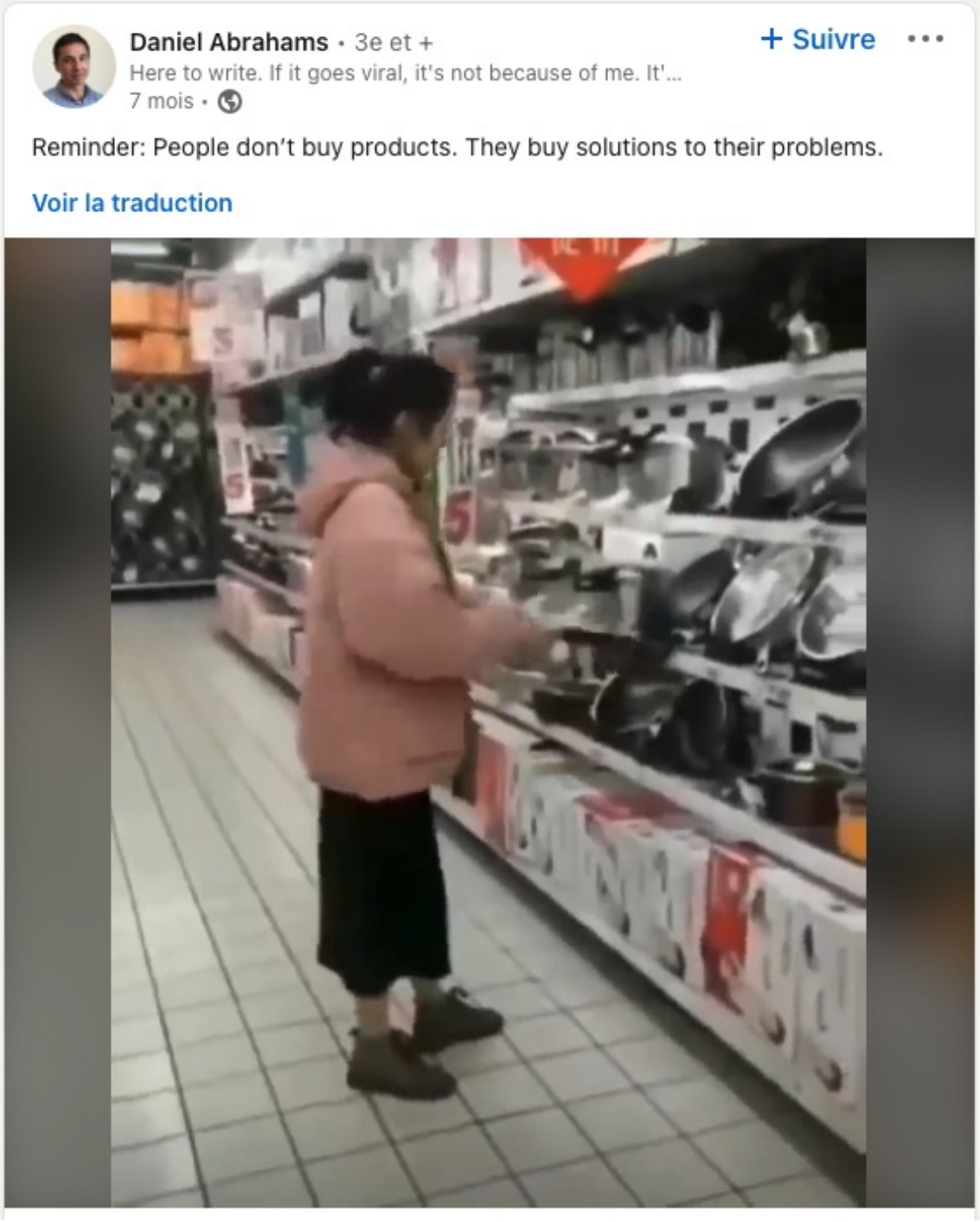
Modélisation UML
Introduction

Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA

Membre de UNIVERSITÉ CÔTE D'AZUR



https://www.linkedin.com/posts/daniel-abrahams_reminder-people-dont-buy-products-they-ugcPost-7010015948820680704-CTJD/?utm_source=share&utm_medium=member_android

People don't buy products

They buy solutions to their problem

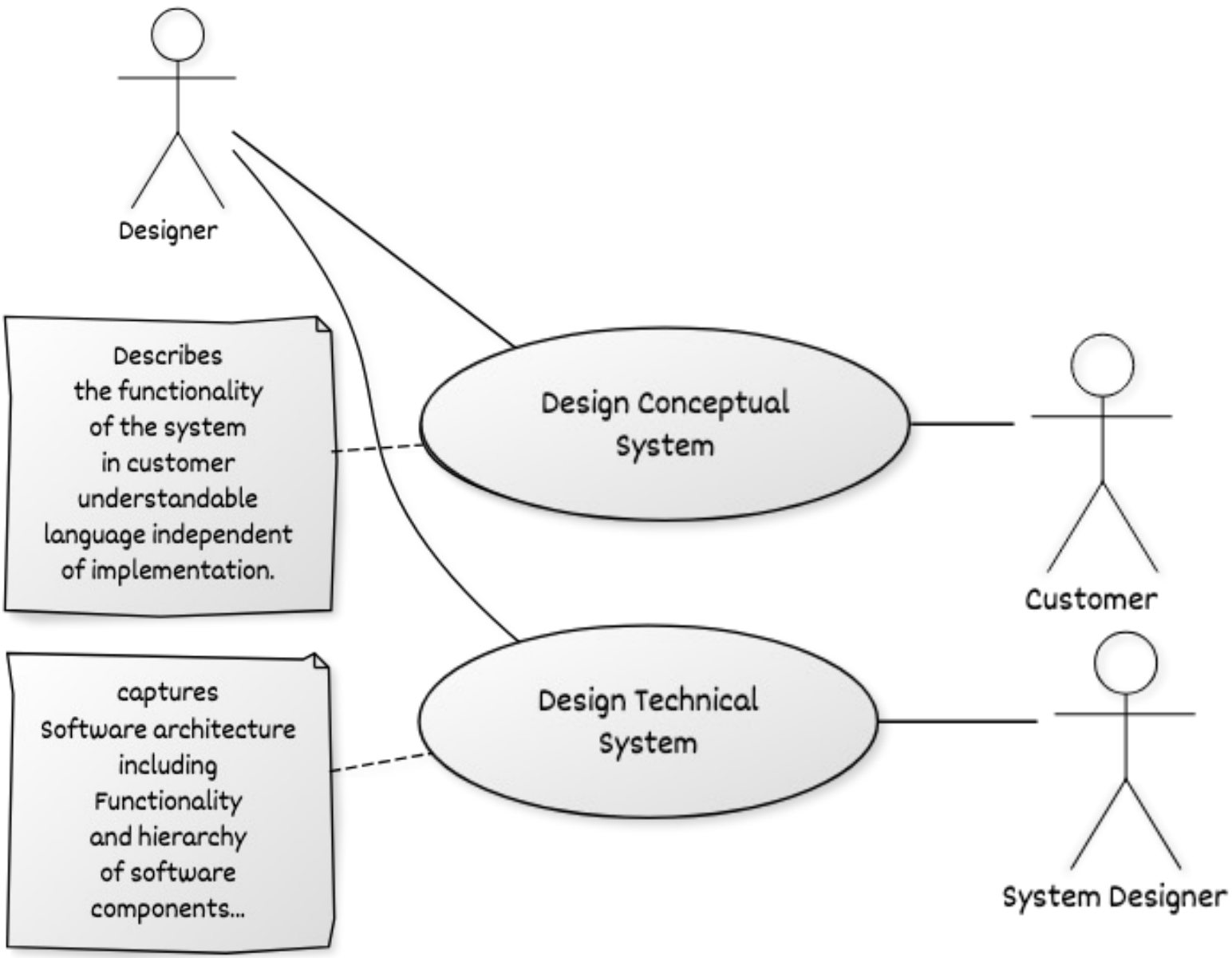
From https://jmbruel.netlify.app/

INGÉNIERIE LOGICIELLE
QUELLE PLACE À LA MODÉLISATION ?

Software Design

- A **blueprint of the solution** of the system to be developed. It is concerned
  - with the **high-level architecture** of the system,
  - as well as **the detailed design** that describes the algorithms and functionality of the individual programs.

  The detailed design is then implemented in a programming language such as C++ or Java.
- Software design is a **creative process** that is concerned with how the system will be organized and implemented.
- The choice of the architecture of the system is a key design decision, as it affects the **performance and maintainability** of the system.



https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/

Table I.1. The 15 SWEBOK KAs	
Software Requirements	★★☆
Software Design	★★★
Software Construction	★★☆
Software Testing	★☆☆
Software Maintenance	★☆☆
Software Configuration Management	
Software Engineering Management	★☆☆
Software Engineering Process	★☆☆
Software Engineering Models and Methods	
Software Quality	★☆☆
Software Engineering Professional Practice	
Software Engineering Economics	
Computing Foundations	★☆☆
Mathematical Foundations	
Engineering Foundations	☆☆☆

La place de la ***modélisation***, vision personnelle.

Bourque, P. and Fairley, R. E. (eds) (2014) *{SWEBOK}: Guide to the Software Engineering Body of Knowledge*. Version 3. Los Alamitos, CA: IEEE Computer Society. Available at: http://www.swebok.org/.

Software Design et un peu plus loin

But … when my students ask "***are we part of the solution, or part of the problem***?", I also look at … teaching topics, and I feel compelled to question the contributions of these topics to the development and impacts of the digital world as a whole. It is tempting to look at the positive impacts only. … Our research community has the responsibility to consider several hypotheses, **including one in which the digital world is part of the problem**. [2]

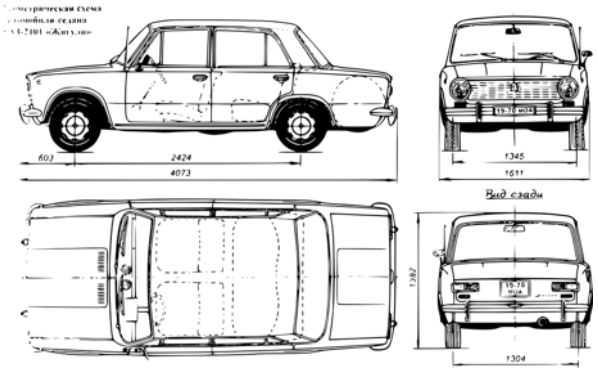
« Don't leave the problem specification task in the hands of the data scientist »[1]

[1] G. A. Lewis, S. Bellomo, and I. Ozkaya, "Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems," Mar. 2021, Accessed: Jul. 19, 2021

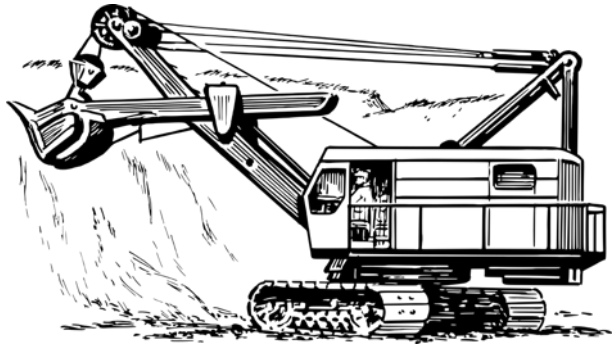
[2] Let Us Not Put All Our Eggs in One Basket, By Florence Maraninchi
Communications of the ACM, September 2022, Vol. 65 No. 9, Pages 35-37

Pourquoi modéliser ? On résume

Spécifier la structure et le comportement d'un système



Aider à la construction d'un système



Visualiser un système existant

Documenter les décisions

UML INTRODUCTION

UML un peu d'histoire

- Plein de notations et de méthodes différentes à la fin des années 90 : OMT, **Booch**, etc.

- Besoin d'un langage « standard »

- Plusieurs auteurs de méthodes orientées objets s'allient et créent l'entreprise Rational
  - Grady Booch, Ivar Jacobson, James Rumbaugh

UML, avec un U comme Unified

UML : un mélange de plusieurs notations

- Développé par Rational puis standardisé par l'OMG (Object Management Group à partir de 1997
- Tous les grandes compagnies entrent dans l'OMG : Oracle, HP, Microsoft, IBM…

Dernière version : UML 2.5.1, Dec. 2017 – 800 pages de spécification…

UML : standard industriel

- En 2005 déjà, 97% des développeurs connaissaient UML et 56% l'utilisaient dans leurs projets
  - http://www.prweb.com/releases/2005/04/prweb231386.htm

- Mais c'est quoi ? Une notation visuelle…
  - Descriptions graphiques et textuelles
  - Syntaxe et sémantique (presque pas ambiguë)
  - Architecture et comportement
  - Génération ou rétro-ingénierie de code

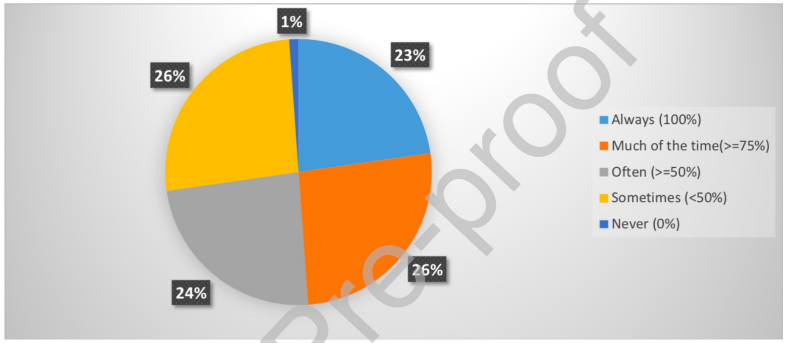


Figure 12: The frequency of the participants who model the information view(s) of software systems in UML

Ozkaya, M., & Erata, F. (2020). A survey on the practical use of UML for different software architecture viewpoints. *Information and Software Technology*, *121*, 106275.

109 different responses from 34 different countries around the world

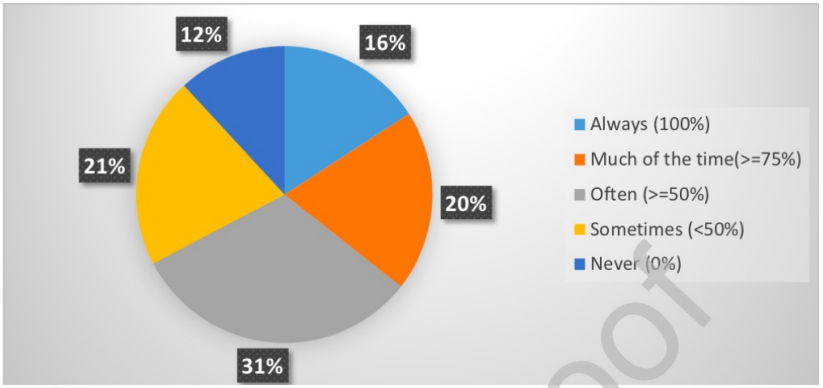


Figure 7: The frequency of the participants who use UML for modeling their software architectures from multiple viewpoints

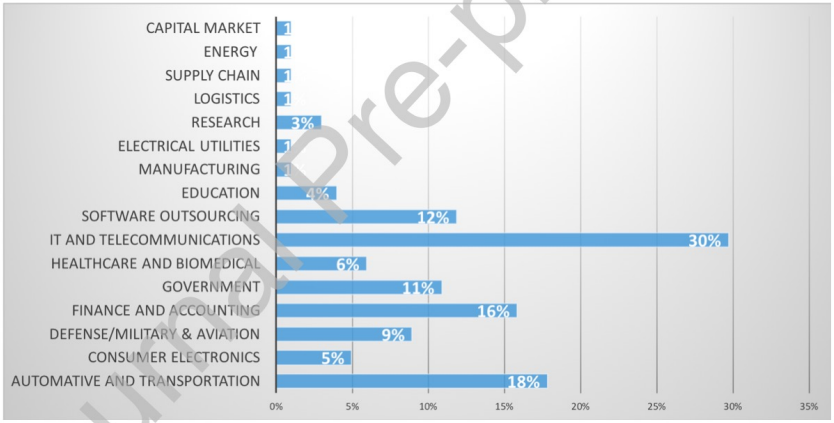


Figure 4: The work industries for the participants

Ozkaya, M., & Erata, F. (2020). A survey on the practical use of UML for different software architecture viewpoints. *Information and Software Technology*, *121*, 106275.

UML : des points forts

- Un langage normalisé
  - Précision
  - Stabilité
  - Facilite l'outillage
- Un support de communication
  - Cadrage de l'analyse
  - Compréhension d'abstractions complexes
  - Universalité

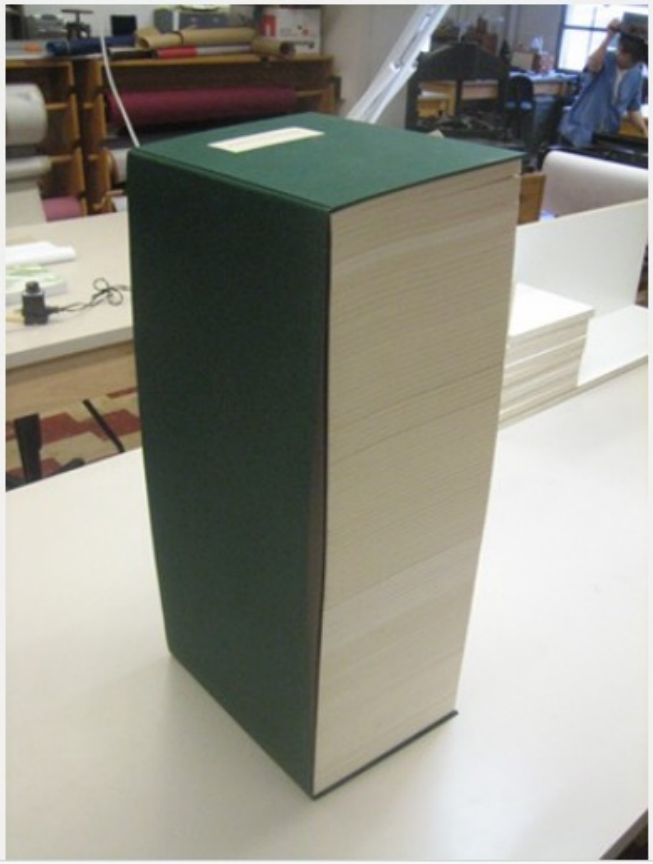
UML : des points faibles

- Période d'adaptation nécessaire
- Lourdeur de la spécification
- UML n'est qu'une notation, l'utiliser ne donne pas de « méthode » pour bien concevoir

Sébastien Mosser @petitroll · 8 min

UML v0.1 (poke @Grady_Booch)

> **Madza** @madzadev · 21h
>
> Give this book a coding-related title
>
> Afficher cette discussion



1 1

Grady Booch
@Grady_Booch

En réponse à @petitroll

You aren't wrong

Organisation d'UML

- 13 diagrammes réalisés à partir des besoins utilisateurs
- Des aspects fonctionnels
- Des aspects liés à l'architecture

Intérêts d'UML

- Beaucoup moins dans l'utilisation complète de la notation

- Beaucoup plus dans l'utilisation de certains éléments en fonction du contexte :
  - Rétro-ingénierie de code
  - Documentation
  - Zoom sur un problème, décision avant refactoring
  - Création et évolution d'architectures logicielles
  - Génération de code dans des parties système

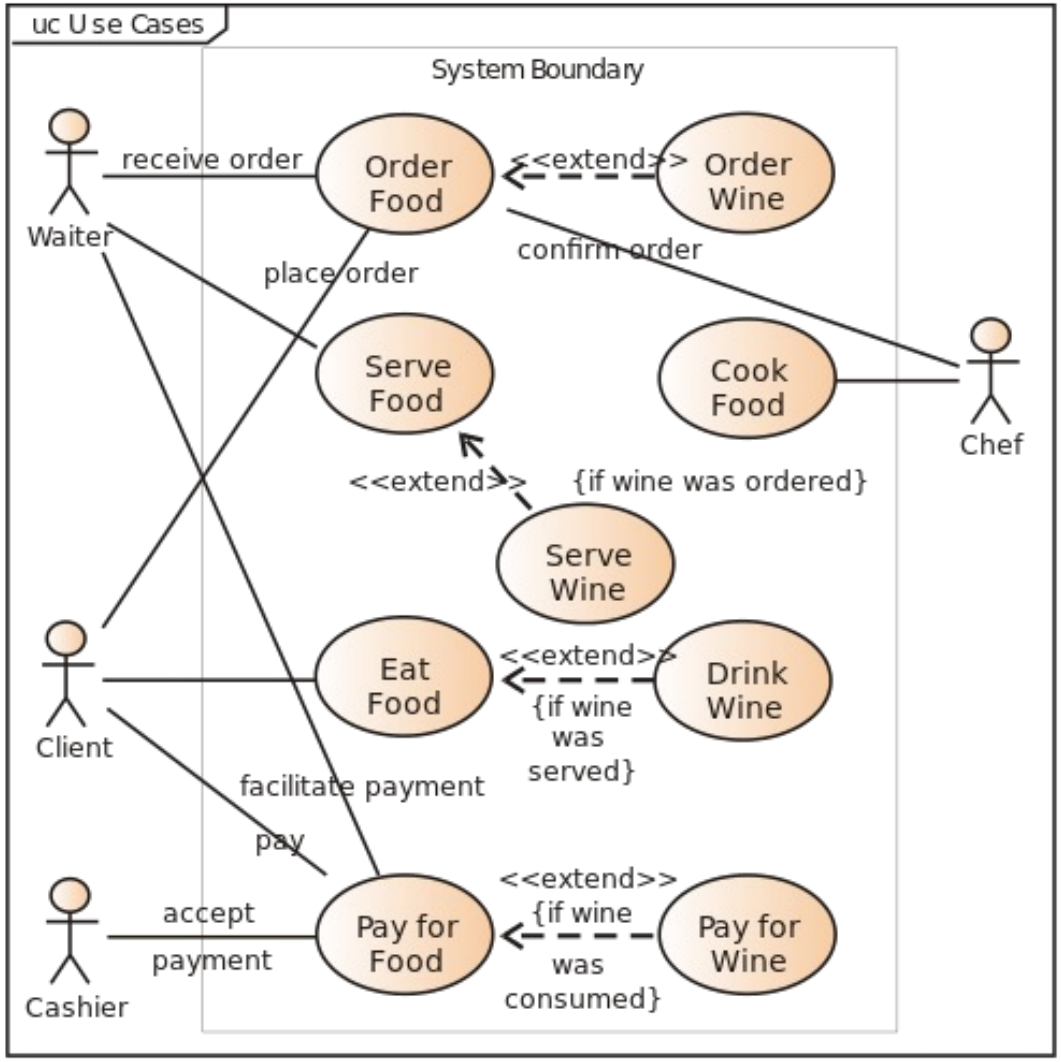
Les vues d'un système

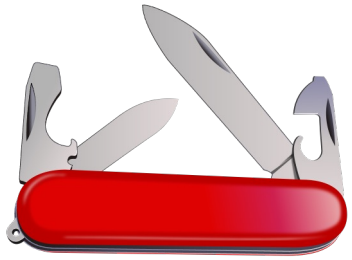
Fonctionnel = interactions utilisateurs/systèmes
- Cas d'utilisation
- Activités
- Séquence

Statique = structurelle, identifications des objets et Composants, et de leurs relations
- Classes
- Objets
- Packages
- Composants
- Déploiement

Dynamique = évolution des objets dans le temps
- Etats
- Activités
- Séquence
- Communication

Diagramme de cas d'utilisation





- Périmètre d'un système (pas forcément OO)
- Discussion utilisateur

interactions

Diagramme d'activités

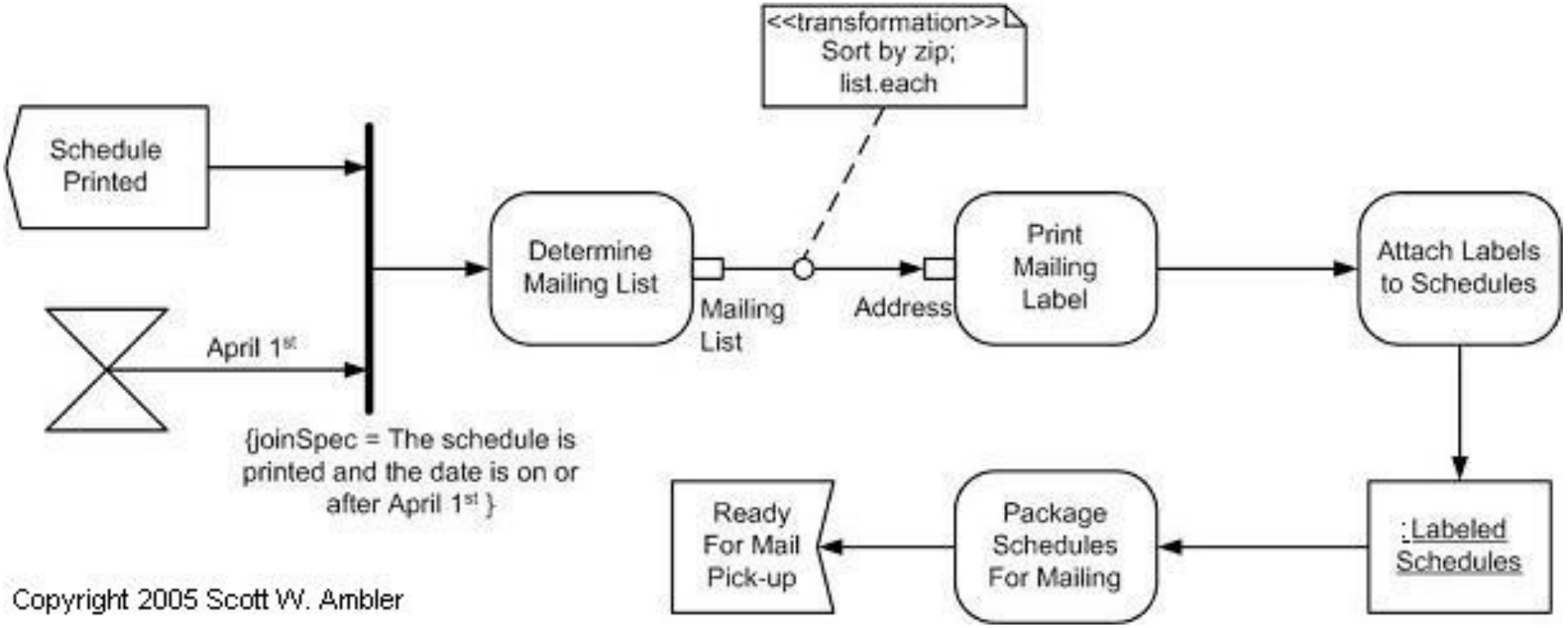


Diagramme de séquences

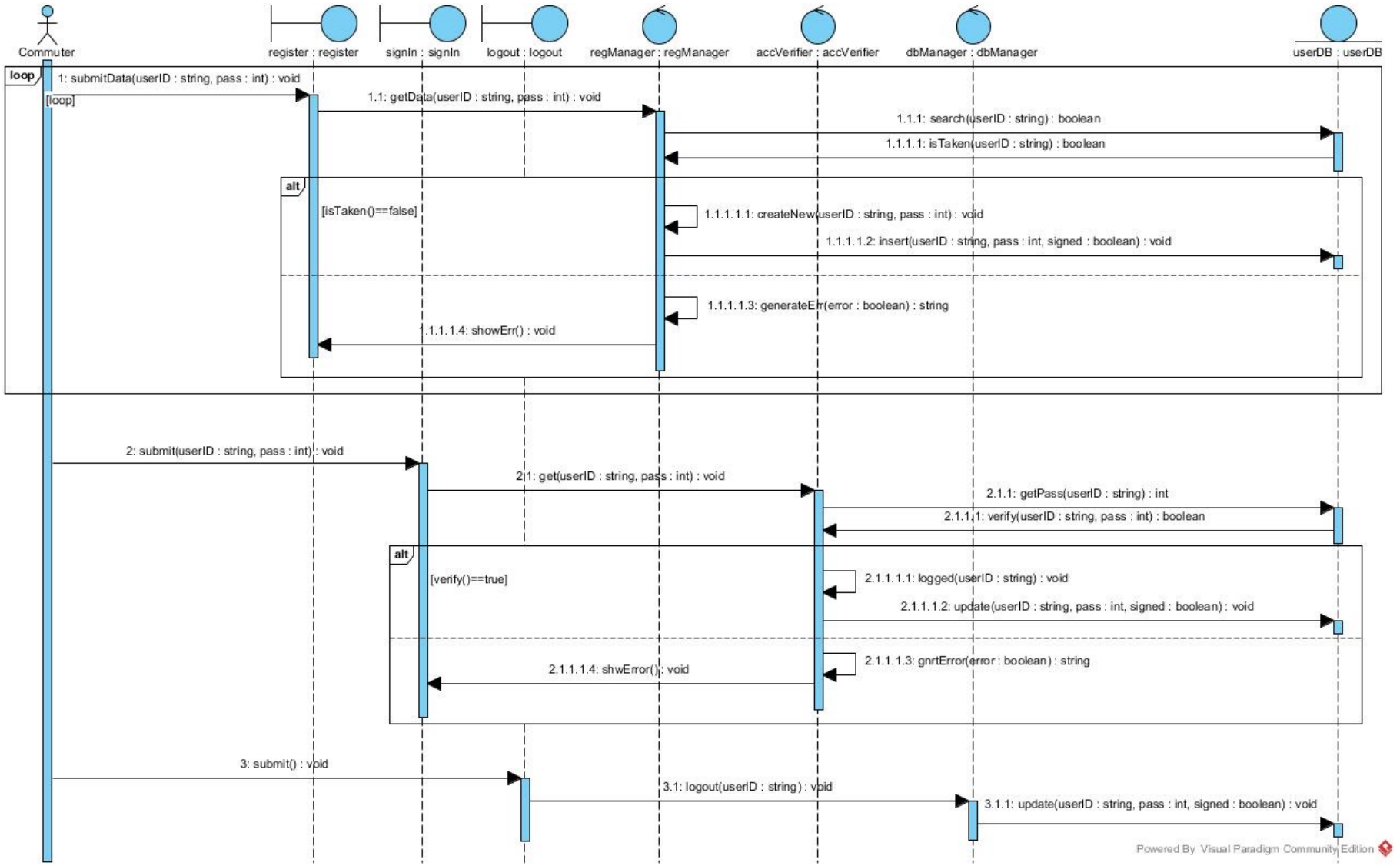
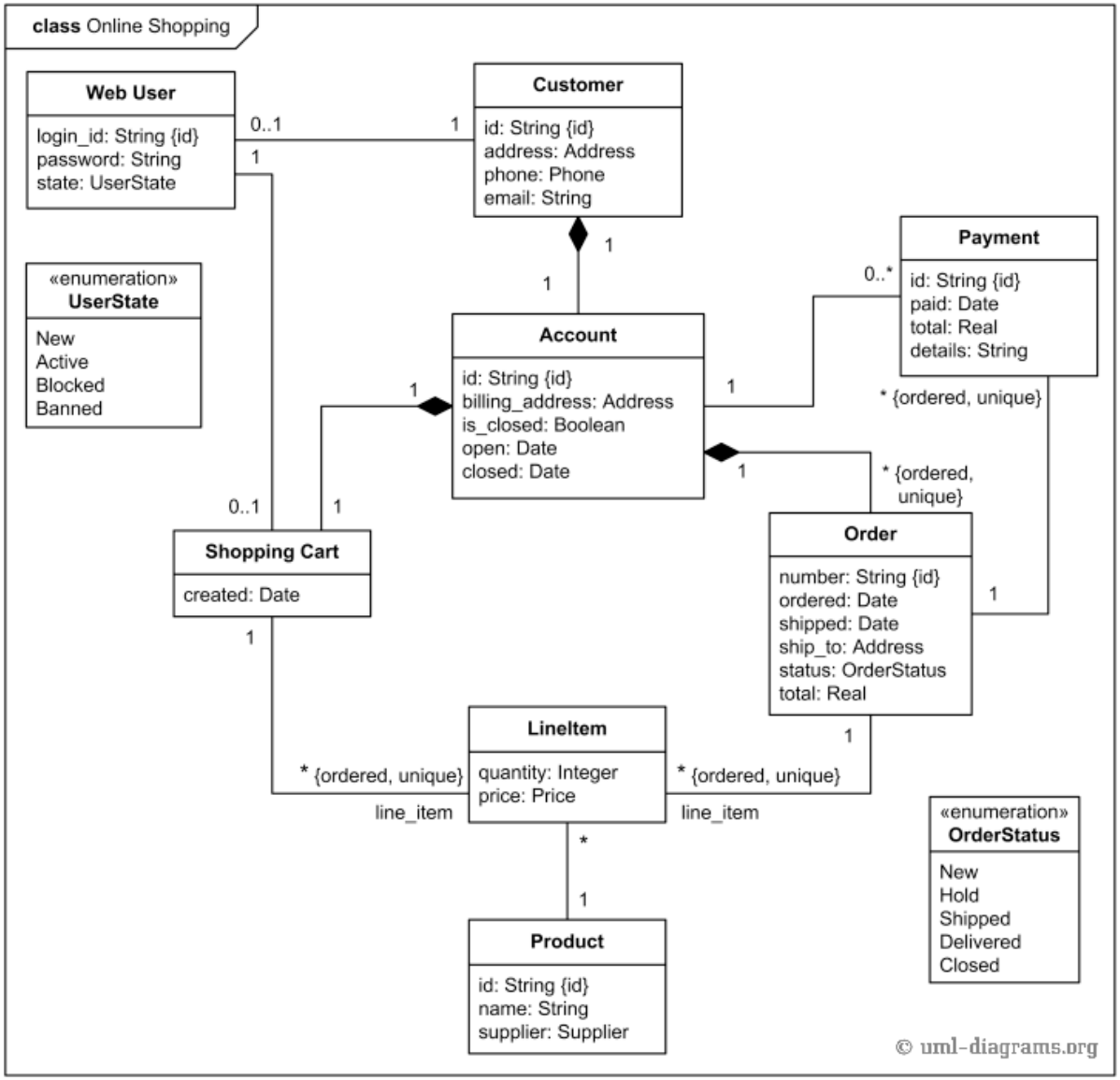


Diagramme de classes



Structure

Diagramme de packages

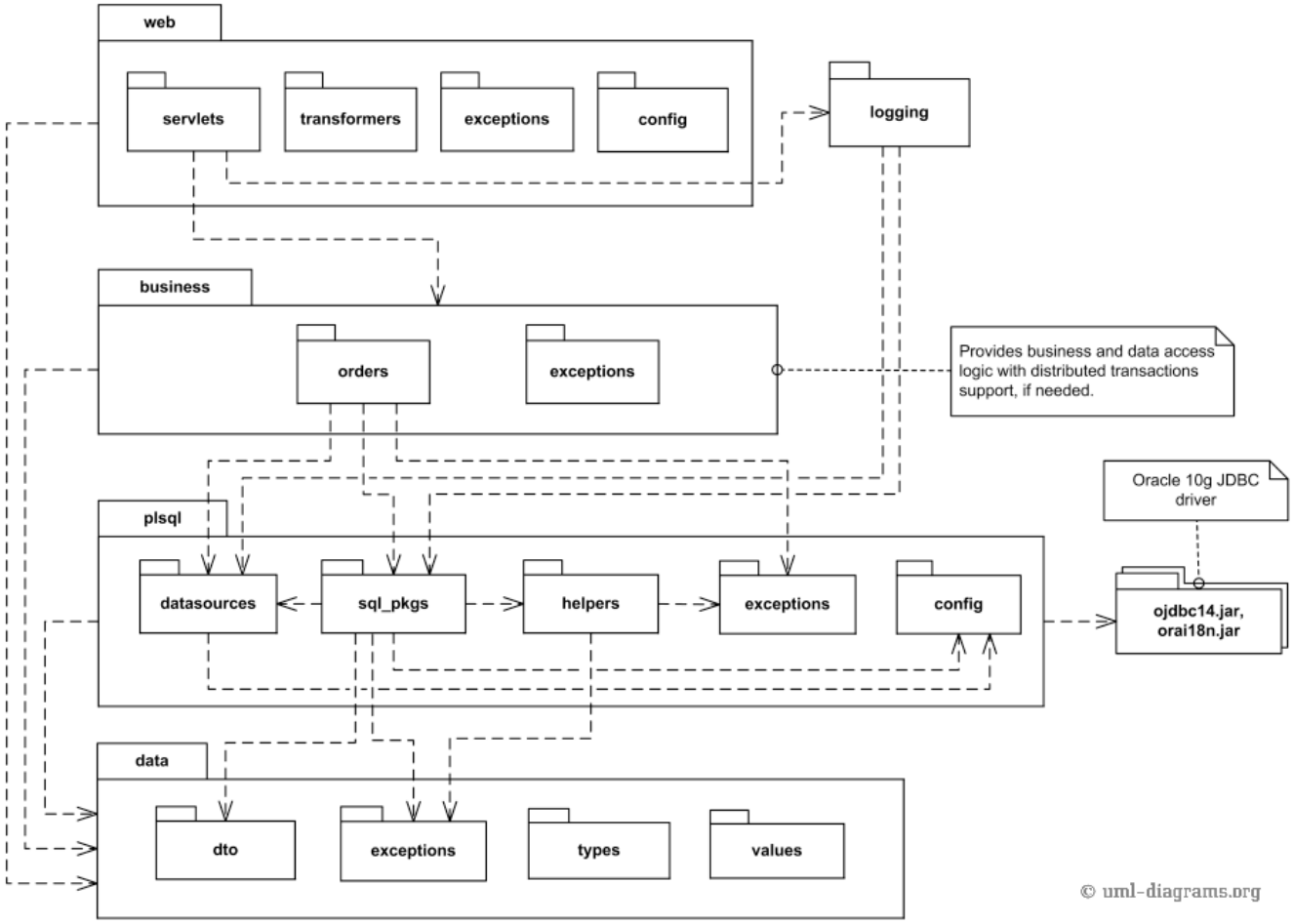
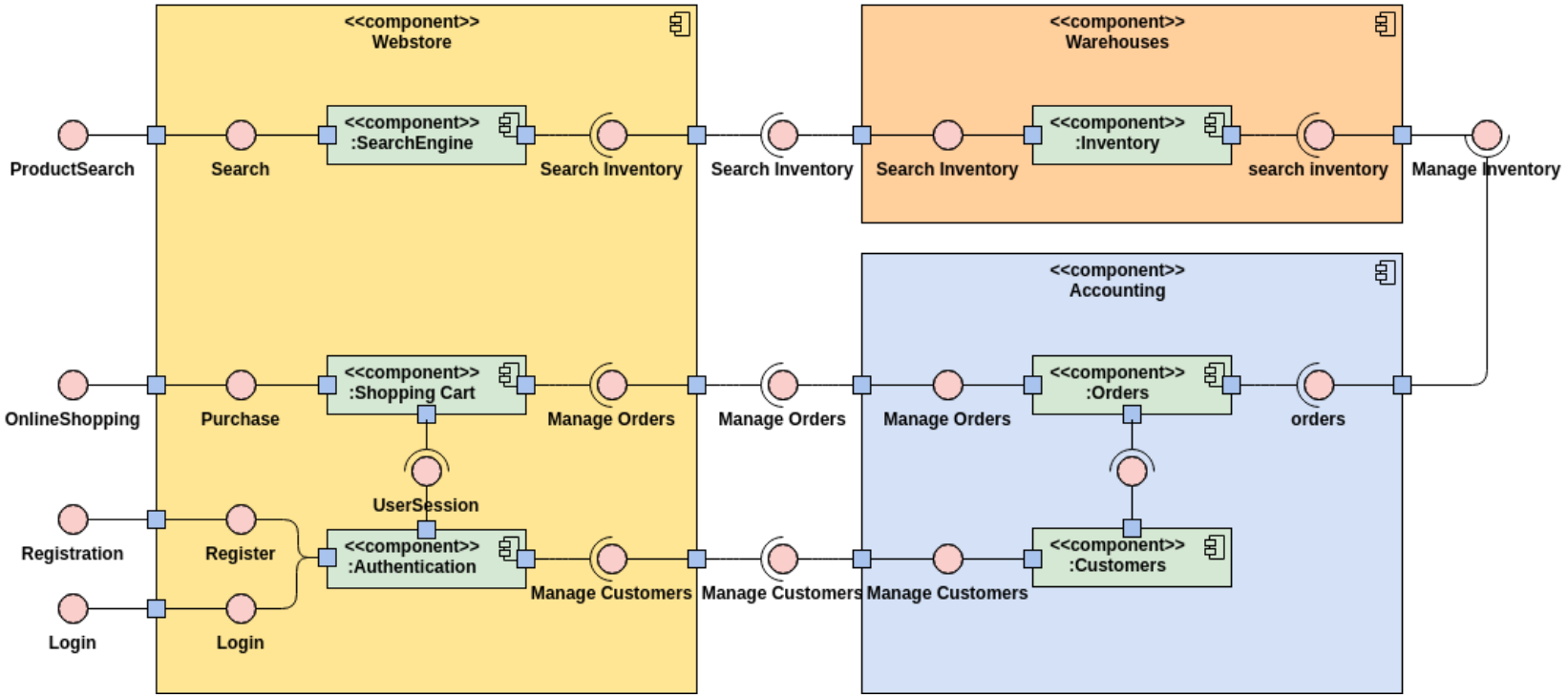
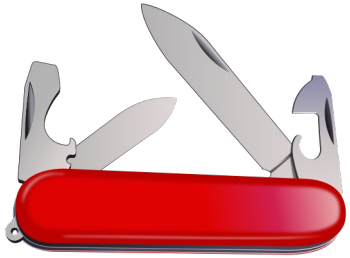


Diagramme de composants

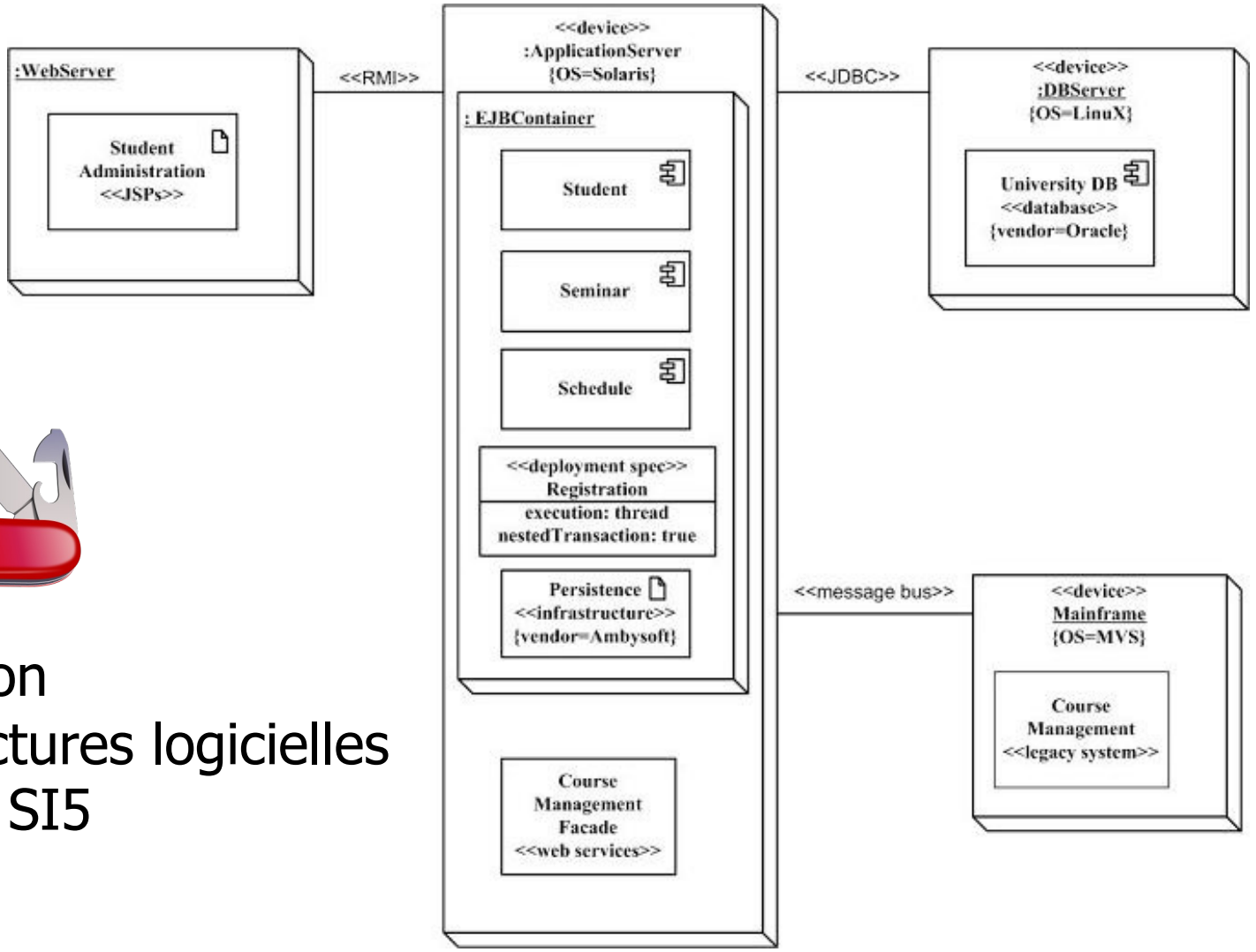


https://online.visual-paradigm.com/diagrams/templates/component-diagram/uml-component-diagram-example-web-store/



Description d'architectures logicielles -> fin de module

Diagramme de déploiement

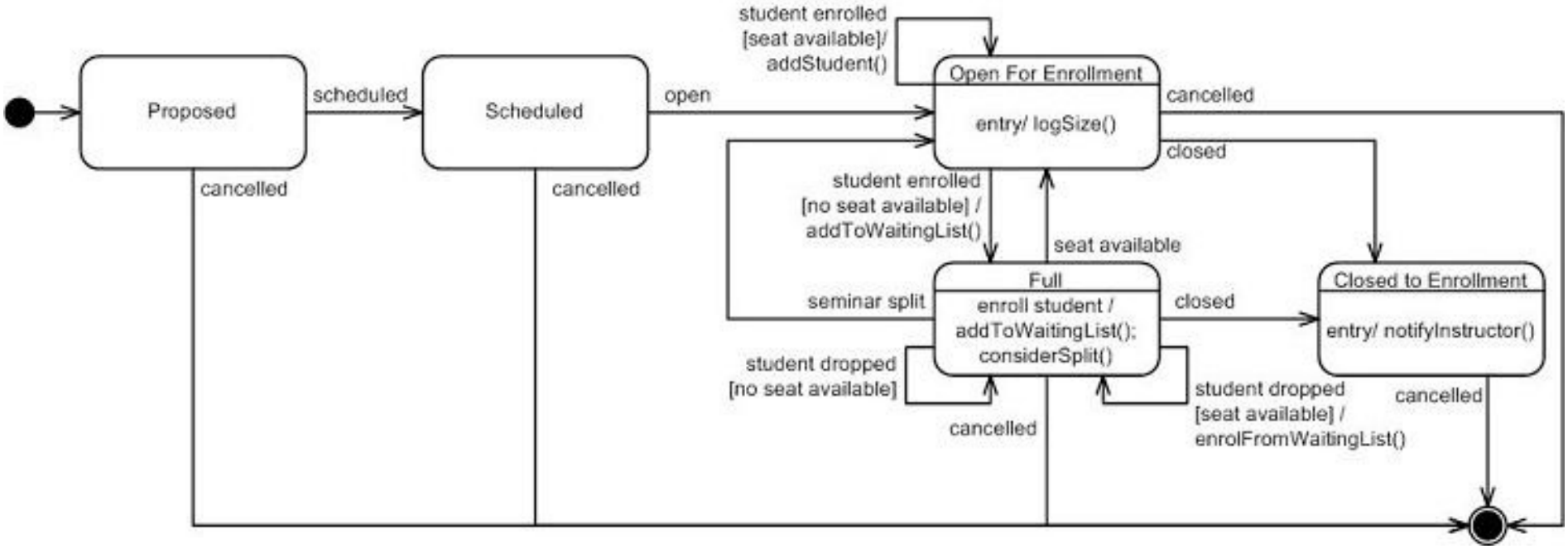


Description d'architectures logicielles -> AL en SI5

Structure

Diagramme d'états

(cf. cours Finite State Machines)



- Partout où on a besoin de spécifier une machine a états (processus métier, système « bas niveau », etc.)