Requirements Capture and Specification for Enterprise Applications: a UML Based Attempt

Christine Choppy (LIPN, Université Paris XIII, France)
Gianna Reggio (DISI, Università di Genova, Italy)
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications: the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
Motivation

- a software development method for enterprise applications
Motivation

- a software development method for enterprise applications combines
  the use of the structural concepts provided by problem frames
Motivation

- a software development method for enterprise applications combines
  - the use of the structural concepts provided by problem frames
  - the use of the UML notation
Motivation

- a software development method for enterprise applications combines
  - the use of the structural concepts provided by problem frames
  - the use of the UML notation

Method

1. Match the problem with a problem frame
2. Develop the UML description
Motivation

- a software development method for enterprise applications combines
  - the use of the structural concepts provided by problem frames
  - the use of the UML notation

Method

1. Match the problem with a problem frame
2. Develop the UML description

Guidelines to develop all the required artifacts dedicated choice of appropriate UML diagrams predefined schemas or skeletons for their contents
Motivation

- a software development method for enterprise applications combines
  - the use of the structural concepts provided by problem frames
  - the use of the UML notation

Method

1. Match the problem with a problem frame
2. Develop the UML description

Guidelines to develop all the required artifacts
- dedicated choice of appropriate UML diagrams
- predefined schemas or skeletons for their contents

- modelling the domain, the requirements capture and specification, and their relationships
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications : the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
Introduction

- "Enterprise Applications are about the display, manipulation and storage of large amounts of often complex data and the support or automation of business processes with that data." (M. Fowler)
Introduction

“Enterprise Applications are about the display, manipulation and storage of large amounts of often complex data and the support or automation of business processes with that data.” (M. Fowler)

patterns: “ready-to-use” structures drawn from experience, granularity differs

problem frames → overall problem structure
architectural styles → overall system structure
design patterns → design structure
Introduction

- “Enterprise Applications are about the display, manipulation and storage of large amounts of often complex data and the support or automation of business processes with that data.” (M. Fowler)
- patterns: “ready-to-use” structures drawn from experience, granularity differs
  problem frames $\rightarrow$ overall problem structure
  architectural styles $\rightarrow$ overall system structure
  design patterns $\rightarrow$ design structure
- UML notation, variety, consistency $\rightarrow$ guidance
Introduction

▶ “Enterprise Applications are about the display, manipulation and storage of large amounts of often complex data and the support or automation of business processes with that data.” (M. Fowler)

▶ patterns: “ready-to-use” structures drawn from experience, granularity differs

- problem frames → overall problem structure
- architectural styles → overall system structure
- design patterns → design structure

▶ UML notation, variety, consistency → guidance

▶ here: a new problem frame for Enterprise Applications, the Enterprise frame, composed of two parts (the Business Frame, devoted to the domain description, and the EA Frame), and UML based associated development method
a small e-commerce site, $\mu$EC,

- clients buy products chosen in a browsable catalogue and pay using an external payment system
- the products are produced by an external factory
- stocked, and then delivered by a dedicated department of the company
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications: the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
Problem Frames (M. Jackson)

- can be used by themselves or in combination
- a good tool to tackle with a first structuring of problems
- diagram: involved domains, requirements, design, interfaces
- five basic problem frames, variants

We propose associated development methods
A basic problem frame: transformation

There are some computer-readable input files whose data must be transformed to give certain required output files. The output data must be in a particular format, and it must be derived from the input data according to certain rules. The problem is to build a machine that will produce the required outputs from the inputs.
A basic problem frame: commanded behaviour

There is some part of the physical world whose behaviour is to be controlled in accordance with commands issued by an operator. The problem is to build a machine that will accept the operator’s commands and impose the control accordingly.
A basic problem frame: commanded information

\[ \text{Real world} \]

\[ \text{Answering machine} \]

\[ \text{Enquiry operator} \]

\[ \text{Answer rules} \]

\[ \text{C1 phenomena are referred to later as Events issued by the Real world} \]

\[ \text{E1 are Enquiries from the Enquiry operator} \]

\[ \text{E2 are Display Acts and Error Messages displayed by the Answering Machine to the Enquiry Operator} \]
Problem Frame notation

<table>
<thead>
<tr>
<th>Machine domain</th>
<th>Designed domain</th>
<th>Given domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

Lexical domain

Biddable domain

Causal domain
Problem Frame notation

- Machine domain
- Designed domain
- Given domain

- Lexical domain
- Biddable domain
- Causal domain

D1 \(\text{D1!ph1(args)}\) \(\text{D2!ph2(args)}\) D2

Interface and shared phenomena
Problem Frame notation and extensions

```
D1

...     ...

Dn

ph-1(args)

Requirements

ph-n(args)
```

A service $S$ groups a set of phenomena:

- Internal (non-sharable) phenomena $i-ph$
- External phenomena $e-ph$

$S = \{D_1!ph^{-i}(...), ..., D_2!ph^{-k}(...), ...\}$
A service $S$ groups a set of phenomena internal (non sharable)/ external phenomena $i$-ph/$e$-ph.

$$S = \{D1!\text{ph}_i(...), ..., D2!\text{ph}_k(...), ...\}$$
A service $S$ groups a set of phenomena internal (non sharable)/ external phenomena $i$-ph/$e$-ph.

$$S = \{D1!ph_1(...), \ldots, D2!ph_k(...), \ldots\}$$

A composite phenomena $\text{CPH}$
Overview

Motivation
Introduction
Problem Frame notation

The Enterprise Frame

Enterprise Applications : the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
The Enterprise Frame

composed of 2 parts

- (reuseable) **Business (domain) Frame** describing the business (in large) to be managed by the EA business rules (subtle)

- **EA Frame**: “classical” problem frame, EA machine, domains and requirements.
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications: the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
Enterprise Applications : the Business (domain) Frame

Business Frame domains (three categories) :

- **BO** business objects (causal domains)
  - no distinction between working for the business and business actors

- **BW** business workers acting in the business (biddable domains)
  - no distinction between working for the business and business actors

- **EC** or **EB** external systems : entities external to the business used for outsourcing some activities (causal or biddable domains)
  - biddable → 1+ internal phenomena
  - causal → 1+ external phenomena (no internal)
Enterprise Applications: the Business (domain) Frame

Business Frame domains (three categories):

- BO business objects (causal domains)
  - no distinction between working for the business and business actors
- BW business workers acting in the business (biddable domains)
- EC or EB external systems: entities external to the business used for outsourcing some activities (causal or biddable domains)

biddable $\rightarrow$ 1+ internal phenomena
causal $\rightarrow$ 1+ external phenomena (no internal)
business case: cooperation for relevant procedures (composite phenomena)
μECcase : Business Frame

Client

Payment System

Browse

Put money Get money

Put order

Catalogue

Update

Orders

Manager

Delivery Dept

Stock

Refill

Factory

modular decomposition ...
μECBusiness Frame: Put order Business Case

Client

putOrder BW

Payment System

getMess(...) putProd(...) Catalogue BO

addOrder(...) Orders BO

takeProd(...) Stock BO

putOrder

getMess(...) putProd(...) Catalogue

addOrder(...) Orders

takeProd(...) Stock

BW

EC
BF associated (UML) Business Model

precise method of Astesiano-Reggio

- A class diagram: a class for each BF domain
  - biddable \rightarrow active
  - causal \rightarrow passive
  - stereotypes: \( \ll bo \rr \) (“Business Object”), \( \ll bw \rr \) (“Business Worker”) and \( \ll es \rr \) (“External System”), cf. domain markings BO, BW and EB, EC.

- internal phenomenon \rightarrow private operation
- autonomous acts (stereotype \( \ll A \rr \))
- external phenomena: operations of the corresponding class
BF associated (UML) Business Model

precise method of Astesiano-Reggio

- A class diagram: a class for each BF domain
  biddable $\rightarrow$ active
  causal $\rightarrow$ passive
  stereotypes: $\langle bo \rangle$ (“Business Object”),
  $\langle bw \rangle$ (“Business Worker”) and $\langle es \rangle$ (“External System”), cf. domain markings BO, BW and EB, EC.
  internal phenomenon $\rightarrow$ private operation
  autonomous acts (stereotype $\langle A \rangle$)
  external phenomena: operations of the corresponding class
- behaviour: active class $\rightarrow$ statechart
  passive class $\rightarrow$ operations behaviour
BF associated (UML) Business Model

precise method of Astesiano-Reggio

▶ A class diagram: a class for each BF domain
  biddable → active
  causal → passive
  stereotypes: ≪bo≫ (“Business Object”),
  ≪bw≫ (“Business Worker”) and ≪es≫ (“External System”), cf. domain markings BO, BW and EB, EC.
  internal phenomenon → private operation
  autonomous acts (stereotype ≪A≫)
  external phenomena: operations of the corresponding class

▶ behaviour: active class → statechart
  passive class → operations behaviour

▶ A description of each business case:
  UML collaboration and activity diagram
μECBusiness Model: class diagram

**Product**
- id: ProdID
- price: Euro
- descr: String

**Order**
- client: ClientID
- prod: ProdID
- quant: Int
- date: Date
- id: OrderID
- status: {wait, deliv}

**Manager**

**Client**
- getMess(String)
- shown(Set(Product))
- lookCatalogue()
- putOrder(ProdID, Int, Date, ClientID)

**Orders**
- os: Set(Order)
- addOrder(ClientID, ProdID, Int, Date)
- delivered(OrdID)
- first(): Order

**Factory**
- deliverProd(ProdID, Int)

**Order**

**DeliveryDept**
- deliver(Order)
- delivered(OrdID)

**PaymentSystem**
- pay(Euro, ClientID)

**Stock**
- receiveProd(ProdID, Int)
- availableProd(ProdID): Int
- takeProd(ProdID, Int): Bool

**Catalogue**
- ps: Set(Product)
- getProd(ProdID): Product
- contents: Set(Product)

**Orders**
- os: Set(Order)
- addOrder(ClientID, ProdID, Int, Date)
- delivered(OrdID)
- first(): Order

**Client**
- getMess(String)
- shown(Set(Product))
- lookCatalogue()
- putOrder(ProdID, Int, Date, ClientID)

**Manager**

**Orders**
- os: Set(Order)
- addOrder(ClientID, ProdID, Int, Date)
- delivered(OrdID)
- first(): Order

**Factory**
- deliverProd(ProdID, Int)

**Order**

**DeliveryDept**
- deliver(Order)
- delivered(OrdID)

**PaymentSystem**
- pay(Euro, ClientID)

**Stock**
- receiveProd(ProdID, Int)
- availableProd(ProdID): Int
- takeProd(ProdID, Int): Bool

**Catalogue**
- ps: Set(Product)
- getProd(ProdID): Product
- contents: Set(Product)
µECBusiness Model : Catalogue Behaviour View

method newProd(PI,E,S) {
P = create(Product); P.id = PI; P.price = E;
P.descr = S; ps = ps->including(P)}

countext deleteProd(PI) post :
not ps.id ->includes(PI)

countext changePrice(PI,E) post :
ps->select(id = PI).price = E
μECBusiness Model: Put Order business case

collaboration

Client

Put Order

Catalogue

Stock

Orders

PaymentSystem

CI: ClientID
PI: ProdID
Q: Int

CT

ST

ORS

CI: ClientID
PI: ProdID
Q: Int

Put Order

CT

ST

ORS

CI: ClientID
PI: ProdID
Q: Int

Put Order

CT

ST

ORS

CI: ClientID
PI: ProdID
Q: Int
µECBusiness Model: Put Order business case

activity diagram
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications: the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
The EA Problem Frame
A BO-D is a designed domain with a full model of the business object BO that EA works with.
An ES-I/BW-I domain corresponds to some limited information about ES/BW that EA needs to interact with it (e.g., its name and the way to access it).
some checks

- Each enclosed domain must be connected by a chain of business cases having a common participant with an outside domain
- There should be at least one domain outside
- ...
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications: the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
Requirement Specification

precise method of Astesiano-Reggio

▶ each frame domain → a class diagram + (biddable → active, causal → passive) stereotypes (≪bo≫, ≪bw≫ and ≪es≫)

shared phenomena operations

a class for the EA :

shared external phenomena → operations
Requirement Specification

precise method of Astesiano-Reggio

- each frame domain → a class diagram +
  (biddable → active, causal → passive)
  stereotypes (≪bo≫, ≪bw≫ and ≪es≫)
  shared phenomena operations
  a class for the EA:
  shared external phenomena → operations

- A complete behaviour definition of all the
  ≪bo≫ classes and possibly others
Requirement Specification

precise method of Astesiano-Reggio

- each frame domain → a class diagram +
  (biddable → active, causal → passive)
  stereotypes (≪bo≫, ≪bw≫and ≪es≫)
  shared phenomena operations
  a class for the EA:
  shared external phenomena → operations

- A complete behaviour definition of all the
  ≪bo≫ classes and possibly others

- A use case diagram, use cases description (actors:
  domains connected with EA), use case behaviour:
  a statechart associated with the class EA, such that
  - events: timed or call events,
  - conditions w.r.t. its attributes only
  - actions: attribute upd or actors operations calls
**Motivation**

**Introduction**

**Problem Frame notation**

**The Enterprise Frame**

**Enterprise Applications : the Business (domain) Frame**

**The EA Problem Frame**

**Requirement Specification**

**Conclusions**

---

**μEC Requirement Specification : Class Diagram**

- **Product**
  - id: ProdID
  - price: Euro
  - descr: String

- **Euro**
  - Date
  - OrdID

- **Order**
  - client: ClientId
  - prod: ProdId
  - quant: Int
  - date: Date
  - id: OrderId
  - status: {wait, deliv}

- **Client**
  - getMess(String)
  - show(Set(Product))
  - passWdIs(String)

- **Factory**
  - deliver(Order)

- **PaymentSystem**
  - pay(Euro, ClientID)

- **Stock**
  - receiveProd(ProdID, Int)
  - availableProd(ProdID): Int
  - takeProd(ProdID, Int): Bool

- **Orders**
  - os: Set(Order)
  - addOrder(ClientID, ProdID, Int, Date)
  - first(): Order

- **Manager**
  - correct(ClientID, String): Bool
  - newPsw(): String

- **ClientRecords**
  - cls: ClientRecords
  - putOrder(ProdID, Int, ClientID, Date)
  - login(ClientID, String)
  - getOrderToDeliver()
  - delivered(Order)
  - register(ClientID)
  - refilledStock(ProdID, Int)

- **Catalogue**
  - ps: Set(Product)
  - getProd(ProdID): Product
  - contents: Set(Product)

- **PaymentSystem**
Requirements Capture and Specification for Enterprise Applications: a UML Based Attempt

Christine Choppy (LIPN, Université Paris XIII, France)
Gianna Reggio (DISI, Università di Genova, Italy)

Motivation
Introduction
Problem Frame notation
The Enterprise Frame
Enterprise Applications: the Business (domain) Frame
The EA Problem Frame
Requirement Specification
Conclusions

μEC Requirement Specification: Use Case Diagram

- C: Client
  - microEC
    - Register
    - Deliver
    - Put Order
    - Browse
    - Update
    - Refill
- M: Manager
  - DP: Delivery Dept.
  - ORS: Orders
  - ST: Stock
  - PS: Payment System
  - CT: Catalogue
  - F: Factory
μECE Requirement Specification: Put Order

Use Case Description

![Diagram of the use case description for the put order scenario.]

- **Login (CI, x)**
  - [not cls.correct(CI, x)]
  - /C.getMess("wrong psw")

- **Put Order (PI, Q, CI, now)**
  - [ST.get(Prod(PI, Q) and PS.pay(Q*CT.getProd(PI).price))]
  - C.getMess("Accepted")

- **Put Order (PI, Q, CI, now)**
  - [not ST.getProd(PI, Q)]
  - /C.getMess("Refused")

- **Put Order (PI, Q, CI, now)**
  - [not PS.pay(Q*CT.getProd(PI).price)]
  - /C.getMess("Refused")
Overview

Motivation

Introduction

Problem Frame notation

The Enterprise Frame

Enterprise Applications: the Business (domain) Frame

The EA Problem Frame

Requirement Specification

Conclusions
Conclusions

- a software development approach for Enterprise Applications
  
  1. match the Business Frame and the EA Frame with the problem
  2. model the various frames parts following the proposed UML diagrams
Conclusions

- a software development approach for Enterprise Applications
  1. match the Business Frame and the EA Frame with the problem
  2. model the various frames parts following the proposed UML diagrams

- problem frames are very good at providing a first requirement structure that is invaluable to start the analysis of a problem and understand its nature. A means to reuse experience helpful to start a complex problem analysis with some structuring concepts in mind.
Conclusions

- A software development approach for Enterprise Applications
  1. match the Business Frame and the EA Frame with the problem
  2. model the various frames parts following the proposed UML diagrams

- Problem frames are very good at providing a first requirement structure that is invaluable to start the analysis of a problem and understand its nature. A means to reuse experience helpful to start a complex problem analysis with some structuring concepts in mind.

- Reduce the time spent to decide which UML constructs to use and how to model the domain and the requirements
Conclusions (follwd)

- a more direct path to the UML models, improved models quality (relevant issues are addressed, a uniform style is offered)
Conclusions (followed)

- a more direct path to the UML models, improved models quality (relevant issues are addressed, a uniform style is offered)
- requiring that the modelling of the domain and of the problem is accompanied by their "framing" helps the developer to manage complex things, and offers a support to navigate the complex UML models produced
Conclusions (follwd)

- a more direct path to the UML models, improved models quality (relevant issues are addressed, a uniform style is offered)
- requiring that the modelling of the domain and of the problem is accompanied by their ”framing” helps the developer to manage complex things, and offers a support to navigate the complex UML models produced
- essence: use and combination of the relevant underlying concepts, possible different notations (e.g. a graphical language for a different level of abstraction, UML, formal specification language, etc).