

Introduction

à la Modélisation par Objets :

UML à la rescousse !

Isabelle BLASQUEZ
@iblasquez

Janvier 2022



disponible en ligne sous <https://github.com/iblasquez/enseignement-but1-developpement>



Isabelle Blasquez

@iblasquez

Associate Professor in Software Engineering [#SoftwareCraftsmanship](#) [#iutagile](#)
[#limouzicodev](#) [@duchessfr](#) [@CodeWeekEU](#) [@MuseomixLIM](#) [@aperscope87](#)

Sur une feuille blanche,

Expliquer à l'aide d'un dessin
comment vous faites (feriez)
griller votre pain le matin

(n'oubliez pas vos nom, prénom et groupe au verso de la feuille)



<http://e.ggtimer.com>

Vous avez 3 minutes



Bravo !

vous venez de réaliser votre premier modèle !

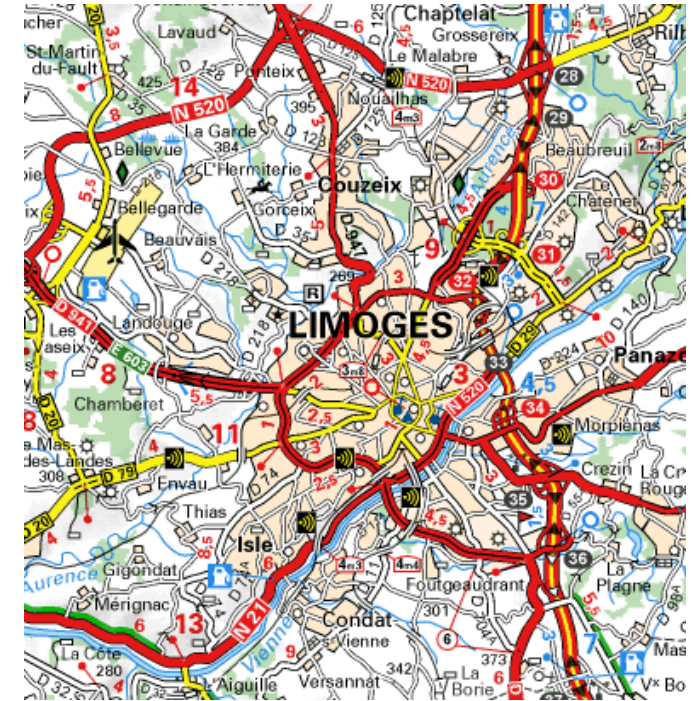


Quelques extraits de :
<http://www.drawtoast.com/gallery.html>

D'après une idée originale de Tom Wujec : <http://www.tomwujec.com/design-projects/draw-toast>

Modélisation

Qu'est-ce qu'un modèle ?



<http://www.viamichelin.fr>

Un modèle est une **abstraction (représentation abstraite)** de la réalité

(Image **simplifiée** du monde réel **selon un point de vue**
suffisante pour **comprendre** le système modélisé
et **répondre** aux questions que l'on se pose sur lui)

Un modèle permet ...

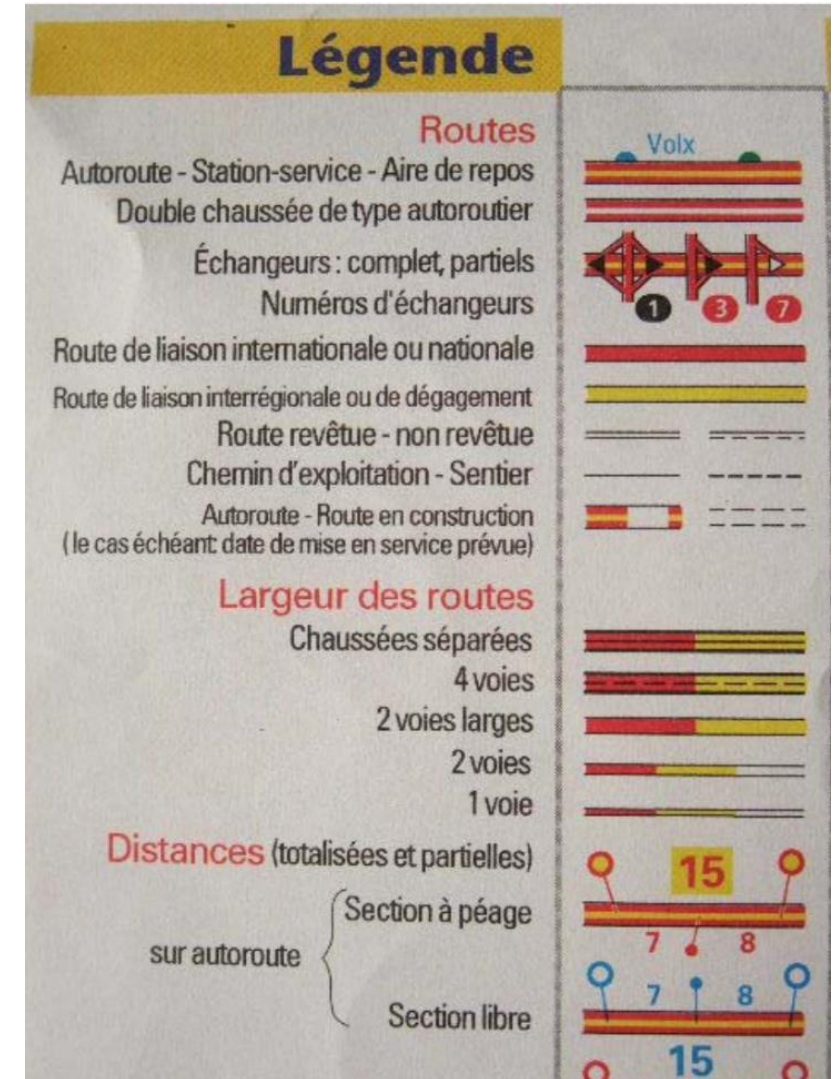
- de **comprendre** et **décrire la réalité** (en réduisant la complexité)
- de **simuler le monde réel**
- de **trouver** et **d'exprimer une solution** à un problème du monde réel
 - ... en **communiquant** à l'aide d'un **langage commun** composé d'un nombre restreint de **concepts**

Le métamodèle : une légende est nécessaire pour la bonne compréhension d'un modèle

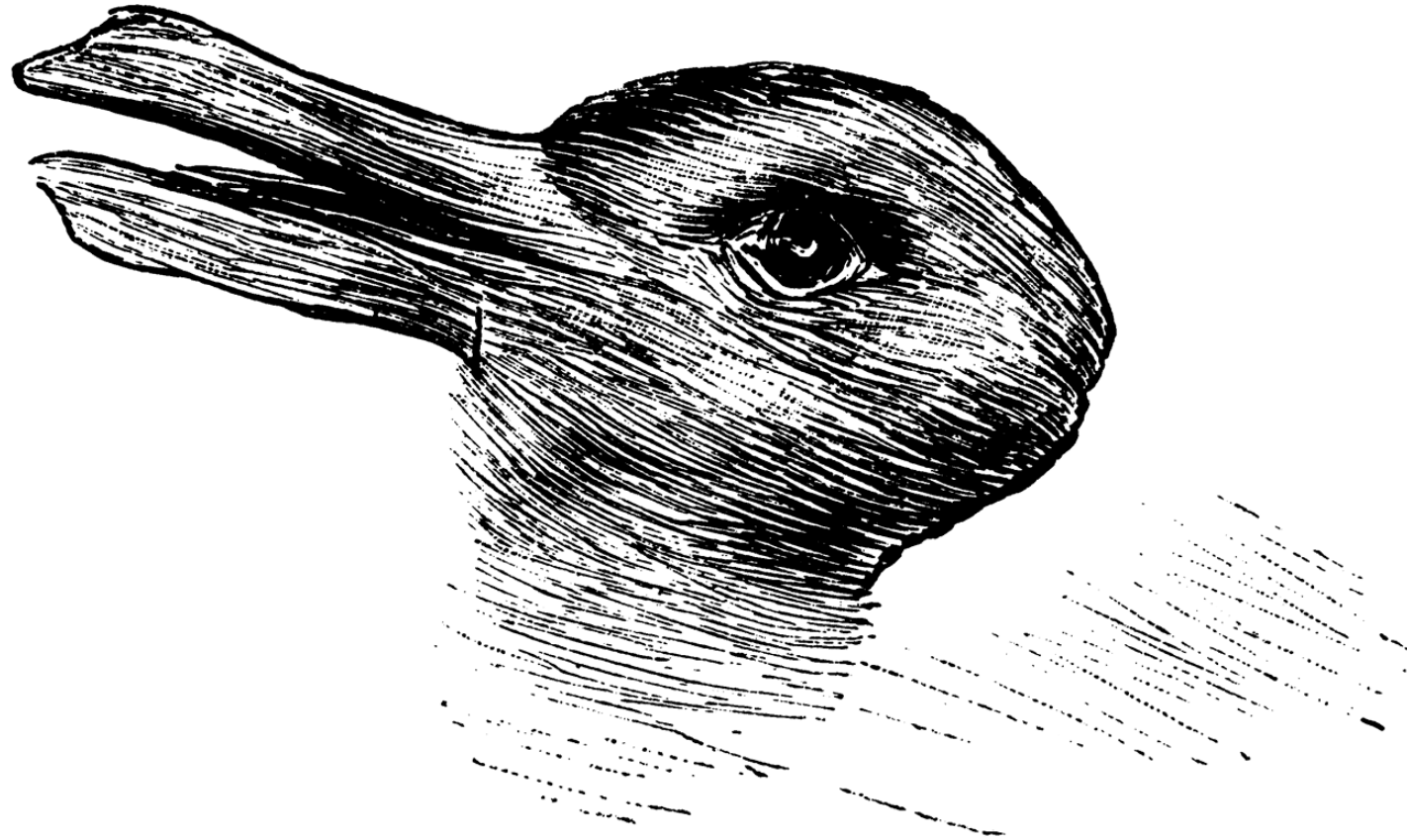
La légende, c'est le **langage** du modèle
A partir d'une **grammaire** précise et documentée,
elle permet d'interpréter les **concepts**/dessins
composant le modèle

La légende est, elle-même, un modèle !
⇒ On l'appelle un **métamodèle**

Un metamodelle est un **modèle** qui définit
le langage d'expression d'un modèle,
c.-à-d. le **langage de modélisation**.



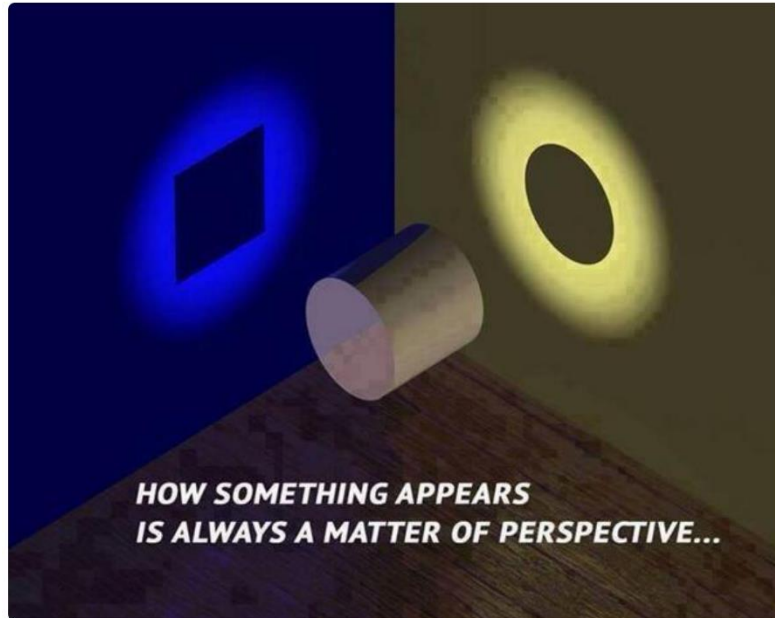
Quizz : Que représente ce modèle ?



Extrait : <http://www.laboiteverte.fr/>

Un modèle dépend forcément d'un point de vue

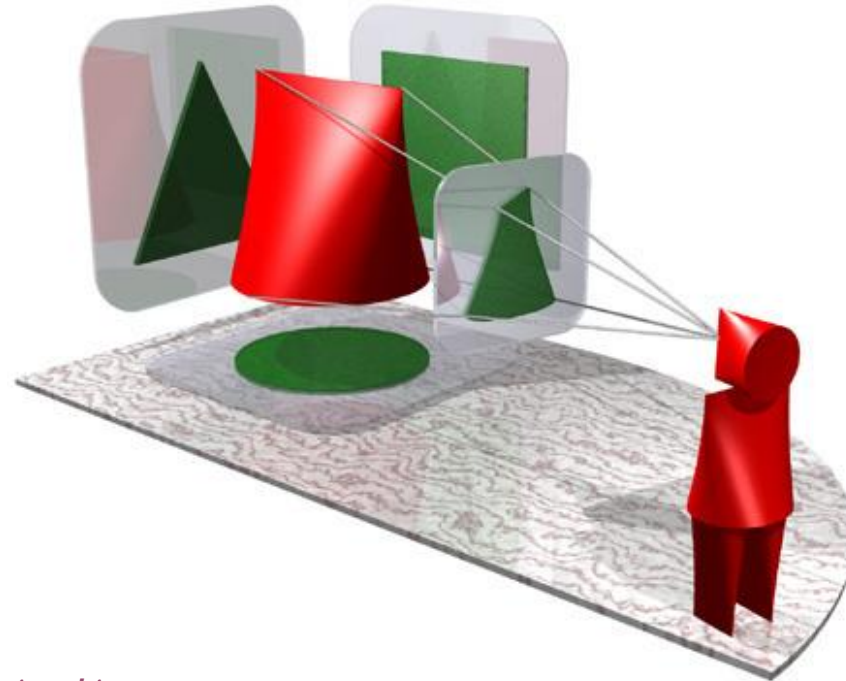
Place perspective on everything you look at.



Extrait :

<https://twitter.com/SciencePorn/status/424992760530481155>

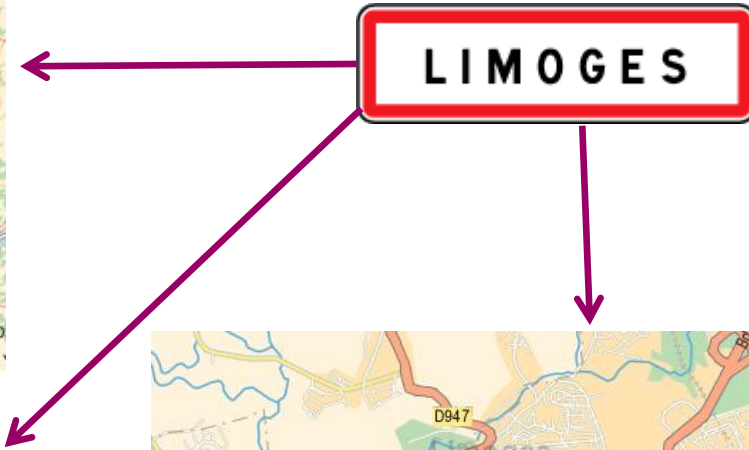
Tout est question de points de vue ...



Extrait :

<http://lucacardelli.name/Topics/TheoryOfObjects/ObjectSubject.html>

A detailed map of the Agen region in France. The Garonne river flows through the center, with Agen marked by a green shield icon. Major roads include the A20 (blue), N21 (red), E09 (green), and E62 (green). Surrounding towns such as Magnac-Laval, Guéret, Saint-Junien, and Saint-Yrieix-la-Perche are labeled. The map also shows smaller towns like Bessines-sur-Gartempe, Ambazac, and Bourgneuf.



Avantages d'un modèle (récapitulatif)

- **Abstrait** : Il fait ressortir les points importants tout en enlevant les détails non nécessaires
- **Compréhensible** : Il permet d'exprimer une chose complexe dans une forme plus facilement compréhensible par l'observateur
- **Précis** : Il représente fidèlement le système modélisé
- **Prédictif** : Il permet de faire des prévisions sur le système modélisé
- **Peu coûteux** : Il est bien moins coûteux à construire et étudier que le système lui-même

Un modèle ?



... des modèles !



**Un modèle est
construit pour
répondre
à une
problématique
donnée**

Un modèle est construit pour apporter une solution



... et le développement logiciel consiste à passer d'un espace Problème à un espace Solution



Ron Jeffries

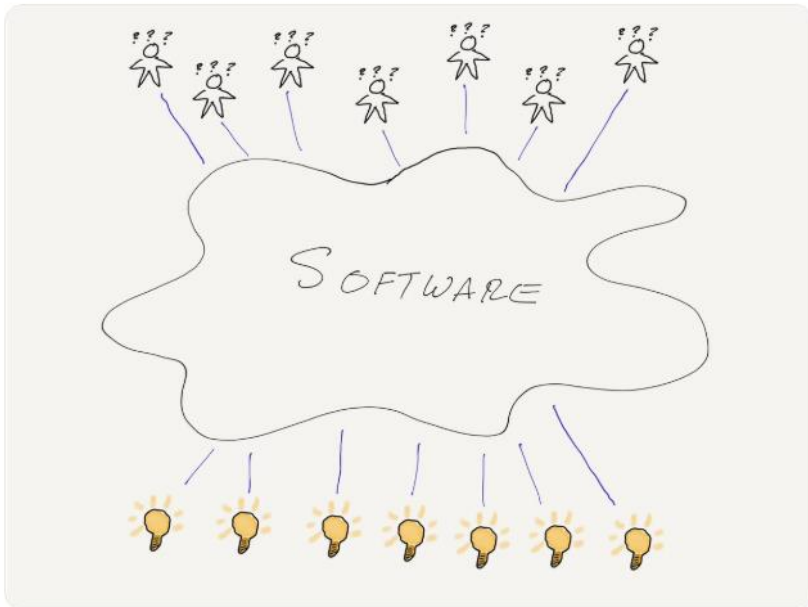
@RonJeffries

Abonné



The point of software development, it seems to me, is to connect people who have problems with the solutions to those problems, using software.

Traduire le Tweet



06:42 - 23 nov. 2018

Goal to achieve
Business context
Solution environment

Problem space

Development

Solution space

Design
Construction
Operation

Extraits : <https://twitter.com/RonJeffries/status/1065978837840904193>

<https://speakerdeck.com/lilobase/entre-industrialisation-et-artisanat-le-metier-de-developpeur-agilepaysbasque-2018> (vidéo :

<https://www.youtube.com/watch?v=bQfumbBN6YQ>)

Isabelle BLASQUEZ

En développement logiciel, UML permet de créer des modèles

(pour un paradigme de programmation orientée objet)

UML
(**U**nified **M**odeling **L**anguage)

Un langage pour modéliser ...

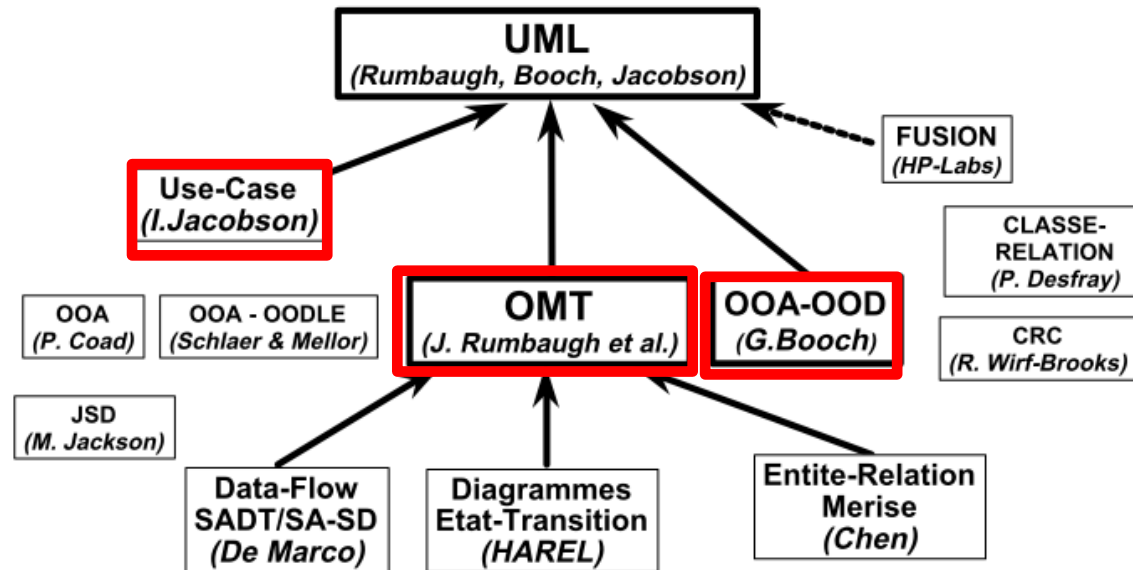
... des démarches par objet ...

Unifié par OMG

Historique d'UML

1995 : les « 3 Amigos » commencent un travail **d'unification des 3 démarches par objet**

- **OMT** : **O**bject **M**odeling **T**echniques (Rumbaugh de Rational Software)
- **OOD** : **O**riented **O**bject **D**esign (Booch de Général Electric)
- **OOSE** : **O**riented **O**bject **S**oftware **E**ngineering (Jacobson d'Ericsson)



Objectifs :

- Créer un langage de modélisation utilisable à la fois par **les hommes et les machines**,
- permettant de représenter des **systèmes** par des concepts objets.

Sites de référence pour UML

INTRODUCTION TO OMG'S UNIFIED MODELING LANGUAGE™ (UML®)



(Updated July 2005 to reflect formal adoption of UML 2.0 Superstructure.)

Large enterprise applications - the ones that execute core business applications, and keep a company going - must be more than just a bunch of code modules. They must be structured in a way that enables scalability, security, and robust execution under stressful conditions, and their structure - frequently referred to as their architecture - must be defined clearly enough that maintenance programmers can (quickly!) find and fix a bug that shows up long after the original authors have moved on to other projects. That is, these programs must be designed to work perfectly in many areas, and business functionality is not the only one (although it certainly is the essential core). Of course a well-designed architecture benefits any program, and not just the largest ones as we've singled out here. We mentioned large applications first because structure is a way of dealing with complexity, so the benefits of structure (and of modeling and design, as we'll demonstrate) compound as application size grows large. Another benefit of structure is that it enables code reuse: Design time is the easiest time to structure an application as a collection of self-contained modules or components. Eventually, enterprises build up a library of models of components, each one representing an implementation stored in a library of code modules. When another application needs the same functionality, the designer can quickly import its module from the library. At coding time, the developer can just as quickly import the code module into the application.

Modeling is the designing of software applications before coding. Modeling is an Essential Part of large software projects, and helpful to medium and even small projects as well. A model plays the analogous role in software development that blueprints

and other plans (site maps, elevations, physical models) play in the building of a skyscraper. Using a model, those responsible for a software development project's success can assure themselves that business functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendibility, and other characteristics, before implementation in code renders changes difficult and expensive to make. Surveys show that large software projects have a huge probability of failure - in fact, it's more likely that a large software application will fail to meet all of its requirements on time and on budget than that it will succeed. If you're running one of these projects, you need to do all you can to increase the odds for success, and modeling is the only way to visualize your design and check it against requirements before your crew starts to code.



<http://www.uml.org>

Spécifications officielles

<http://www.omg.org/spec/UML>

ABOUT THE UNIFIED MODELING LANGUAGE SPECIFICATION VERSION 2.5.1

2.5.1 • UML • SPECIFICATIONS

UML®

Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

Title: Unified Modeling Language
Acronym: UML®
Version: 2.5.1
Document Status: formal ⓘ
Publication Date: December 2017
Categories: Modeling Software Engineering
IPR Mode ⓘ RF-Limited ⓘ



TABLE OF CONTENTS

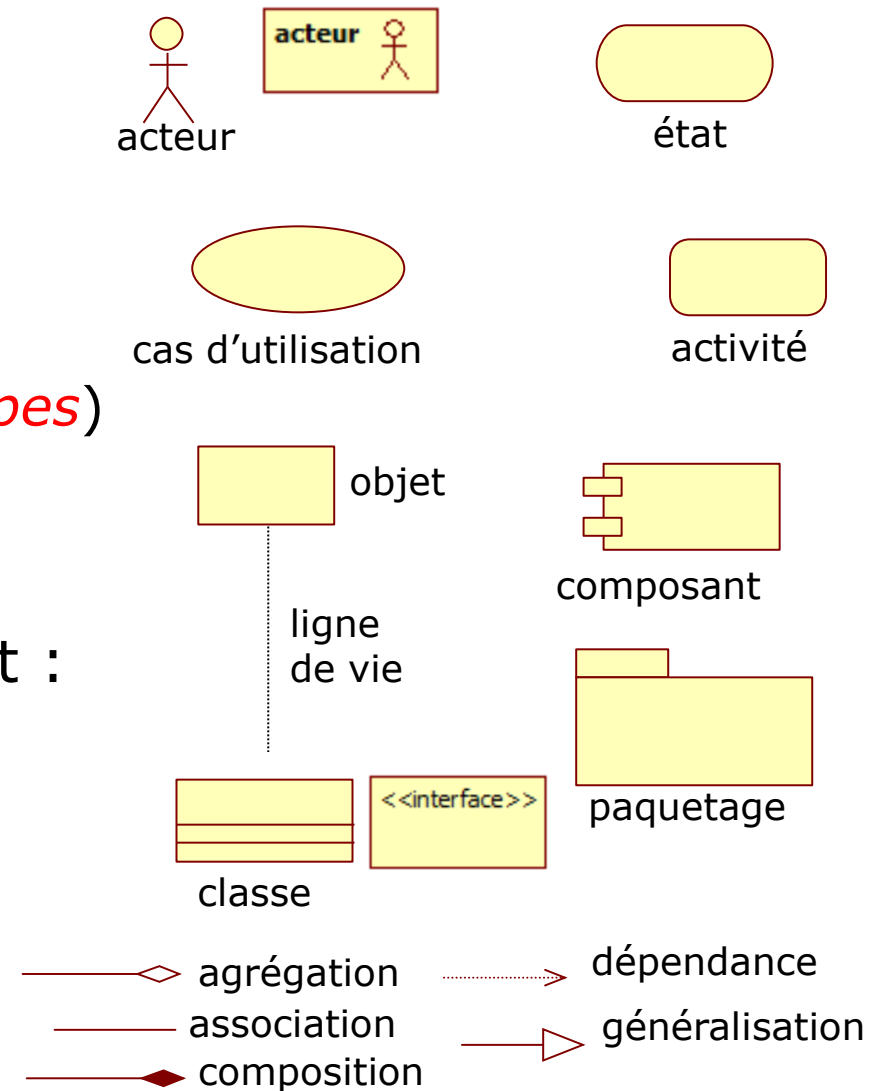
- About the Specification
- Specification Documents
 - Normative Documents
 - Normative Machine Readable Documents
 - Informative Documents
- History
 - Formal Versions
- Links

Quelques éléments du langage UML (méta-modèle)

Le langage UML propose une **notation** :
→ composée d'une **Syntaxe graphique**
→ et respectant une certaine **Sémantique**
(avec des points de variation sémantique et des *stéréotypes*)

Cette notation graphique, support du langage UML, est :

- Normalisée
- Semi-formelle
- Universelle
- Indépendante du langage de programmation
- Supportée par de nombreux outils



... et bien d'autres ...

Pyramide de modélisation de l'OMG

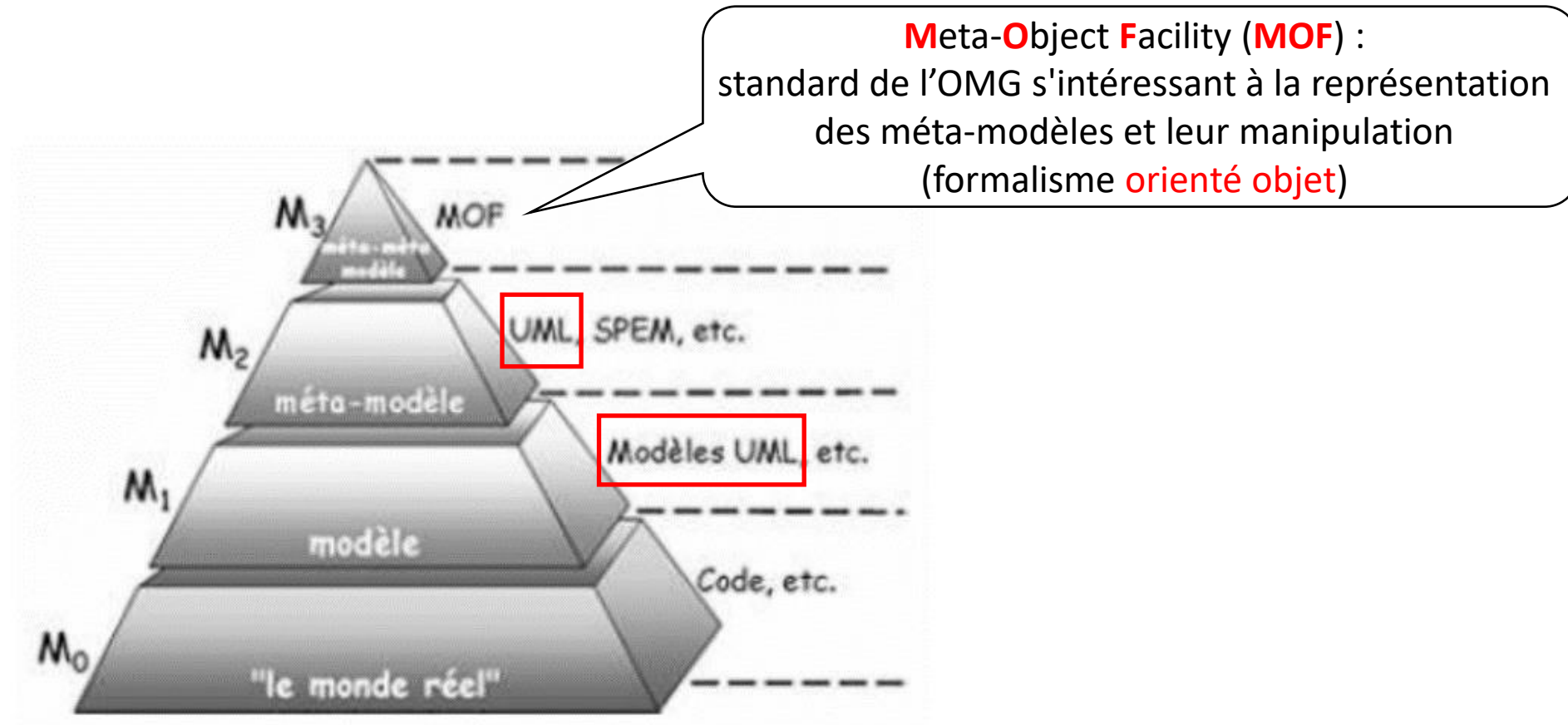


Figure 3. *Pyramide de modélisation de l'OMG (Bézivin, 2003)*

Figure 3 extraite de : *Etat de l'art sur le développement logiciel dirigé par les modèles.*

disponible sur : ftp://ftp.irit.fr/IRIT/MACAO/Article_TSI-IDM-final-coulette.pdf

Originalement proposé par Jean Bézivin. La transformation de modèles. Ecole d'Été d'Informatique CEA EDF INRIA 2003, cours #6.

UML propose une notation qui permet de représenter des modèles

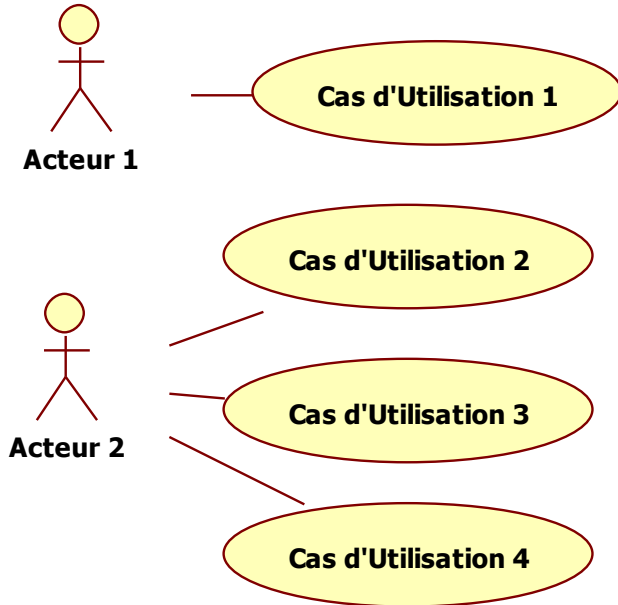


Diagramme des cas d'utilisation

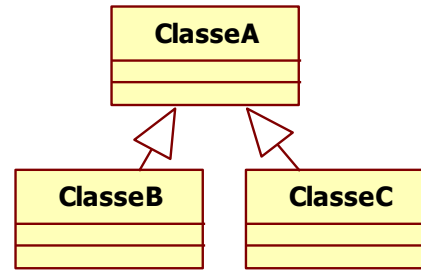


Diagramme
de classes

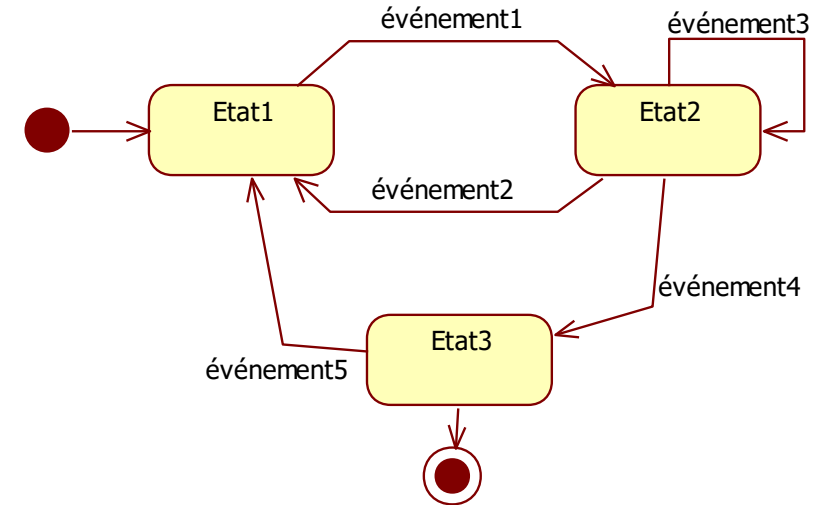
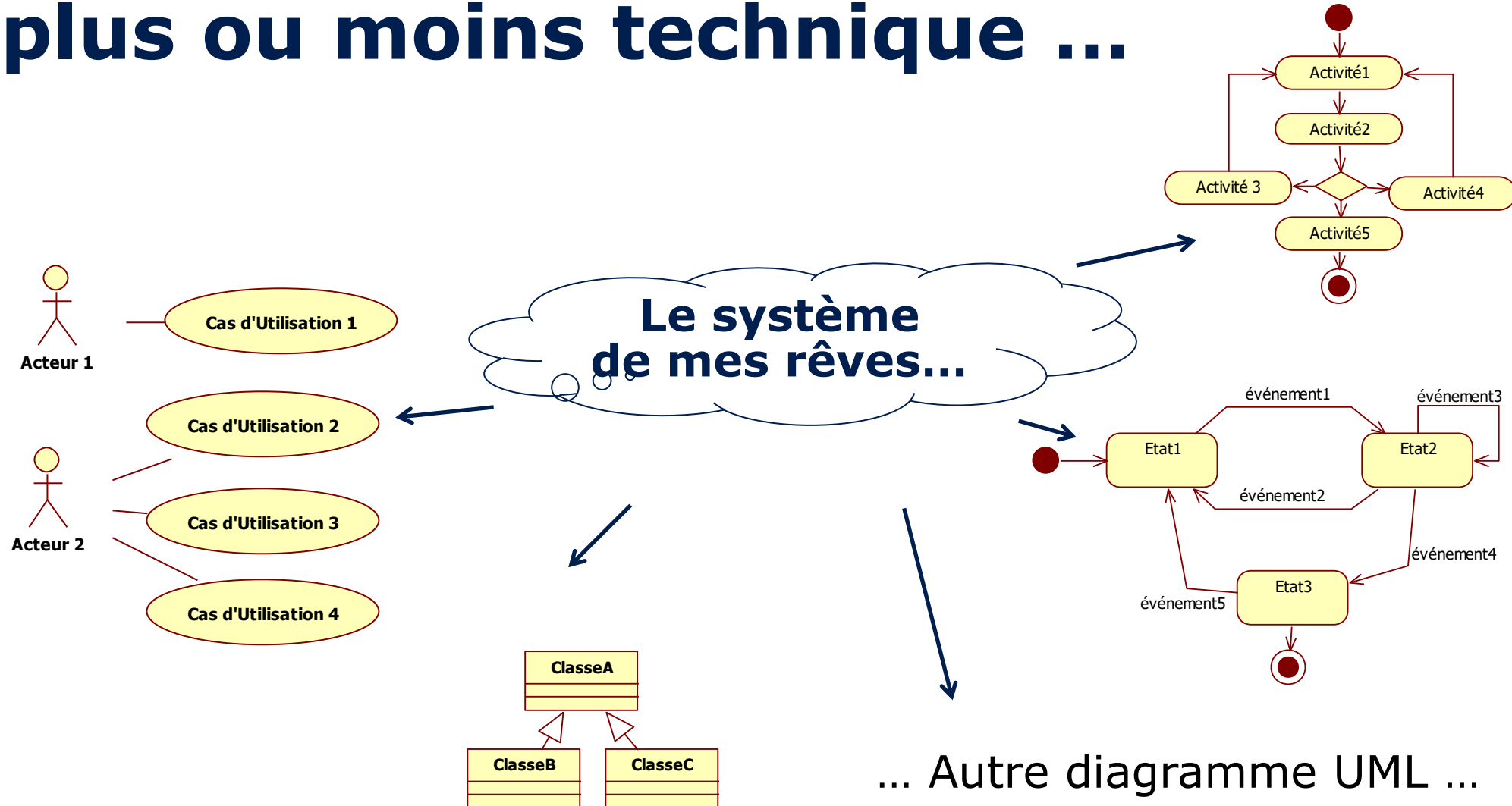
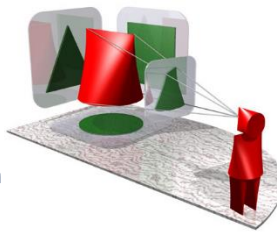


Diagramme
d'états-transitions

Exemples de modèles respectant le méta-modèle UML
(**représentation graphique** sous forme de **diagrammes**)

Chaque diagramme UML représente une VUE partielle du système, différente et plus ou moins technique ...

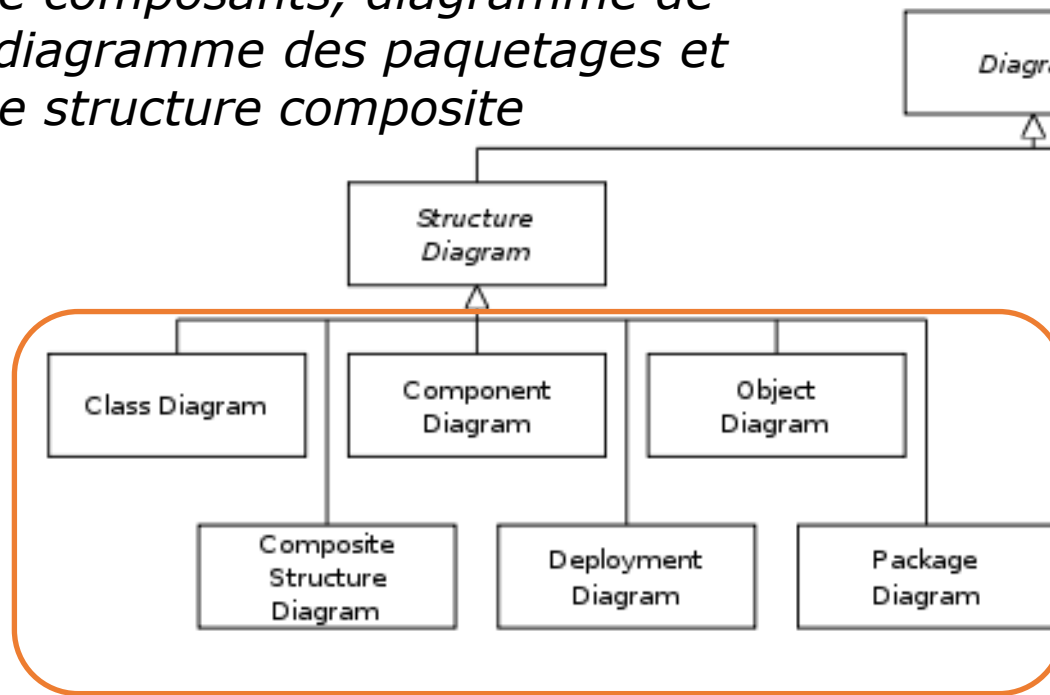


... Autre diagramme UML ...

Les 13 diagrammes d'UML 2

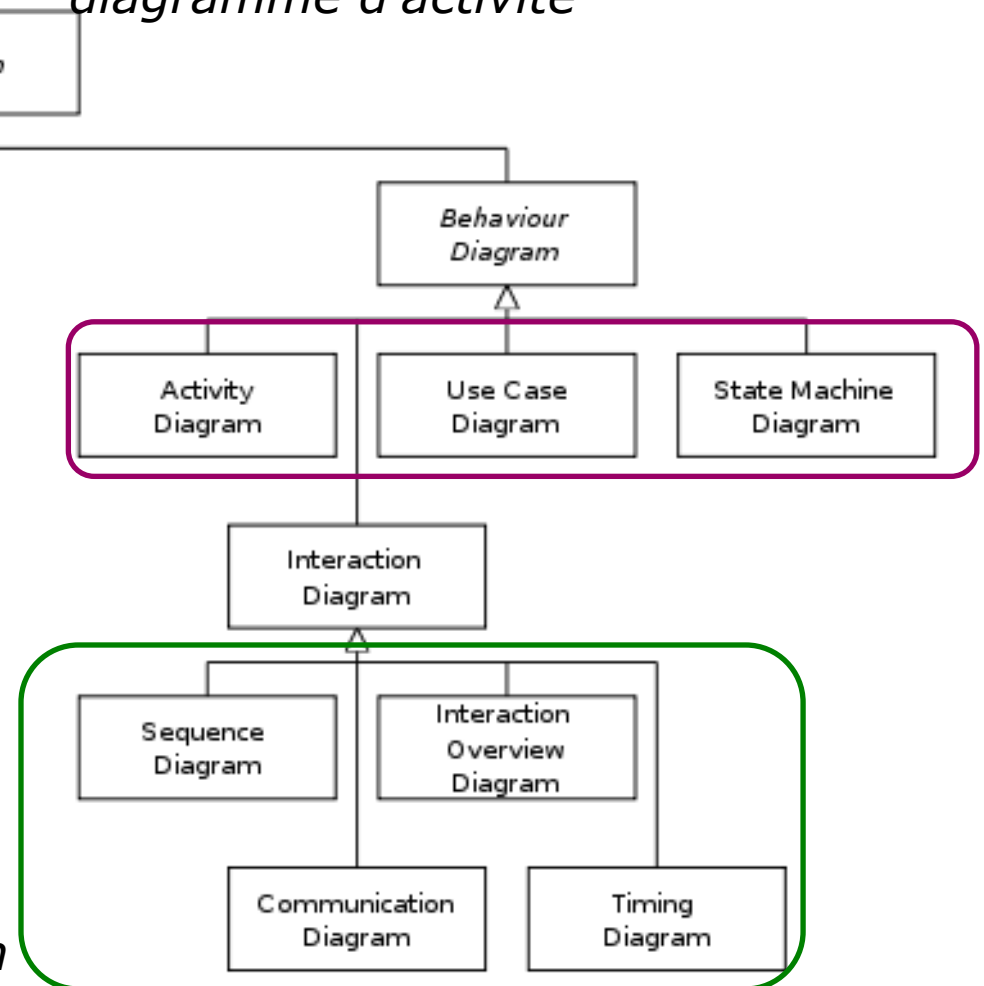
➤ Les diagrammes structurels ou statiques (Structure Diagram)

diagramme de classes, diagramme d'objets, diagramme de composants, diagramme de déploiement, diagramme des paquetages et diagramme de structure composite



➤ Les diagrammes comportementaux ou dynamiques (Behavior Diagram) :

diagramme des cas d'utilisation, diagramme états-transitions, et diagramme d'activité



➤ Les diagrammes d'interaction ou dynamiques (Interaction Diagram)

diagramme de séquence, diagramme de communication et diagramme global d'interaction

Modéliser avec UML permet de construire un **modèle quadridimensionnel** selon 4 points de vue

services rendus, architecture statique, comportement dynamique, et déploiement (installation)

QUOI ?

Aspect fonctionnel

Services rendus par le système
cas d'utilisation

QUI ?

Aspect statique

(organisation des éléments)

description des objets et de leurs relations
structuration en paquetages

OU ?

Installation du système

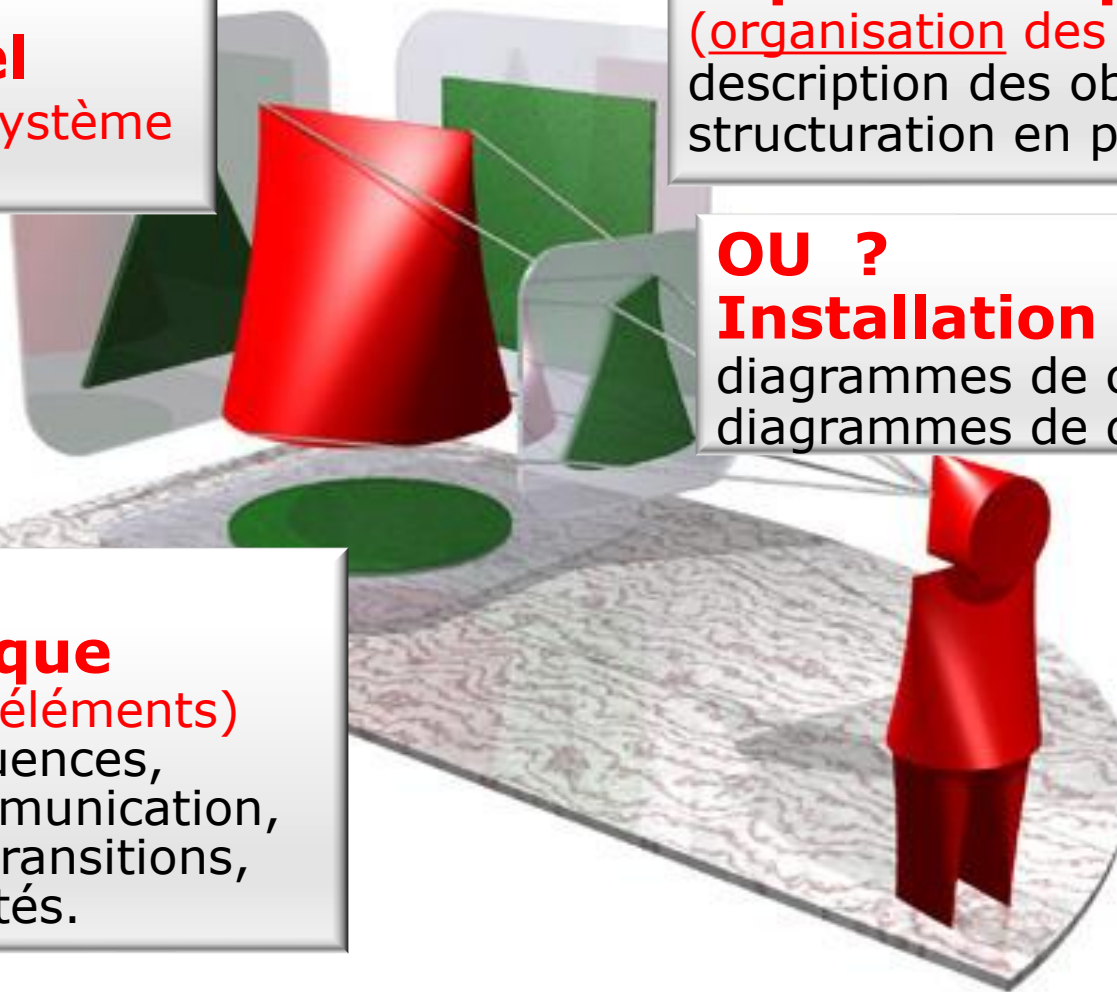
diagrammes de composants et
diagrammes de déploiement.

QUAND ?

Aspect dynamique

(comportement des éléments)

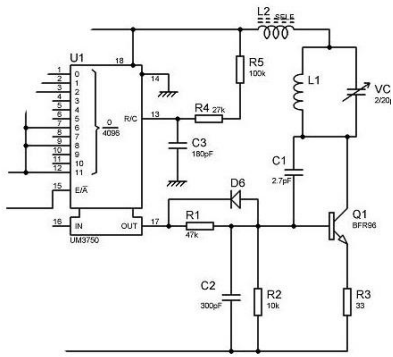
diagrammes de séquences,
diagrammes de communication,
diagramme d'états-transitions,
diagrammes d'activités.



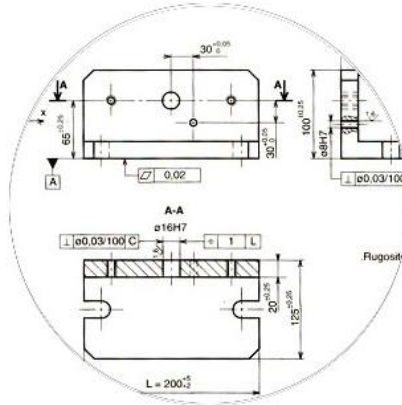
UML en pratique

UML : un support de communication ...

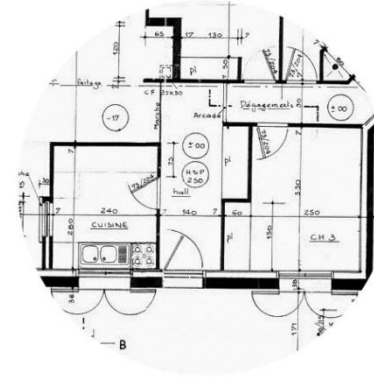
Ingénierie Electrique



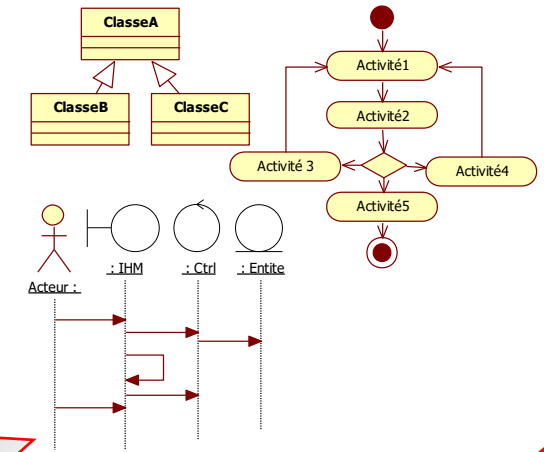
Ingénierie mécanique



Ingénierie du bâtiment



Ingénierie logicielle

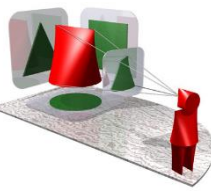


UML est un langage de modélisation graphique
avec un formalisme **orienté objet** (*méta-méta-modèle MOF*)

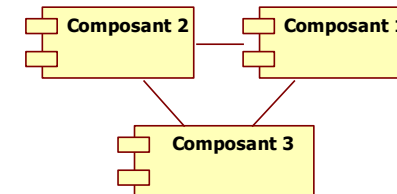
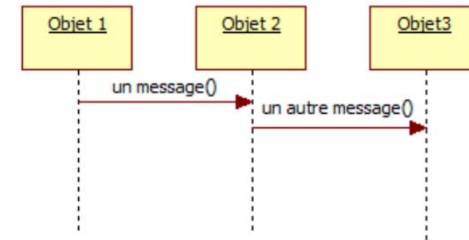
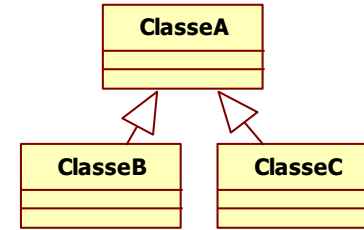
UML n'est *pas une méthode*

... et donc ne définit aucun processus de développement de produit !

Utiliser en reverse-engineering, UML permet de proposer une autre vision sur du code existant



```
function use_array(a, b) {  
  var c = -1;  
  for (var i = 0; i < a.length; i++) {  
    c.push(a[i]);  
  }  
  return c;  
}
```



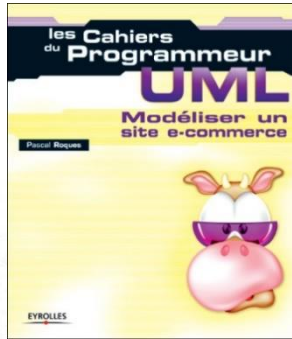
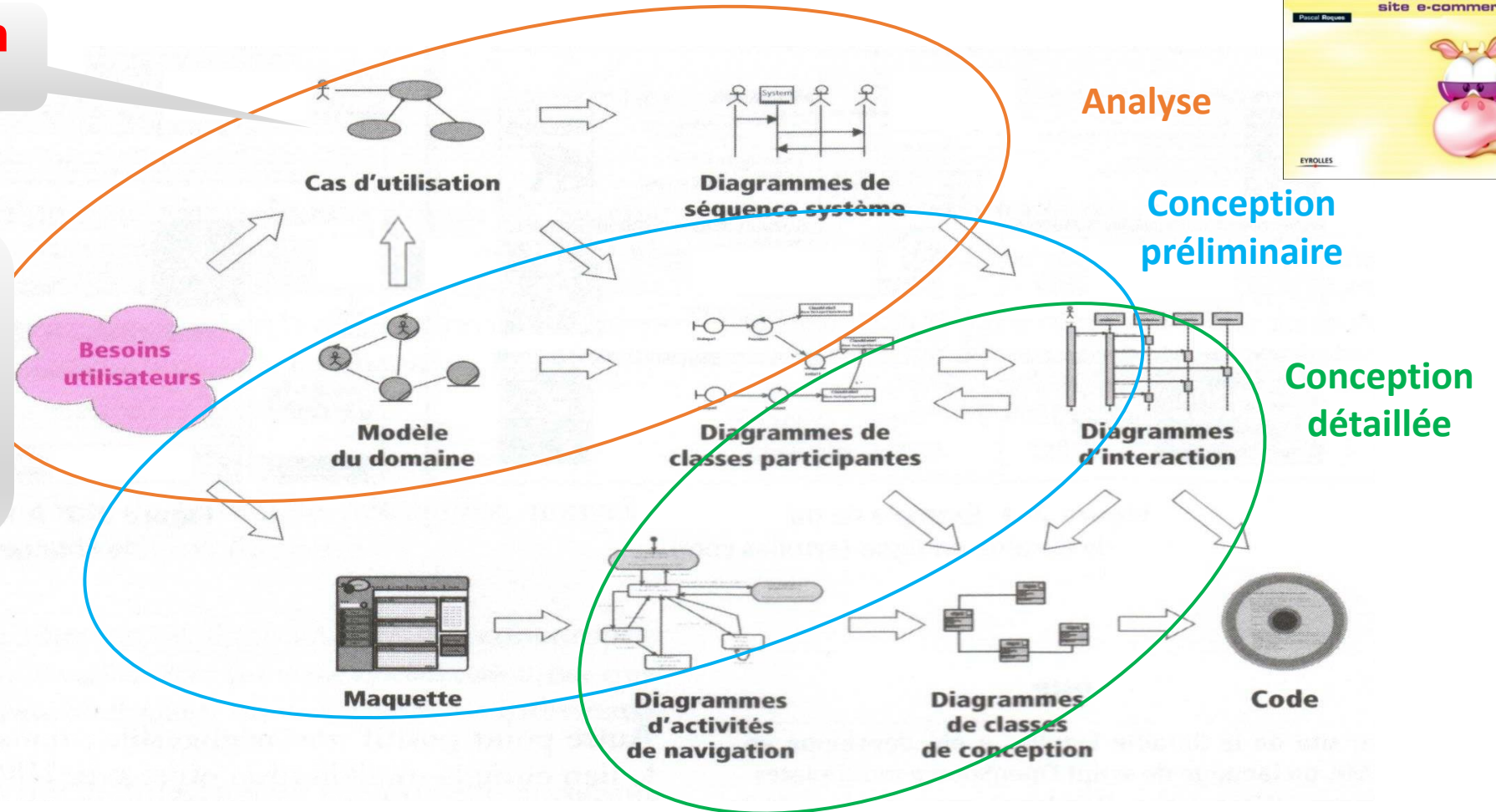
Autre diagramme ...

Une démarche possible de développement logiciel dirigée par les modèles basée sur une succession de diagrammes UML depuis l'expression des besoins (**Analyse**) jusqu'au code (**Implémentation**)

Les cas d'utilisation
pilotent la démarche

Démarche
centrée sur l'architecture
(autour de la Conception :
diagrammes de classes &
de séquences
au cœur de la démarche)

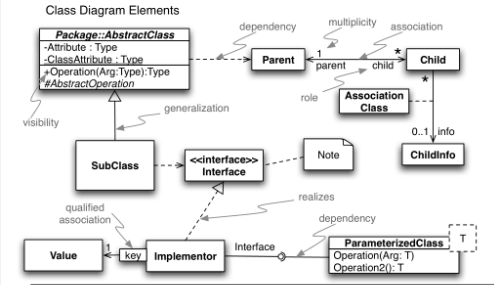
Démarche
**Itérative et
incrémentale**



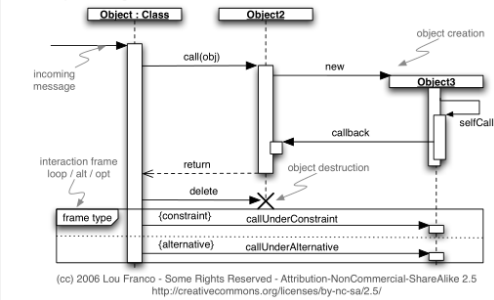
Quelques références qui pourraient vous être utile...

UML Cheatsheet

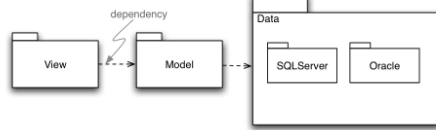
Class Diagram Elements



Sequence Diagram Elements



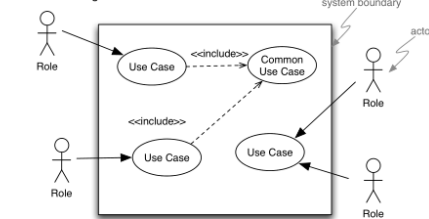
Package Diagram Elements



Object Diagram Elements



Use Case Diagram Elements



Cheatsheet sur la syntaxe UML :

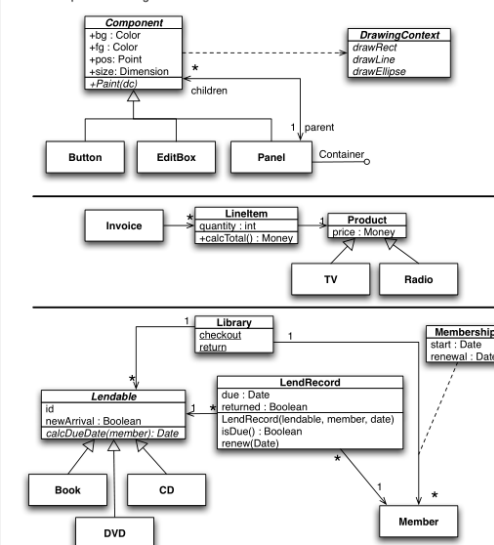
- ✓ [Cheatsheet de LouFranco](#)
- ✓ [Best UML cheatsheet and UML reference guides](#)

UML@Classroom de Springer Verlag

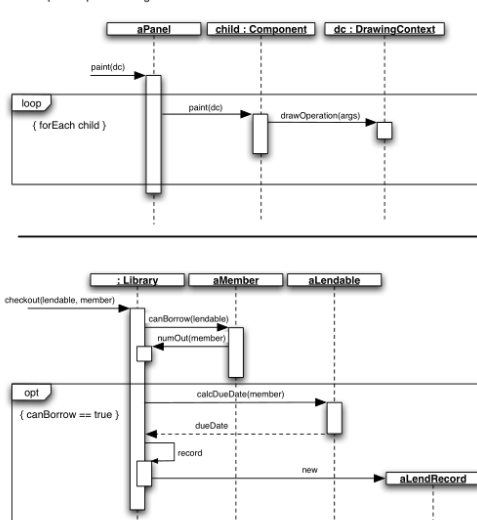
notamment la rubrique Material :

<http://www.uml.ac.at/en/lernen>

Sample Class Diagrams



Sample Sequence Diagrams




Annexe

Draw Toast Challenge : une idée de Tom Wujec

DrawToast workshops provide an effective introduction to **systems thinking** and **design collaboration**.

DrawToast INTRO TEMPLATES GALLERY WPS™ QUESTIONS




DRAW HOW TO MAKE TOAST

A simple and fun introduction to Systems Thinking

An Introduction to Systems Thinking and Wicked Problem Solving™

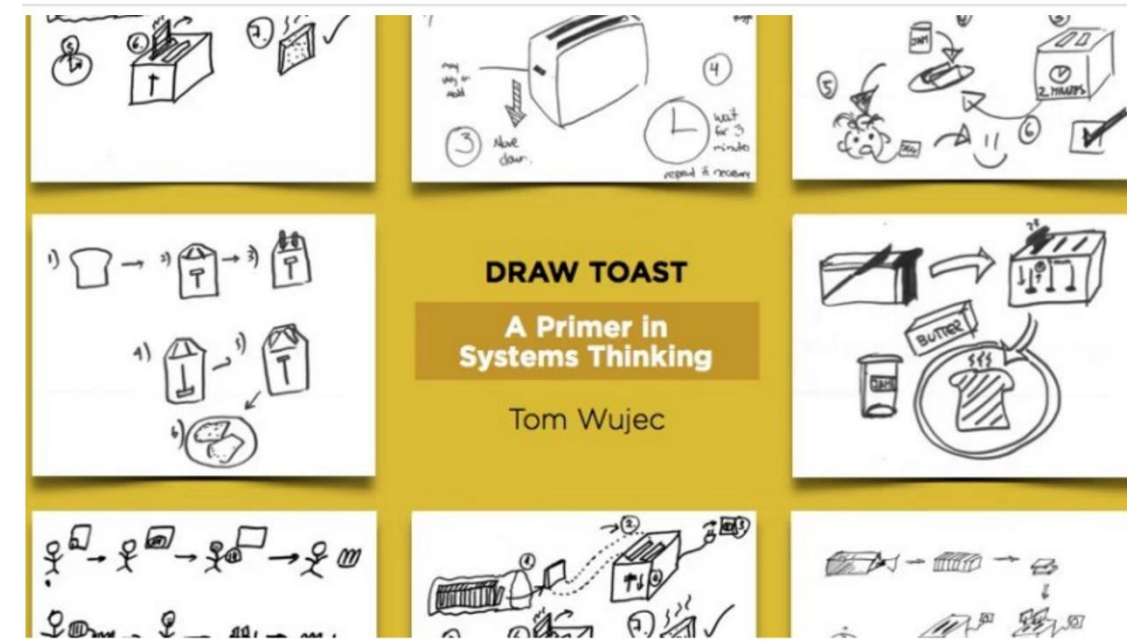
HOW TO RUN DRAW TOAST WORKSHOPS

DrawToast workshops are a great way to get groups to think freshly about mental models. In just 3 minutes, each person sketches a diagram of how to make toast. When comparing diagrams, people are shocked at how diverse the diagrams are, revealing a wide range of models of what's important in making toast. It's a great launch pad for drawing out what's really important to the group.



TOM WUJEC
GOT A WICKED PROBLEM? FIRST, TELL ME HOW YOU MAKE TOAST

TED



En savoir plus sur Draw Toast : <http://www.drawtoast.com/>
<http://www.tomwujec.com/design-projects/draw-toast/>

Et pour finir ...



What do their creators think about UML now?

By **Jordi Cabot** 5/08/2016 | 5:09

Posted in [opinion](#), [UML](#) and [OCL](#)

15



Everybody has its own opinion about the [Unified Modeling Language](#) but I think it's interesting to collect some UML opinions expressed by the people that created the language in the first place some twenty years ago.

Grady Booch's views on UML

"

The UML should be used to reason about alternatives. Put up some diagrams. Throw some use cases against it. Throw away those diagrams then write some code against you best decision. Repeat ★

"

When we began with the UML, we never intended it to become a programming language

"

We never got the notation for collaborations right. Component and deployment diagrams needed more maturing. The UML metamodel became grossly bloated, because of the drive to model driven development. I think that complexity was an unnecessary mistake.

"

I rather still like the UML 😊 Seriously, you need about 20% of the UML to do 80% of the kind of design you might want to do in a project – agile or not – but use the UML with a very light touch: use the notation to reason about a system, to communicate your intent to others...and then throw away most of your diagrams.

A lire sur : <http://modeling-languages.com/uml-opinions-creators/>