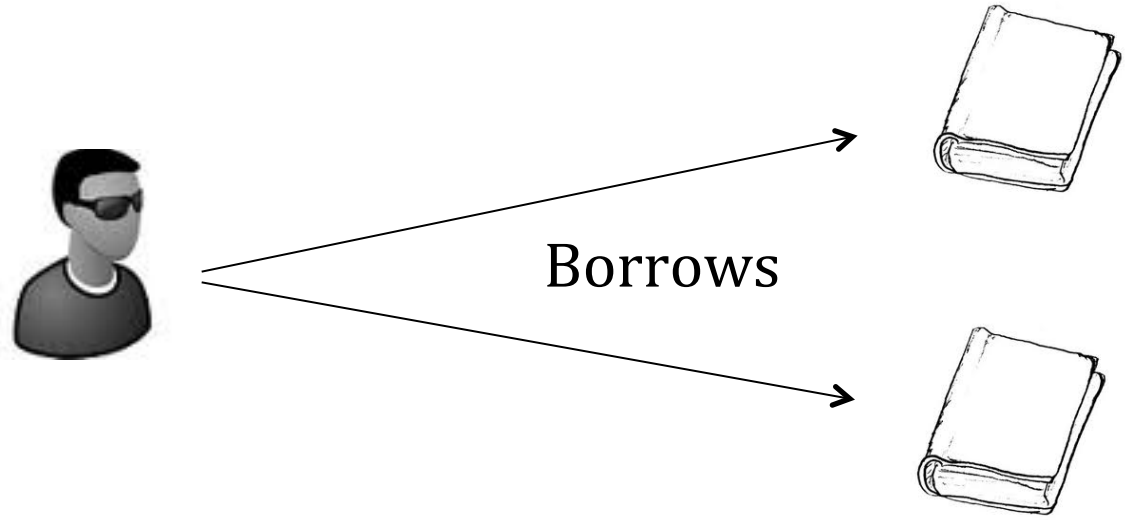
Person
Name Birthday Address

Entity-relationships diagram

- A fact that must be remembered by the system and cannot or is not computed or derived
  - Several instance of a relationship can exist
  - Entity can be related in many ways

Cardinality

- A same object can have a multiple relationships

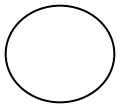


Data Flow Diagrams (DFD)

- Relationships between functions and data
  - How data flow into the system
- Data are
  - Within repositories (data sources)
  - Between functions (flow)
  - From outside
- Expressed in simple, graphical languages
  - Major explanation of their success

Functional analysis - DFDs

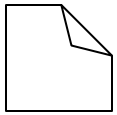
- Base elements
  - Functions: circles
  - Data flows: arrows
  - Data: two horizontal lines



Function

Input

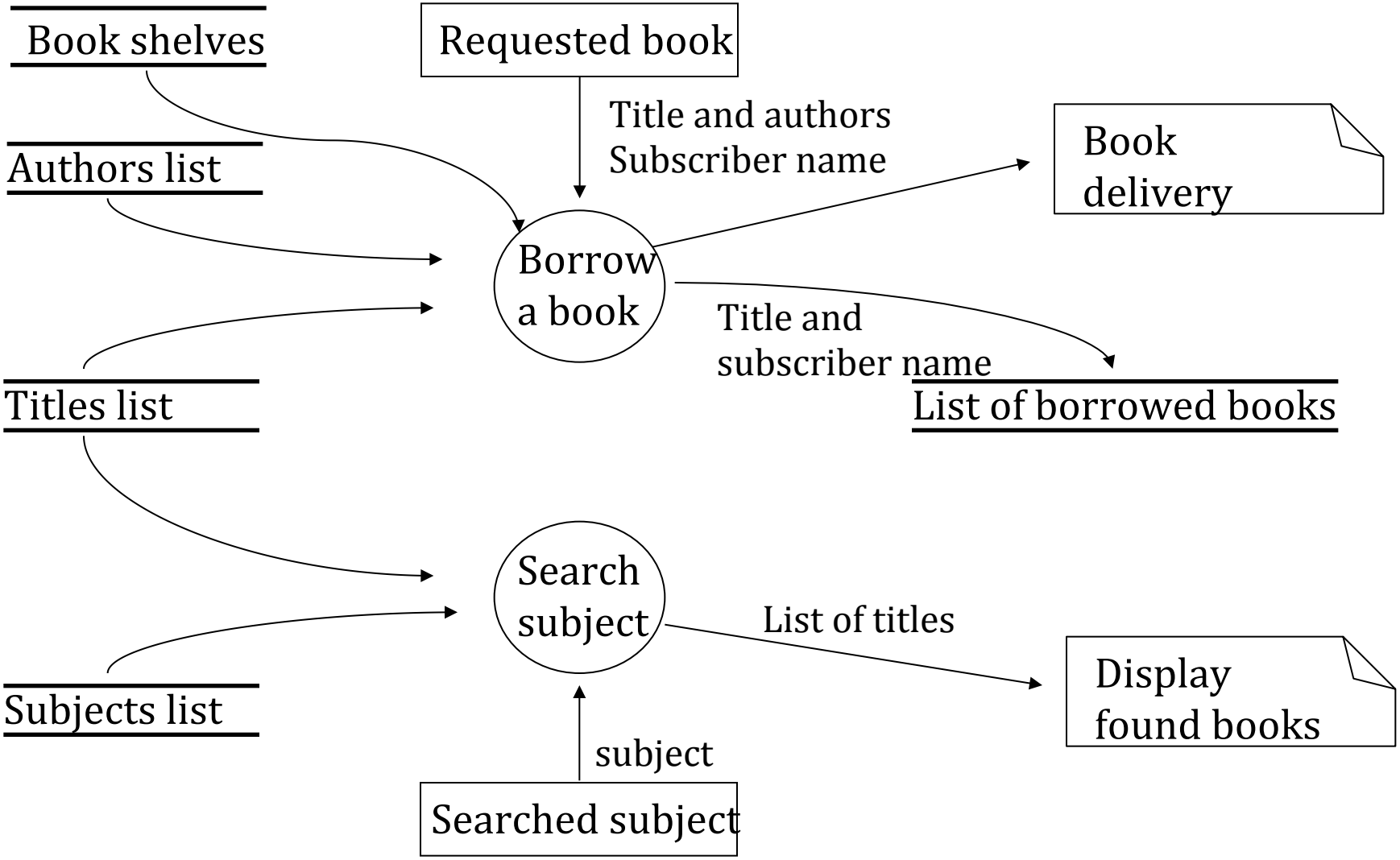
Data flow



Output

Data source

Example – from Ghezi, 2003



DFDs: semantics

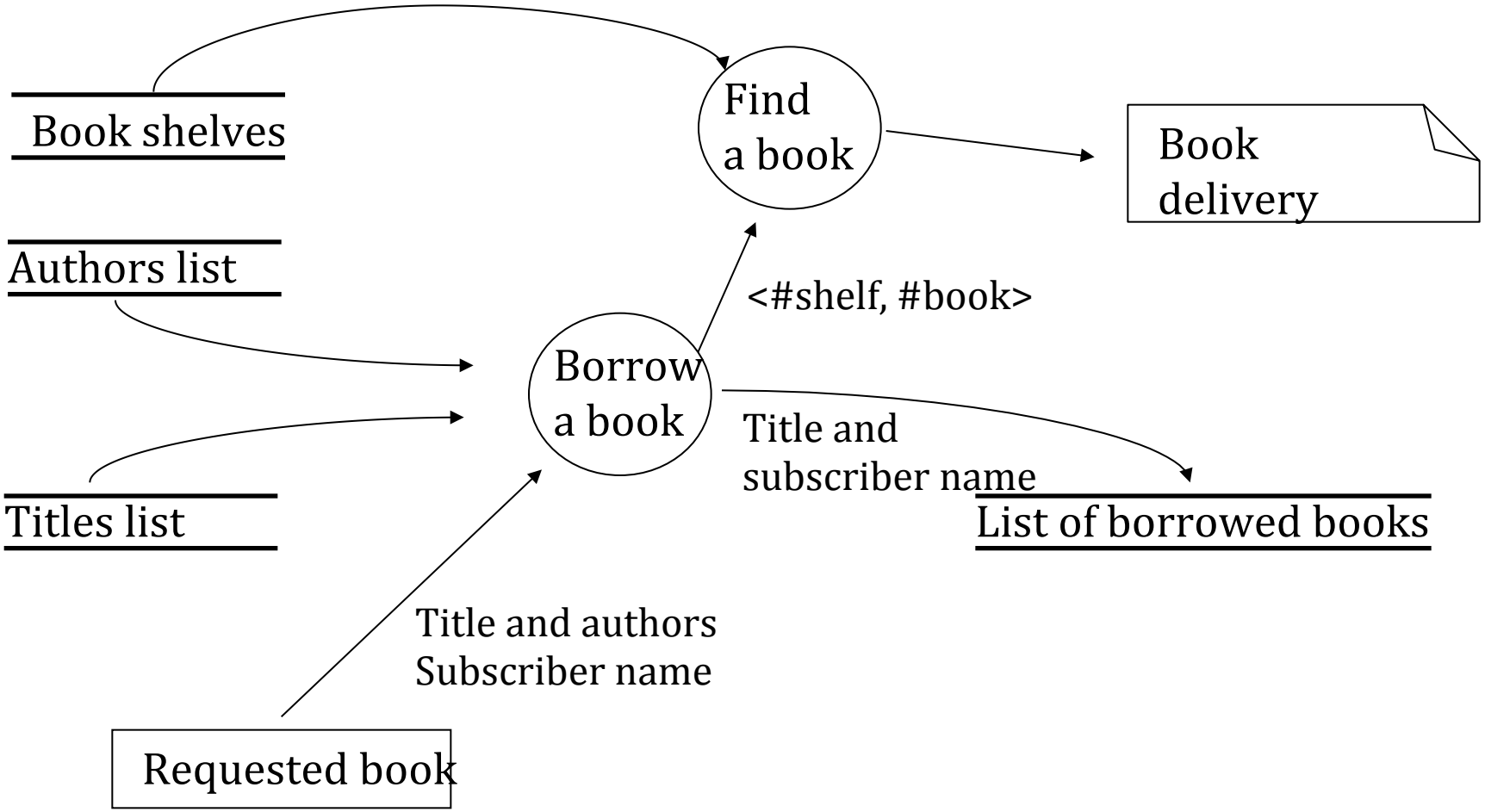
- Functions and data names are unique
  - Allows clear identification
- No event ordering
  - No notion of data flows order
- No decision points
  - No alternative paths
- Low level information is differed

DFDs: structuration

- Abstraction
  - Hierarchies of diagrams
  - The more abstract diagram = context diagram
  - Refinement with sub-diagrams
    - To specify a function
    - To precise data flows

- Separation of concerns
  - Definition of focalized diagrams

Example of refinement – from Ghezi, 2003



DFDs: advantages and limits

- Advantages
  - Simple, understandable, good for communication
  - Provide a global vision
  - Decomposition, abstraction, focalization
- Limits
  - Lack of semantics
    - No possible abstract machine
  - Limited power of expression
    - No decision point
  - Hard to manage a big number of diagrams
    - Coherence management

Data dictionaries

- Complement DFDs
  - Names and aliases
  - Descriptions
  - Structure
  - Relationships
  - Possible values
  - Links with data flows

Dictionary example

Name	Generated by	Used by	Description	Structure
Book	Record book	- Borrow book - Reserve book -- Find item	Every work that can be consulted at the library	- Title - Author - ISBN - Date
Subscriber	Create Subscriber	- Borrow book - Reserve book -- Find item - Take book	Every person registered at the library.	- Name - First name - age - address

Outline

- Analysis phase
- Analysis techniques
- Prototypes
- Requirement specification with natural language
- Requirement specification with models
- Conclusion

Analysis models

- Models complement natural language
  - More expressive (if simple modeling language !)
  - Materialize discussions
  - Prepare design
  - Developed together with LN requirements
    - Mutual enrichment
    - Discovery of inconsistence, incompleteness, …

Structure of a requirements document

- Context
  - Positioning, documents to be referenced, …
- Needs
  - Context
  - Use cases
- Functional requirements
  - Natural language
  - Analysis models (functional, object, …)
- Non functional requirements
- HMI requirements
- Implementation constraints

Conclusion


What we say to dogs
Okay, Ginger! I've had it! You stay out of the garbage! Understand, Ginger? Stay out of the garbage, or else!
What they hear
blah blah GINGER blah blah blah blah blah blah GINGER blah blah blah blah blah...