A Requirements Engineering Approach for Object-Oriented Conceptual Modeling

PhD Thesis

by Emilio Insfrán Pelozo

Supervisor: Óscar Pastor López
Department of Information Systems and Computation
Valencia Polytechnic University
October 2003
Contents

- Introduction
  - Motivation
  - Problem statement
  - Objectives

- Approaches for Requirements Engineering
  - Goal-based methods
  - Scenario-based methods
  - Foundations
  - Requirements Model
  - Requirements Analysis Process
  - Traceability Rules Catalog
  - A Requirements Engineering Environment

- Work Development

- Conclusions
  - Contributions
  - Publications
  - Current and Future Research
Motivation

“The hardest single part of building a software system is deciding precisely what to build... therefore, the most important function that the software builder performs for the client is the iterative extraction and refinement of the requirements”


If you don't understand the user's requirements, it doesn't matter how you code it.

Ed Yourdon
Motivation

- A survey on 100 SW developers for the car industry in the EU shows that 90% of participants use Natural Language text edited in MS-Word  
  (Alen Dutoit and Barbara Peach, May 2000)

- A State-of-Practice Survey, Telecom Business Research Center (Finland, 2001)
Motivation

- Why is it so difficult to introduce RE research results into mainstreams RE practice?  (Panel at CAiSE ’00 Conference)

  - Sjaak Brinkkemper (Baan Company R&D)
    - better composition and communication of reqs; tracking and tracing from reqs to design and code
  
  - Janis Bubenko (Royal Institute of Technology Stockholm)
    - no agreement regarding what reqs, or approaches for reqs specifications really stand for; we cannot tell what the benefit is of applying research results to practical cases
  
  - Barbara Farbey (University College London)
    - RE research will only be introduced into practice if a) they are shown to be aligned with the goals of the business and b) the introduction of new thinking should be seen as an organizational innovation
  
  - Ivar Jacobson (Rational Software)
    - finding reqs is just one set of activities in the whole chain that need to work seamlessly together: business design, system design, implementation, etc.
Problem Statement

Requirements

- Natural Language
- (Sometimes) IEEE Std. 830
  - Tasks descriptions
  - User’s documents
  - Interview reports
  - ...

Challenges

- Improve understanding of the problem space
- Provide guidance in order to deal aligned with reqs and conceptual modeling
- Provide traceability structures
Objectives

- Main Goal:

  “To provide a Requirements Engineering Environment with a sound, flexible, and precise transition to an OO-Method Conceptual Schema”
Objectives

- **Subgoals**
  - Defining an ontological background of terms
  - Defining a Requirements Model to represent external interactions of the system
    - To specify Functions, Communication, and Behavior at different abstraction levels
    - To deal with identification, organization, reasoning, and specification of high-level scenarios
  - Defining a Requirements Analysis Process
    - To propose an object-oriented high-level specification of scenarios
    - To define traceability structures between requirement specifications and conceptual schemas
    - To define integration mechanisms among individual scenario specifications into a coherent conceptual schema
  - Implementing a Requirements Engineering environment
Approaches for Requirements Engineering
Goal-driven approaches

- Modeling organizational objectives
- Representing the why part of RM
- E. Yu argues that explicit representation of goals in RM provides a criterion for requirements completeness
- Examples: i* (Yu, Mylopoulos), KAOS (Dardenne, Fickas, van Lamsweerde), EKD (Bubenko), ...

Some problems detected...

- Goal discovery is not a simple task
- Enterprise goals, which initiates the goal discovery process, do not reflect the actual situation but an idealized environmental one
- Goal reduction (goals of a goal) is not straightforward
- The fuzzy concept of a goal seems to be difficult to deal with
Approaches for Requirements Engineering

Scenario-driven approaches

- Many times scenarios are easier to get in the first place (recognized in cognitive studies on human problem solving)
- Scenarios describe how a user task can be achieved with a sequence of interactions with the system
- Scenarios can be represented as Use Cases, task descriptions,...
- Many approaches based on scenarios: OOSE (Jacobson), CREWS (Jarke, Rolland,...), Inquiry Cycle (Potts),...

Some problems detected...

- Modeling the behavior of a whole system may require a great multitude of scenarios
- Scenarios can be highly redundant (this makes changes tedious and can cause inconsistencies)
- UML interaction diagrams (often used) lack adequate expressiveness and semantics – misunderstandings)
Approaches for Requirements Engineering

Coupling goals and scenarios

- A. Cockburn uses goals as a means to structure Use Cases
- J. Leite and K. Pohl consider goals as contextual properties of scenarios
- A. Anton, C. Potts and C. Rolland use the goal and scenario combination to operationalize goals
- P. Haumer and K. Pohl to check whether or not the current system usage, captured through multimedia scenarios, fulfils its expected goals
- A. van Lamsweerde infers goals specifications from operational scenarios
- C. Rolland discovers new goals through scenario analysis
Approaches for Requirements Engineering
Synthesizing object behavior from scenarios

- The importance of generating behavioral descriptions from scenarios was noted early by Harel in 1987

- Whittle and Schumann propose characteristics for any synthesis algorithm:
  - Scenarios must be consistent each other
  - Scenarios will likely duplicate behaviors
  - Obtained behavioral descriptions should be readable
  - Obtained behavioral descriptions will be modified manually

- Whittle and Schumann (NASA Research)
  - Messages from SD with pre-post conditions annotations (OCL) to check valid event sequences
  - Annotated SDs are translated into a set of FSM (one per object)
  - Finally, the set of FSM for an object are merged using the annotations as a guide
Approaches for Requirements Engineering
Synthesizing object behavior from scenarios

- Somé (University of Montreal)
  - Alternatives for the same scenario are combined to obtain a single STD skeleton
  - Scenarios are obtained one by one, and merged their partial behavior with that obtained from scenarios previously acquired
  - Composing may unveil contradictions causing to revise the acquired requirements

- Koskimies (University of Tampere)
  - State Diagrams obtained from SD as a problem of grammatical inference. Analyst has to respond to questions to propose a solution

- Harel (Weizmann Institute)
  - Uses a graphical language LSCs (extension of MSC) including possible and mandatory behavior
  - Play-in / Play-out environment
  - Play-in builds LSCs automatically from a GUI
  - A Play-out process allows to check an intra-object behavior merging partial behaviors
Work Development Proposal: OO-Method with Requirements

**Mission Statement**

F1 F2 F3 F4 F5

**Use Case Model**

- Use Case Description
  - Name: 
  - Description: 
  - Steps:

**Object Model**

- Class 1
- Class 2
- Class 3

**Class:**

- `rent` state="rented"
- `return` state="free"

**Dynamic Model**

- `s1` `s2` `s3`

**Functional Model**

- **Class: Car**
  - `rent` state="rented"
  - `return` state="free"

**Presentation Model**

- **OASIS**
  - Automatic Translation

- **Java RDB**
  - Automatic Translation

- **VB RDB**

**OO-Method Conceptual Model**

**Execution Model**

- **Use Case Model**
  - RAP
  - Guided Translation

- **Object Model**
  - RAP
  - Guided Translation

- **Dynamic Model**
  - RAP
  - Guided Translation

- **Functional Model**
  - Automatic Translation
Work Development Foundations

- A system delivers a service to its environment by interacting with it

- Two dimensions to specify properties:
  - external interactions
  - internal composition
Foundations
Requirements Model – External Interactions

- At this level systems are viewed as black boxes
- Three kinds of properties describing external interactions
  - functions
    - atomic/complex. They are not components of a system
  - behavior
    - responsibilities structure
    - ordering over the time
  - communication
    - message exchange (actors)
    - between functions
Foundations
Requirements Model - Internal composition

- A system as an assemblage of parts that serves a useful purpose
- These internal parts interacts each other to accomplish the desired system responsibility

- From a specification of external interactions it is possible to design a system decomposition!
Requirements Model Techniques

- **Mission Statement (MS)**
  - overall functionality of the entire system (main responsibility)

- **Function Refinement Tree (FRT)**
  - the *root* is the mission statement (main goal)
  - *intermediate nodes*: groups of elementary functions or business areas
  - *leaves*: elementary functions

- **Use Case Model (UCM)**
  - shows system *functions* and *communications*
  - UC as an interaction between the system and the environment
  - the UC description is decomposed into steps (*behavior* - scenario textual specification)
Techniques MS and FRT

Mission Statement →

Groups of elementary functions → Car Rental System

Groups and functions →

Functions →

- FRT shows an hierarchical decomposition of business functions
- FRT relates the goal of the system to their functions (or groups) showing why the system must have each function (or group) in order to realize its mission
- All the functions of the system are identified and organized
- It does not say anything about the internal decomposition of the system
Techniques
Use Case Model

- Shows each system function as a Use Case
- Adds communication properties relating actors and Use Cases
- The lack of precision of when and how use Use Cases is a spread problem. We avoid this relating UCM to FRT

- The behavior of the function is specified by the UC description:
  - A template organize this information in three sections (as usual):
    - Summary, Basic Course, and Alternatives
  - Steps are classified according its nature:
    - General: outside of the system context
    - Actor/system communication: communication with the interface
    - System response: change inside of the system
## Use Case Model

### Use Case Description

<table>
<thead>
<tr>
<th>Use Case:</th>
<th>Sale items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors:</td>
<td>Customer (init), Cashier</td>
</tr>
<tr>
<td>Cross reference:</td>
<td>F1.a</td>
</tr>
</tbody>
</table>

**Description:** A customer arrives at a checkout with items to purchase. The Cashier records the purchase items and collects payment.

### Use case specification

<table>
<thead>
<tr>
<th>General</th>
<th>Actor/System communication</th>
<th>System response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a customer arrives at a POST checkout with items to purchase.</td>
<td>2. The cashier starts a new sale session.</td>
<td>3. The cashier creates a new sale.</td>
</tr>
<tr>
<td>4. The cashier records the identifier of each item.</td>
<td>5. The system determines the item price and description and add information to the current sales transaction.</td>
<td></td>
</tr>
</tbody>
</table>

### Summary

- Basic Course:
  - Types of steps

---

21
Requirements Analysis Process

OO-Method Conceptual Model

Execution Model

OO-Method Conceptual Model

Mission Statement

F1 F2 F3

F4 F5

Use Case Model

Requirements Model

Traceability Rules

Class 1

Class 2

Class 3

Object Model

Dynamic Model

Functional Model

Class: Car

[rent] state="rented"

[return] state="free"

Use Case Model

Use Case Description

Name: Description: Steps:

Guided Translation

Automatic Translation

OASIS

Java RDB

VB RDB

Presentation Model

Automatic Translation

RAP

Directed Translation
Requirements Analysis Process

- Each UC is specified identifying interacting *classes* and *messages*

- Structure and Notation

- Blocks Notation

- A **UML extension** is introduced (stereotypes) to classify messages according its nature: «signal», «service», «query», «connect»
Sequence Diagram
Stereotypes extension

- «Signal» stereotype

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Properties</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;Signal&gt;&gt;</td>
<td><strong>Direction</strong></td>
<td>Input</td>
<td>Related to messages where the origin is an actor class and the receiver is the boundary class system.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output</td>
<td>Related to messages where the origin is the boundary class system and the receiver is an actor class.</td>
</tr>
</tbody>
</table>

: User

: System

starts_rent

<<signal>>

introduce_rent_data

(ssn, type, platenumber, ...)

<<signal>>
### Sequence Diagram

**Stereotypes extension**

- «Service» stereotype

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Properties</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;&lt;Service&gt;&gt;</td>
<td>Kind of Change</td>
<td>New</td>
<td>An interaction that represents the creation of a new object instance of the receiver class. It’s a strong change of state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Destroy</td>
<td>An interaction that represents the destruction of an existing object of the receiver class. It’s a strong change of state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update</td>
<td>An interaction that represents the creation of a new object instance of the receiver class. It’s a strong change of state.</td>
</tr>
</tbody>
</table>
**«Query» stereotype**

- **Type**
  - **Population**
  - **State**

- **Result**
  - **<<name>>**
    - It's the name of the variable to store the result.
    - Syntax: `result_name = message name`

- **Description**
  - **--**
    - A natural language description of the query.

- **Multiplicity**
  - **Min. Max**
    - Number of objects participating in the query.

---

**Sequence Diagram**

**Stereotypes extension**

- a) Message before classification
  - «Contract»
  - «Rate»
  - `get_rate`

- b) Message after classification
  - «Contract»
  - «Rate»
  - `subtotal_rent = get_rate`
Sequence Diagram

Stereotypes extension

- «Connect» stereotype

<table>
<thead>
<tr>
<th>Stereotype</th>
<th>Properties</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>Multiplicity</td>
<td>0,N</td>
<td>Minimum number of objects in the receiver class that may be (des)connected to/from the sender of the interaction.</td>
</tr>
<tr>
<td>Maximum</td>
<td>Multiplicity</td>
<td>1,M</td>
<td>Maximum number of objects in the receiver class that may be (des)connected to/from the sender of the interaction.</td>
</tr>
<tr>
<td>Activity</td>
<td>Insert</td>
<td></td>
<td>The interaction is to connect the receiver to the sender.</td>
</tr>
<tr>
<td></td>
<td>Deleted</td>
<td></td>
<td>The interaction is to disconnect the receiver from the sender.</td>
</tr>
<tr>
<td>Exclusivity</td>
<td>Exclusive</td>
<td></td>
<td>This property is used only with the Insert Activity. It means that the receiver object/s will be connected for exclusive use of the object sender.</td>
</tr>
<tr>
<td></td>
<td>No Exclusive</td>
<td></td>
<td><em>No Exclusive</em> means that the receiver object could be connected to as many sender objects as desired.</td>
</tr>
</tbody>
</table>
Sequence Diagram
Stereotypes extension

- «Connect» stereotype example
RAP – An example

1- Functions are identified (organized in a FRT)

2- Communication is specified

3- Behavior is specified
   Textual: very high level
   SD: showing internal composition
RAP – Sequence Diagram (1/8)
(Relevant actors are identified)

: cashier

: System

Sale
items

Cashier
RAP – Sequence Diagram (2/8)
(Messages between actors and the system)

: cashier

(starts_a_sale)

:introduce_sale_data

: System

Cashier

Sale
items
RAP – Sequence Diagram (3/8)

(Internal messages, Sale and Sale-line are identified)

1. The sale starts when the cashier press the "START SALE" button.
2. The cashier starts a new sale session.
3. The cashier creates a new sale.
4. The cashier records the identification of each item.
5. The system determines the tax and tag of each item, then it calculates the total value of the current sale transaction.
RAP – Sequence Diagram (4/8)
(To create a sale-line it’s necessary an item)
RAP – Sequence Diagram (5/8)

(the cashier finishes the data entry)

: cashier

: System

: Sale

: sale-line

: item

(steps 1-6)

1. starts_a_sale
2. introduce_sale_data
3. create_sale()
4. until end of lines
5. create_sale_line()
6. entry_completed
7. select_one_item
8. get_item_info()
RAP – Sequence Diagram (6/8)

(Subtotal has to be calculated and also the item stock has to be updated)

: cashier

: System

: Sale

: sale-line

: item

(steps 2)

starts_a_sale

introduce_sale_data

(create_sale)

(create_sale_line)

(select_one_item)

(get_item_info)

(get_subtotals)

(entry_completed)

(change_stock)

(until end of lines)

(until end of lines)
RAP – Sequence Diagram (7/8)
(Final information about the sale is shown)
RAP – Sequence Diagram (8/8)
(Messages are stereotyped)
Function Decomposition Table (CRUD Table)

<table>
<thead>
<tr>
<th></th>
<th>Sale item</th>
<th>Return purchased item</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cashier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sale</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sale_line</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item</td>
<td>RU</td>
<td>RU</td>
<td></td>
</tr>
<tr>
<td>Order</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Return</td>
<td></td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Return_line</td>
<td></td>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Collaborations:
- The role of the class in one specific scenario determines the service alphabet for the class.
Traceability

"A software requirements specification is traceable if (i) the origin of each of its requirements is clear and if (ii) it facilitates the referencing of each requirement in future development or enhancement documentation" ANSI/IEEE Std 830-1998

Gotel’s classification
(Pre-RS and Post-RS traceability)
Traceability

- In our approach we subdivide Post-RS traceability in:

  - **Weak traceability** is concerned with those aspects of the RAP that will be translated to the CS and that can be changed or even deleted on destination.
    
    This means that elements with weak traceability will be only *proposed* as elements in the CS.

  - **Strong traceability** is concerned with those aspects of the RAP that will be translated to the CS and that cannot be changed or deleted on destination.
    
    This means that elements with strong traceability will be *mandatory* elements in the CS.
Traceability from:

Requirements Model

Conceptual Model

Presentation Model

Object Model

Dynamic Model

Use Case Diagram

Sequence Diagram

FRT

F1 F2 F3

F4 F5

Inheritance

HAT

Classes

Attributes

Local transactions

Events

Agents

Associations

Integrity constraints

States

Transitions

Global transactions

Triggers

Class 1

Class 2

C1
Traceability Rules Catalog

- They specify structured mechanisms to analyze RM and SD to generate the main components of a Conceptual Schema (CS).

- The resultant CS is a skeleton that represents the main framework that should be refined and completed to have a precise representation of the software to be developed.

- These rules (38) are grouped according the kind of CS element they generate:
  - Classes, Attributes, Services, Association, Cardinalities, Aggregation, Agent, Inheritance, Conditions and Constraints, STD and OID diagrams, and Presentation Model.
Rule 14. For every message between two classes labeled with the stereotype <<service/new>> where both classes are distinct from the system class, an association relationship between these classes will be generated.

Traceability: strong

Conceptual Schema generated

[Diagram of a process involving classes and messages with stereotypes and relationships]
Rule 15. For every message between two classes labeled with the stereotype <<connect>> an association relationship between these classes will be generated.

- **Traceability**: strong
Rule 16. For every message with the stereotype <<service/new>> or <<connect>> where classes using role names appears, an association relationship between these classes will be generated using these role names on the ends of the relationship. Traceability: weak
View of a generated traceable Class Diagram

- Customer -> Contract
  - 1..1
  - 0..*

- Contract -> Car
  - 1..1
  - 1..1

- Car -> Rate
  - 0..1
  - 1..1

- Car -> Sale
  - 1..1
  - 0..*

- Car -> ExtraContract
  - 1..1
  - 0..*

- ExtraContract -> ExtraType
  - 1..1
  - 0..*

- Operation
  - 0..1
  - 0..*

- InsuranceCompany
  - 1..1
  - 0..*

- Insurance
  - 0..1
  - 1..1

- User

- Administrator

Questions:
- Why this cardinality is 1..1?
- Do I need this class?
- Should I change this cardinality to 0..M?
- Why this relationship is here?
- May I delete this class?
- One week after the interview with customers: Are there Sales in the system?
A Requirements Engineering Environment

RETO (Requirements Engineering TOol)

- Gives support to the Requirements Model and the Requirements Analysis Process
- Developed using the OlivaNova Modeling Software®
- A joint team between CARE Tech. and UPV
- A tool generated by a tool
RETO Metamodel
Function Refinement Tree (Form-based version)
FRT management options
Use Case specification (form-based version)

Step creation window
Function Refinement Tree (Icon-based version)
FRT management options

- Manually coded
- User friendly
- Usual tree component operations:
  - move node position
  - change parent
  - change name selected
  - edit/delete selected
  - shows description of selected
  - expand/collapse
  - resize window
- Synchronized with the database
FRT + Use Case Diagram + Sequence Diagram (Icon-based version) A typical icon-graphical interface

Usual diagrams operations:
- change diagram view with selected node
- UCD and FRT synchronized
- move UC and packages
- zoom in/out
- change name selected
- edit/delete selected
- create/modify actors
- establish relationships: actor/actor, actor/UC, UC/UC
- create/modify classes
- create messages
- assign/change stereotypes
- create blocks
Use Case specification

Forms are reused, some of them need to be more user friendly, yet...
Conclusions
Conclusions

Contributions

- Definition of a Requirements Model
  - Integration of three techniques (Mission Statement, Function Refinement Tree, and Use Case Model) into a three-level abstraction structure
  - A mechanism to identify and organize the system functionality starting from the Mission Statement and refining it until elementary functions
  - Integration of FRT and UCM to avoid the classical Use Case problem of abstraction level definition
  - A new Use Case template to organize and trace textual Use Case specifications

- Definition of traceability structures
  - Guidelines to move from textual UC descriptions to SD (according to the nature of steps and messages)
Contributions

- Messages classification
  - Stereotypes to label different kind of interaction between objects

- Traceability rules catalog
  - 38 traceability rules providing precise mappings between requirements and a conceptual schema
  - This provides a framework for integrating different scenario structures into a conceptual schema structure
  - A classification of Post-RE traceability: strong and weak

- RETO, a Requirements Engineering environment (under development)
  - A precise metamodel specification using OO-Method
  - Tool automatically generated based on the Requirements Model metamodel
  - A requirements repository in DB and XML (with future facilities for multi-user)
Contributions

- RETO, a Requirements Engineering environment (contd)
  - XML communication with commercial tools (ON ME®) based on XSL Transformations
  - A new user friendly GUI under development
  - A first full version of the RETO is planned for the first quarter of 2004

- There were developed case studies
  - Three medium size projects at CARE Technologies
    - Document management system, Title deed system, and a Project management system
  - Several projects with students as part of their final course project
  - As a result, in general, users feel that the efforts spend on requirements specification are useful
  - The tool will be used in a Software Development course next year

- In addition to the RETO commercial version, an academic version is also under development with students
Publications

Journals (6)


Publications

- **Book chapter (1)**

- **International Conferences and Workshops (16)**
Publications

- **International Conferences and Workshops (16) contd.**

- **National Conferences and Workshops (6)**
Current and Future Research

- Development of an **Hyper-Dictionary**. Capturing terms of a problem domain in the context of requirements specification (a project under development among the Univ. Twente, UPV, and CARE Tech)

- Development of more **case studies** to verify and refine the set of traceability rules catalog
  - At CARE Technologies
  - At the University: in a last year course on Software Development and final course projects

- Enhance the current Requirements Model with **non-functional requirements** specification

- Finish the academic and industrial version of **RETO**
Current and Future Research

- **Business Modeling.** Integration of this proposal with business modeling techniques. Identifying scenarios or functions in the context of goals seems to be an interesting research area.

- **Requirements for the web.** Taking into account requirements related to presentation and navigation structures.

- **User interface.** Enhance the study to capture properties related to the communication between users and the system interface.

- **Requirements quality.** Study and propose a quality model for requirements based on the standard ISO 9126.

- **Development of test cases.** Verification of execution traces against Sequence Diagrams.
Current and Future Research

- **Project estimation from Requirements Models.** Due to traceability from Requirements to Conceptual Schemas, and later to code. A functional size estimation based on requirements functions points will be proposed.

End of the presentation