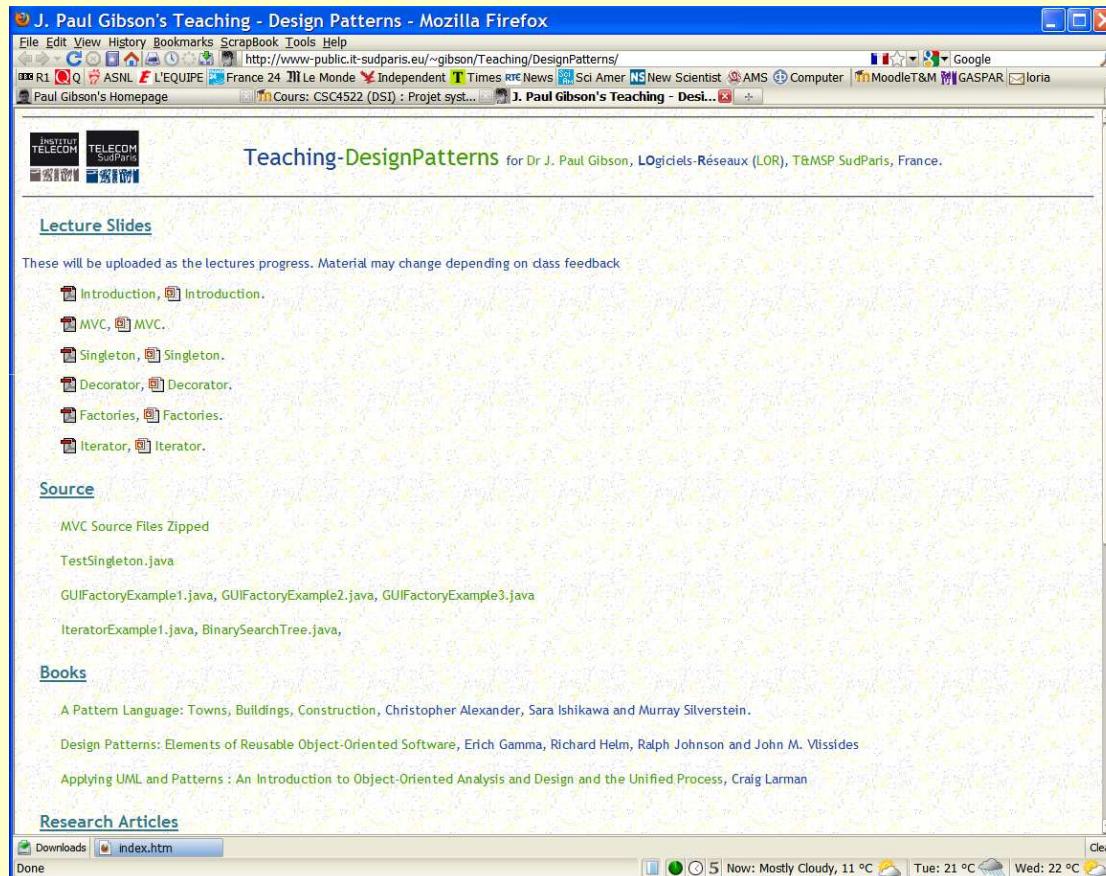


Introduction

<http://www-public.it-sudparis.eu/~gibson/Teaching/DesignPatterns/DesignPatterns-Intro.pdf>
<http://www-public.it-sudparis.eu/~gibson/Teaching/DesignPatterns/DesignPatterns-Intro.ppt>

Design Patterns - *Les patrons de conception*

<http://www-public.it-sudparis.eu/~gibson/Teaching/DesignPatterns/>



Design Patterns - *Les patrons de conception*

Un design pattern -

- *décrit une structure commune et répétitive de composants en interaction (la solution) qui résout un problème récurrent de conception dans un contexte particulier*
- *est une technique avancée de programmation (Objet)*
- *est un outil (standard) dans la **conception (orientée-objet)***

Design Patterns - *Les patrons de conception*

La Conception (Orientée-Objet)

- Un art difficile...
- Une conception réutilisable, extensible, adaptable, performante, ... est extrêmement difficile
- Novice v concepteur expérimenté, typiquement:
 1. le novice hésite beaucoup entre différentes variantes
 2. l'expert trouve « tout de suite » la bonne solution
- Pourquoi la différence ?

l'expérience

Design Patterns - *Les patrons de conception*

Expertise en Conception (Orientée-Objet)

- ne pas réinventer la roue
- réutiliser systématiquement des solutions qui ont fait leurs preuves
- pour une bonne conception (modulaire, élégante, adaptable ...)

... mais comment ???...

répétition de certains profils de classes ou collaboration d'objets: *design patterns, modèles de conceptions, patrons de conception, micro-architectures*

Design Patterns - *Les patrons de conception*

Origines

1970s - Travaux de l'architecte Christopher Alexander: a perfectionné la théorie des « *Pattern languages* » utilisée dans plusieurs domaines –

de l'anthropologie et de l'histoire de l'art, puis dans celui du design (les types architecturaux), ensuite en informatique

1980s –

Exemple « historique » du modèle MVC (**Model-View-Controller**) de Smalltalk (Trygve Reenskaug [Xerox Parc] & Krasner, Pope)

Patterns et OOP (Beck et Cunningham)

1990s - Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides - **Gang of Four** (GoF) *Design Patterns*

Design Patterns - *Les patrons de conception*

Plus récemment (depuis 2000)

UML and patterns

Web 2.0: Design Patterns and Business Models

Agile software development: principles, patterns, and practices

Design Patterns and Aspects

Design Patterns for Concurrent programming

Design Patterns from biology

Composing Design Patterns

Design Patterns for Services and Web Applications

Formal Specification of Design Patterns

Software factories/product lines with patterns ...

Design Patterns - *Les patrons de conception*

References: Google scholar

[BOOK] [A pattern language: towns, buildings, construction](#)

C Alexander, S Ishikawa, M Silverstein - 1977 - [books.google.com](#)

... In this sense, in a healthy society there will be as many **pattern languages** as there are people—even though these **languages** are shared and similar. ...

[Cited by 1436](#) - [Related articles](#) - [Web Search](#) - [SUDOC Catalogue](#) - [All 2 versions](#)

[CITATION] **Using pattern languages for object-oriented programs**

K Beck, W Cunningham - ... and Design for **Object-Oriented** Programming (OOPSLA-87), 1987

[Cited by 101](#) - [Related articles](#) - [Web Search](#)

[A cookbook for using the model-view controller user interface paradigm in Smalltalk-80](#)

GE Krasner, ST Pope - Journal of Object-oriented programming, 1988 - [portal.acm.org](#)

... Search: The ACM Digital Library The Guide. Feedback. A cookbook for using the **model-view controller** user interface paradigm in Smalltalk-80. ...

[Cited by 1227](#) - [Related articles](#) - [Web Search](#)

[Design patterns: Abstraction and reuse of object-oriented design](#) - ► [kfupm.edu.sa](#) [PDF]

E Gamma, R Helm, R Johnson, J Vlissides - ECOOP'93, Object-oriented Programming, 7th European ..., 1993 - [books.google.com](#)

... Vlissides. A catalog of object-oriented **design patterns**. Technical Report in preparation, IBM Research Division, 1992. Page 434. 420 13. ...

[Cited by 401](#) - [Related articles](#) - [Web Search](#) - [BL Direct](#) - [All 13 versions](#)

Design Patterns - *Les patrons de conception*

References: Google scholar

[BOOK] [Design patterns: elements of reusable object-oriented software](#)

E Gamma, R Helm, R Johnson, J Vlissides - 1995 - [portal.acm.org](#)

Google, Inc. (search), Subscribe (Full Service), Register (Limited Service, Free), Login. Search: The ACM Digital Library The Guide. Feedback. Design patterns: elements of reusable object-oriented software. Purchase this Book ...

[Cited by 17237](#) - [Related articles](#) - [Web Search](#) - [SUDOC Catalogue](#)

[Design patterns for object-oriented software development \(tutorial\)](#) - ► [cu.ac.kr](#) [PDF]

W Pree, H Sikora - [Proceedings of the 19th international conference on Software ...](#), 1997 - [portal.acm.org](#)

Page 1. **Design Patterns** for Object-Oriented Software Development ... Case studies based on Java illustrate how to apply **design patterns**. ...

[Cited by 898](#) - [Related articles](#) - [Web Search](#) - [SUDOC Catalogue](#) - [BL Direct](#) - [All 11 versions](#)

[BOOK] Concurrent Programming in Java.: **Design Principles and Patterns** - ► [oswego.edu](#) [PDF]

D Lea - 1999 - Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

0201310090 - Concurrent Programming in Java™: Design Principles and Patterns, Second Edition - In this second edition, you will find thoroughly updated coverage ...

[Cited by 1023](#) - [Related articles](#) - [Web Search](#) - [SUDOC Catalogue](#) - [All 18 versions](#)

[BOOK] Applying UML and **patterns**: an introduction to object-oriented analysis and **design** and the unified ...

C Larman - 2001 - Prentice Hall PTR Upper Saddle River, NJ, USA

0130925691 - Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process, Second Edition - "People often ask me which ...

[Cited by 815](#) - [Related articles](#) - [Web Search](#) - [SUDOC Catalogue](#) - [All 7 versions](#)

Alexander :

As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves.

As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant.

Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.

Prece and Sikora :

Design Patterns describe [software] framework construction on a level higher than the underlying programming language.

Because patterns have become the vogue in the software engineering community, the term now is used wherever possible, adorning even project management or organizational work. So the genericity of the term pattern might be the reason that patterns are found everywhere, a fact which is regarded as a clear indication of a hype.

Erich Gamma (GoF):

Patterns provide you with tools that help you with design problems. They do so not by giving a pat solution but by explaining trade-offs. Even though patterns are abstracted from concrete uses, they also provide you valuable implementation hints. From my perspective it is the fact that patterns are implementable that makes them so valuable.

Patterns are distilled from the experiences of experts. They enable you to repeat a successful design done by someone else. However, since patterns enable many implementation variations you still have to keep the brain turned on.

Since patterns provide you with names for design building blocks they provide you with a vocabulary to describe and discuss a particular design.

I think patterns as a whole can help people learn object-oriented thinking: how you can leverage polymorphism, design for composition, delegation, balance responsibilities, and provide pluggable behavior.

Wikipedia: http://fr.wikipedia.org/wiki/Patron_de_conception

« Les patrons de conception décrivent des solutions standard pour répondre à des problèmes d'architecture et de conception des logiciels. À la différence d'un algorithme qui s'attache à décrire d'une manière formelle comment résoudre un problème particulier, les patrons de conception décrivent des procédés de conception généraux. On peut considérer un patron de conception comme une formalisation de bonnes pratiques, ce qui signifie qu'on privilégie les solutions éprouvées.

Il ne s'agit pas de fragments de code, puisque les patrons de conception sont le plus souvent indépendants du langage de programmation, mais d'une méthode de conception, c'est-à-dire d'une manière standardisée de résoudre un problème qui s'est déjà posé par le passé. Le concept de patron de conception a donc une grande influence sur l'architecture logicielle d'un système.

On peut donc considérer les patrons de conception comme un outil de capitalisation de l'expérience appliqué à la conception logicielle. »

Design Patterns - *Les patrons de conception*

Les éléments d'un patron de conception

1. Nom:

identification d'un concept +...

2. Problème:

situations dans lesquelles le patron s'applique +...

3. Solution:

éléments du modèle de conception +...

4. Conséquences:

effets de l'application du modèle sur la conception +...

Design Patterns - *Les patrons de conception*

Description d'un patron de conception (1) Nom

- ▷ Nom et classification
 - un vocabulaire de référence
- ▷ Intention
 - ◇ but, raison d'être
 - ◇ cas ou problèmes particuliers de conception concernés
- ▷ Alias
 - autres noms (re)connus

Design Patterns - *Les patrons de conception*

Description d'un patron de conception (2) Problème

▷ Motivation

scénario qui illustre un cas de conception et montre comment le modèle contribue à la solution.

▷ Indications d'utilisation

- ◇ cas qui justifient son utilisation,
- ◇ situations peu satisfaisantes qui tirent avantage de son utilisation,
- ◇ contextes d'utilisation.

▷ Structure

- ◇ modèle de classe,
- ◇ (modèle de collaboration).

(notation OMT à l'origine, UML aujourd'hui)

Design Patterns - *Les patrons de conception*

Description d'un patron de conception (3) Solution

▷ Constituants

Description des classes et des objets intervenant dans la collaboration (rôles)

▷ Collaboration

Collaboration entre constituants pour assurer les responsabilités

▷ Conséquences

- ◇ impacts sur l'architecture de conception
- ◇ «degrés de liberté» laissés par le patron
- ◇ compromis éventuels

Design Patterns - *Les patrons de conception*

Description d'un patron de conception (4) Conséquences

▷ Implémentation

- ◇ techniques particulières
- ◇ spécificités du langage de programmation

▷ Exemples de code

- ◇ extraits de programmes

▷ Utilisations remarquables

- ◇ exemples appartenant à des systèmes existants

▷ Modèles apparentés

- ◇ modèles en étroite relation
- ◇ différences importantes
- ◇ utilisation conjointe d'autres modèles

Design Patterns - *Les patrons de conception*

Le catalogue GoF patterns (23)

Création

- * Fabrique abstraite (Abstract Factory) * Monteur (Builder)
- * Fabrique (Factory Method) * Prototype (Prototype)
- * Singleton (Singleton)

Structure

- * Adaptateur (Adapter) * Pont (Bridge) * Objet composite (Composite)
- * Décorateur (Decorator) * Façade (Facade)
- * Poids-mouche ou poids-plume (Flyweight) * Proxy (Proxy)

Comportement

- * Chaîne de responsabilité (Chain of responsibility)
- * Commande (Command) * Interpréteur (Interpreter) * Itérateur (Iterator)
- * Médiateur (Mediator) * Memento (Memento) * Observateur (Observer)
- * État (State) * Stratégie (Strategy)
- * Patron de méthode (Template Method) * Visiteur (Visitor)

Design Patterns - *Les patrons de conception*

Pattern Composition

Exemple: Le patron **Modèle-Vue-Contrôleur** (MVC) est une « combinaison » des patrons *Observateur*, *Stratégie* et *Composite*

The best way to learn about patterns is to look at examples. We shall do these in Java, but any (OO) language can be used to implement/re-use a pattern