N° d'ordre :
N° de série :

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

---

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED
FACULTÉ DES SCIENCES EXACTES
Département D'Informatique

Mémoire de Fin D'étude
Présenté pour l'obtention du Diplôme de

MASTER ACADEMIQUE

Domaine : **Mathématique et Informatique**
Filière : **Informatique**
Spécialité : **Systèmes Distribués et Intelligence Artificielle**

Présenté par :

- Chaabani Rania
- Benglia Mira

Thème

Conception Et Réalisation D'un Outil De Programmation Automatique Basé Sur : Questions/Réponses

Soutenue le xx-xx- 2017 Devant le jury:

M.	……	MCA	Président
M.	……	MAA	Rapporteur
M. Meftah Mohammed Charaf Eddine		MAA	Encadreur

Année Universitaire: 2016-2017


بسم الله الرحمن الرحيم
إن الله
يغفر أن يشرك به ويغفر ما دون ذلك لمن يشاء
ومن يشرك بالله فقد افترى إثما عظيما

Remerciements

« الحمد الله الذي هدانا لهذا و ماكنا لنهتدي لولا أن هدانا الله »

Nous exprimons nos profondes gratitudes et respectueuse reconnaissance à notre encadreur

" Meftah Mohammed Charef Eddine "

Pour sa bonne volonté d'accepter de nous encadrer, pour tout le temps qu'il nous a octroyé et pour tous les conseils qu'il nous a prodigué.

Nous tenons aussi à remercier Monsieur

« Abbas Hamza »

Pour leur directive précieuse, et pour la qualité de leur suivis durant toute la période de notre projet.

Bien entendu, Nous tenons surtout á remercié Nos parents

Pour leurs sacrifices et leur patience,

tout au long de leurs vies.

Que toute personne ayant œuvré de près ou de loin à la réalisation de ce projet par une quelconque forme de contribution, trouve ici le témoignage de notre plus profonde reconnaissance.

Dédicace

Je tiens à dédier ce modeste travail à la lumière de ma vie: mes chers parents,

mon père « chaabani Azzouz » , et ma mère « Nacira » pour leur plus grand amour,

soutien, encouragement de la patience et de l'aide continue pendant mes années

d'études.

Ce travail est également dédié:

À mes sœur « Ilhem »

À mes douce ma nièce « Sana »

Pour tous mes oncles et tantes

Pour toute la famille Chaabani et la famille Nettari.

Je remerciement à notre encadreur

" Meftah Mohammed Charef Eddine "

Je désire dédier ce travail aussi à tous mes amies dans le campus,

et à tous ceux qui ont contribué à la réalisation de ce travail surtout Mr Abbas Hamza

Pour mon cher binôme Mira .

Pour tous ces gens aimables je suis très reconnaissante.

« Chaabani Rania »

Dédicaces

Avant tous ,je remercie dieu le tout puissant de m'avoir donné le courage et la patience pour réaliser ce travail malgré toutes les difficultés rencontrées.

Je dédie ce modeste travail :tous les amis et les professeurs estimés, en particulier le professeur ***« Meftah Mohammed Charaf Eddine »***

Et la famille est précieuse ,surtout ma mére ***Fatna***

Et mon père ***Lakhdar.***

*Ames frères :****Bassem,Ramdan,MED-Ziani.***

Ames sœurs.

Et à mon neveu ***Islam***

A l'esprit de grands pères .

Al'esprit de grands mère .

Aux chers amis .

*Un grand merci à Dr.****Abdul Malik jdiei*** *et collègue* ***Nabil Hani.***

A toute les amis (es) d'études surtout master ***Informatique 2016/2017.***

Aux habitants des Mons village ***Ain Cheikh*** *en général.*

Mira benglia

ملخص

الهدف من مشروعنا هو اقتراح أداة لتوليد البرامج تلقائيا (شفرة المصدر) من الأسئلة (التي تقدمها الأداة) والأجوبة (التي يقدمها المستخدم).

سوف نقوم بتحقيق مثل هذه الأداة بحيث تكون متاحة لجميع المستخدمين، وخصوصا أولئك الذين لا يعرفون البرمجة من اجل تطوير تطبيقاتهم (برامج) بطريقة سهلة وبسيطة.

كلمات المفتاحية : مجال , لغة , أداة برمجة

Résumé

Le but de ce travail est la proposition d'un outil qui permet de générer automatiquement des programmes (codes sources) à partir des questions (fournies par l'outil) et des réponses (fournies par l'utilisateur).

La réalisation d'un tel outil, permet à tous les utilisateurs et surtout ceux qui ne connaissent pas la programmation de développer leurs applications (programmes) de façon facile et simple.

Mots-clés : Domaine , Langage , Outil Programmation

Abstract

The aim of over project is to suggest a tool to gennerate programmes automatically (Source Code) from the questions which the programs provides and the answers which the user provides.

We implement such a tool to be available for all users mainly those who don't know programming to develop their application (programs) easily and simply.

Keywords: domain , language , programming tool

Sommaire

Sommaire

Chapitre :01

Un paradigme de programmation

Chapitre :02

Des travaux et des Outils existants

Chapitre :03

Partie 01 :

Architecture et conception

Partie 02 :

Modélisation Conceptuelle & Organisationnelle

Chapitre :04

Réalisation

Listes

Liste de Figure

Chapitre 02 :

Chapitre 03 :

Chapitre 04 :

Liste Schéma

Chapitre 01 :

Chapitre 03 :

Chapitre 04 :

Liste de Table

Chapitre 01 :

Chapitre 03 :

Chapitre 04 :

Liste d'Algorithme

Chapitre 01 :

Chapitre 03 :

Chapitre 04 :

Introduction

Introduction Générale

Introduction

1- Introduction

On vit aujourd'hui dans un monde qui dépend des systèmes, des logiciels et de la vie de programmation; au point que cette programmation intervienne dans les détails les plus minutieux de la vie.

La programmation est le seul langage entre l'homme et l'ordinateur. Elle se fait en écrivant un ensemble d'instructions transmises à un ordinateur et destiné à mettre en œuvre un ordre particulier.

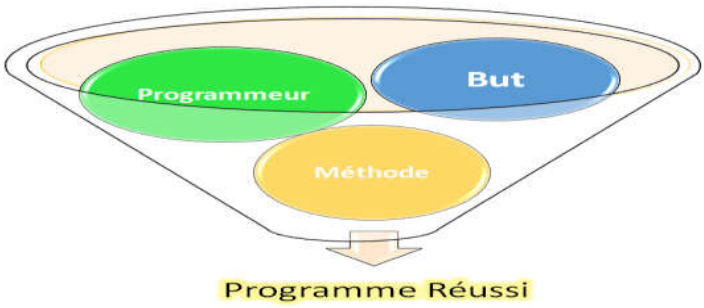
Il existe aujourd'hui des centaines de langages de programmation, qui diffèrent selon leurs objectifs et leurs fonctions.

Or, pourtant l'importance de ces langages de programmation, peu nombreux sont ceux qui peuvent construire effectivement des systèmes à vraie valeur qui peuvent servir d'une manière plus effective et pratique.

- **Problématique:**

Toute sorte de programmation planifiée est une corrélation étroite issue d'un programme intelligent. Ainsi explorer ou monde de programmation implique ce qui suit :

1. Programmeur
2. But
3. Méthode



La présence de ces trois conditions garantit la réussite du programme. Or, le problème réside dans l'ensemble des obstacles auxquels se heurte le programmateur qui se trouve dans la difficulté de:

- ✓ Choisir la langue appropriée, son apprentissage du niveau zéro, connaître ses détails les plus fins, la maitrise de l'échange avec cette langue et l'utiliser pour la programmation.
- ✓ Comment établir ou utiliser toutes ces instructions pour en faire un programme prêt à l'emploi.

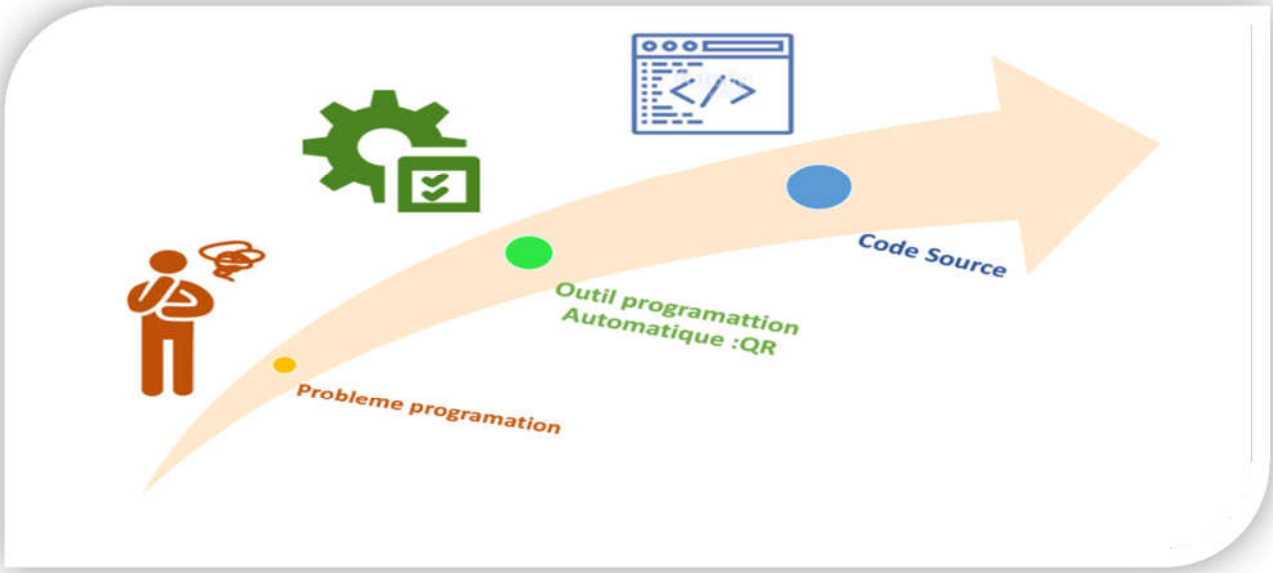
  D'où on conclut donc que la problématique principale du programmeur réside dans:

Comment créer un code source d'un programme a partir des questions (l'application) et des réponses (Utilisateur) ?

- **Solution:**

Pour rendre ce processus de programmation accessible et à la portée de quiconque , nous avons jugé utile de proposer un outil de programmation automatique attrayant basé sur la dualité:

Question/Réponse et cela pour toute personne non professionnelle mais désireuse de le mettre en place en dépit des connaissances rudimentaires



- ## Objectif de Ce Travail :

Notre projet vise à simplifier les choses complexes pour un programmeur pour que ces complexités lui deviennent claires et simples à travers le programme que nous proposons.

- ## Organisation Du Mémoire:

Notre mémoire est structuré en une introduction générale, une conclusion générale et trois chapitres:

- Le premier Chapitre intitulé : « **Les paradigmes de programmation** » consiste dans une étude générale sur la notion de « paradigmes de programmation » ; et ce, pour en donner une vue exhaustive à commencer par sa définition jusqu'à ses caractéristiques via ses classifications.

- Le deuxième chapitre intitulé : « Des travaux et des Outils existents »

- Le troisième chapitre intitulé : « **Architecture et Modélisation** » traite comment concevoir et planifier le programme ; où nous avons établi une structuration, arboration ou plutôt un algorithme convenable et Modélisation conceptuelle et organisationnelle

- Le quatrième chapitre qui correspond à l'étape de la : «**Réalisation**» comporte tous les outils utilisés dans notre projet et quelques aspects de notre propre programme.

Chapitre 01

Les paradigmes De Programmation

1- Introduction

Il existe des milliers de langage de programmation pour commander des machines (ordinateur, téléphone , robots,…) il sont souvent classés par catégories ,correspondant à des principes de fonctionnement différent :ces catégories sont appelées des "paradigmes de programmation ".
Chaque paradigme suggère sa propre façon de concevoir les programmes .

2- Définition des paradigmes de programmation

la notion de paradigme revient régulièrement lorsqu'il s'agit de décrire un langage de programmation. De manière générale un paradigme est un mode de représentation d'une réalité quelconque. C'est une manière structurée de voir et de concevoir. Ainsi, un paradigme de programmation est une manière de penser, d'écrire et de structurer un programme et donc par extension son code source.

Les paradigmes de programmation sont très nombreux et à un langage ne correspond pas un unique paradigme. La plupart du temps un langage autorise plusieurs paradigmes voire des paradigmes «dégénérés» dans le sens où toutes les fonctionnalités de ces derniers ne sont pas forcément présente dans le langage.[7]

3- Les caractéristiques les paradigmes de programmation :

Les langages de programmation partagent avec les langages naturels un certain nombre de caractéristiques qui sont de trois ordres : lexicale, syntaxique et sémantique.[4]

3-1- **Les caractéristiques lexicales**

La suite de caractères représentant le module est la forme concrète de communication entre le programmeur et le traducteur. Il apparaît alors nécessaire de découper le module source, en tant que chaîne de caractères, en une suite de symboles du langage, chaque symbole étant représenté par une suite de caractères. Considérons, par exemple, le petit bout de programme source suivant:

```
begin x := 23; end
```

Cette suite de caractères doit être découpée de la façon suivante:

Begin	x	:=	23	;	end

3-2- Les Caractéristiques Syntaxiques :

Le résultat obtenu après analyse lexicale est une suite de symboles. Mais, toute suite de symboles ne constitue pas un programme. Il faut donc vérifier la concordance de la suite avec la structure du langage

C'est ce que l'on appelle l'analyse syntaxique.

L'analyse syntaxique consiste à retrouver, à partir du programme source, ou en fait à partir de la suite de symboles, quelles sont les règles que le programmeur a appliquées. On utilise une syntaxe indépendante du contexte, pour permettre cette reconstruction.

Plus généralement, l'interprétation est la suivante:

- Les structures syntaxiques s'écrivent sous la forme de mots encadrés par les signes < et >.
- Les symboles s'écrivent sous leur forme lexicale.
- Une règle de production comporte une structure syntaxique unique à gauche du signe ::= et une ou plusieurs suites de symboles et de structures syntaxiques, les suites étant séparées par le signe |indiquant un choix parmi les suites.

3-3- Les caractéristiques sémantiques :

Après avoir contrôlé la correction du programme, et en avoir obtenu la structure syntaxique, la traduction continue par la phase d'analyse sémantique, qui a pour but de trouver le sens des différentes actions voulues par le programmeur.

L'analyse sémantique des actions du programme a pour but de permettre au traducteur de déterminer avec précision la suite des actions qu'il doit exécuter indépendamment de tout aspect langage. Prenons par exemple le morceau de programme suivant:

```
Var i : entier ;
     S :réel ;
Début
    S :=0 ;
    I :=0 ;
     Tantque i<10 faire
                s :=s+sqrt (i) ;
               i :=i+1 ;
    Fait
Fin
```

Alg 01 : Algorithme de Programme Caractéristiques Sémantiques

4- La classification des paradigmes de programmation :

Les langages de programmation sont répartisen Troie catégories: les langages de bas niveau et les langages de haut niveau et Machines-Langages Il importe de caractériser chacune de ces catégories ,pour ensuite traiter des nouveaux développements dans ce domain.

La Classification des langages de Programmation			
	Nom de langage	Annees	Définition
BasNiveau	Le langage machine	Avant 1940	À l'origine, l'écriture d'un programme se faisait en langage machine, à partir d'un alphabet binaire, d'où l'appellation équivalente de langage de première génération.
	Assembleur	1949	Les assembleurs existent depuis le début des ordinateurs. Ils associent un nom symbolique au code du langage machine, par exemple: add bx, 4 cmp [adr], 3 // comparaison jmp address // branchement La programmation en assembleur ne se pratique plus sur les ordinateurs actuels même pour les routines d'exécution rapides.
Haut Niveau	Fortran	1954-1958	John Backus et autres chercheurs d'IBM Langage dédié aux calculs mathématiques. Fortran II en 1958 a introduit les sous-programmes les fonctions, les boucles, une structure de contrôle FOR primitif. Les identifieurs avaient au plus six caractères.
	COBOL	1960	en réponse à des besoins de traitement de données commerciales. C'est donc essentiellement un langage adapté aux applications de gestion. Il a été conçu pour que son utilisation soit indépendante du type d'ordinateur.
	BASIC	1964	C'est un langage de haut niveau qui convient aussi bien aux applications de gestion qu'à celles d'ingénierie ou de type scientifique. Même si à

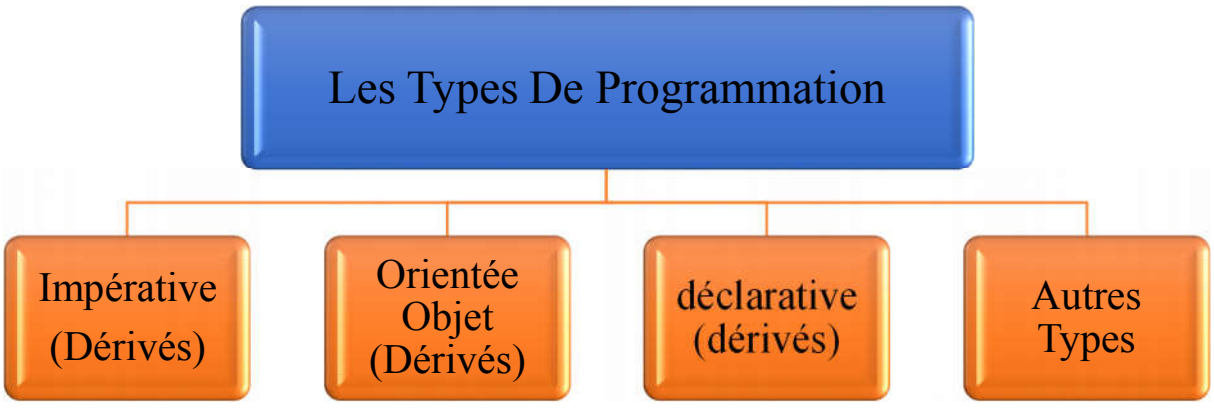
			l'origine il était conçu particulièrement pour les systèmes à temps partagé, il est aujourd'hui très populaire sur les ordinateurs personnels, fonctionnant essentiellement en monoprogrammation.
	Pascal	1970	C'est un langage évolué, conçu par N. Wirth, dont les domaines d'application sont la gestion, la science, l'ingénierie et l'éducation. Il s'est d'abord développé dans la communauté scientifique, particulièrement dans les universités. Actuellement, il est l'un des langages les plus utilisés en éducation et particulièrement sur les ordinateurs personnels.
	langage C	1972	est un langage de programmation à but général , C'est donc un langage qui est recommandé à la fois pour les applications scientifiques, d'ingénierie et de développement de logiciel. Aussi, est-il souvent utilisé pour la programmation de systèmes d'exploitation, de systèmes de base de données et de traitement de texte.
	Le GPSS	1960	est le premier langage mis au point pour simuler par ordinateur des événements tels que les arrivées de bateaux dans les ports, le contrôle de la circulation dans les villes, dans les aéroports, les hôpitaux, etc
	Simula	1967	C'est un langage spécialisé, principalement utilisé pour programmer des applications de type simulation. Plus puissant et plus riche que le GPSS,
Machines-	Lisp	1958 - 1960	Langage fonctionnel de traitement de liste. Il est récursif et non itératif. Les données et les programmes ne sont pas distingués et peuvent être traités de la même façon.

	Prolog	1970+	Il introduit la programmation logique. Un programme est composé de clauses de Horn. Prolog se dit déclaratif parce que son système d'inférences logiques constitue un mécanisme de résolution.

Table 01 : La Classification Des Langages De Programmation

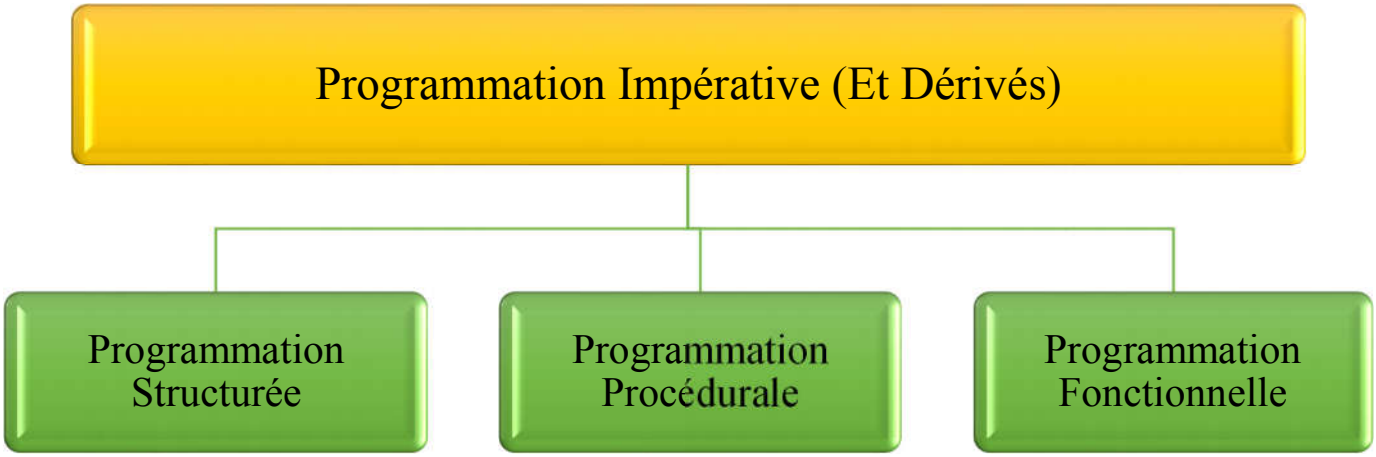
5- Les types de programmation :

Un langage de programmation fournit une abstraction de niveau supérieur pour utiliser une machine, car très peu d'humains comprennent le langage machine. La notion de programme est apparue au cours de la deuxième moitié du XIXème siècle, mais les premiers langages de programmation datent des environs de l'année 1950. Dans les années 60, de nombreux langages spécialisés sont créés. [7]



Schema 01 : Les Types DeProgrammation

5-1 Types de programmation impérative (et dérivés) :



Schema 02 : Types de programmation impérative

5-1-1 Programmation impérative:

Impératif : on décrit ce que l'on fait mécaniquement, il n'y a aucune part de choix qui n'appartient pas au développeur. Par exemple une requête SQL n'est pas impérative, le SGBD fait comme il veut pour récupérer les données, de même pour le HTML, le navigateur internet est libre d'interpréter la page comme il veut. [16]

5-1-2 Programmation structurée :

C'est un paradigme important de la programmation, apparu vers 1970 Structuré : on découpe le code en morceaux plus petits, on parcourt le code de haut en bas et quand on passe à un autre morceau de code c'est toujours par le début qu'on commence. Un langage qui utiliserait des label/goto n'est pas structuré par exemple.[16]

5-1-3 Programmation procédurale :

Procédural : la structure du code est découpé en procédures (souvent appelées fonctions), l'ordre dans lequel ces fonctions sont appelées n'est pas important. On peut revenir pus haut dans le code

pour appeler une autre fonction, voire s'appeler soit même. Généralement les fonctions peuvent avoir des paramètres et des résultats.[16]

5-2 Types de programmation orientée objet (et dérivés)

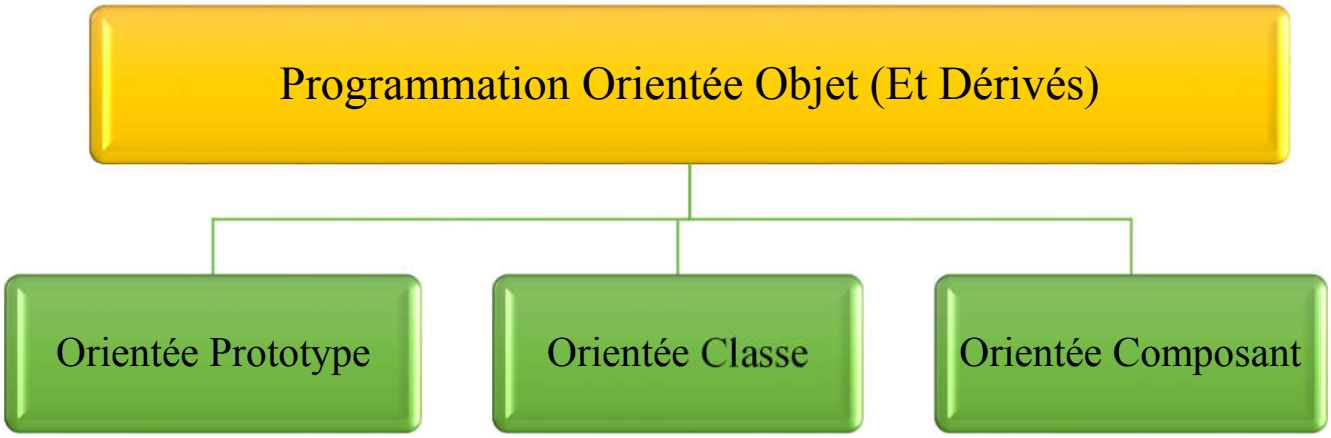


Schéma 03 : Types de programmation orientée objet

5-2-1 Programmation orientée objet

La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique élaboré par les Norvégiens Ole-Johan Dahl et Kristen Nygaard au début des années 1960 et poursuivi par les travaux d'Alan Kay dans les années 1970. Il consiste en la définition et l'interaction de briques logicielles appelées objets, un objet (monde physique: comme une voiture, une personne ou encore une page d'un livre).[22]

5-2-2 Programmation orientée prototype

La programmation orientée prototype est un style de programmation orientée objet qui n'utilise pas les classes. La réutilisation des propriétés d'un objet (appelée héritage pour les langages à classe) est effectuée via des objets qui seront des prototypes pour d'autres objets.
langage de programmation les prototypes :JavaScript, Cecil, Newton Script,.....[23]

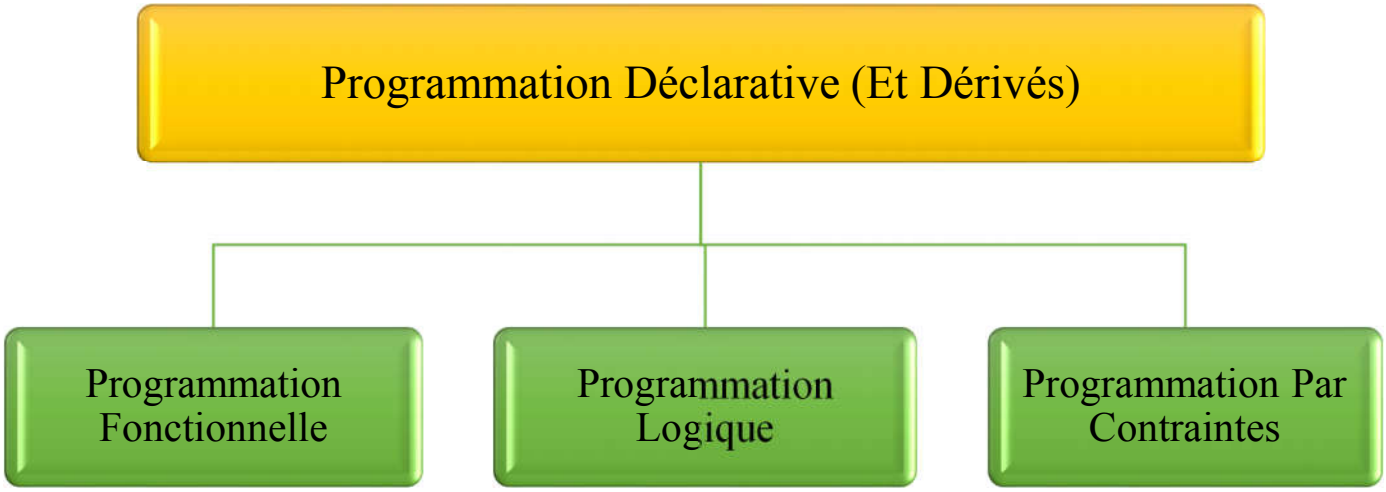
5-2-3 Programmation orientée classe

Une classe est la définition d'un nouveau type de variable particulier. Ce type est similaire à une structure ou enregistrement. Il rassemble plusieurs champs de données[24]

5-2-4 Programmation orientée composant

La programmation orientée composant (POC) consiste à utiliser une approche modulaire de l'architecture d'un projet informatique, ce qui permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance. La POC n'est pas sans similitudes avec la POO, puisqu'elle revient à utiliser une approche objet, non pas au sein du code, mais au niveau de l'architecture générale du logiciel.[25]

5-3 Types de programmation déclarative (et dérivés)



Schema 04 : Types de programmation déclarative

5-3-1 Programmation déclarative

La programmation déclarative est un paradigme de programmation. Elle consiste à créer des applications sur la base de composants logiciels indépendants du contexte et ne comportant aucun état interne. Par exemple, les pages HTML sont déclaratives car elles décrivent ce que contient une page (texte, titres, paragraphes, etc).[25]

5-3-2 Programmation fonctionnelle

La programmation fonctionnelle est un paradigme de programmation. De la même manière que la programmation objet est une évolution de la programmation procédurale, la programmation fonctionnelle est une évolution de la programmation objet.[24]

5-3-3 Programmation logique

La programmation logique est une forme de programmation qui définit les applications à l'aide d'un ensemble de faits élémentaires les concernant et de règles de logique leur associant des conséquences plus ou moins directes. Ces faits et ces règles sont exploités par un démonstrateur de théorème ou moteur d'inférence, en réaction à une question ou requête.[25]

5-3-4 Programmation par contraintes

La programmation par contraintes (PPC, ou CP pour Constraint Programming en anglais) est un paradigme de programmation apparu dans les années 1970 et 1980 , permettant de résoudre des problèmes combinatoires de grandes tailles tels que les problèmes de planification et d'ordonnancement . [25]

5-4 Autres types:

Types De Programmation	Définition
Programmation événementielle	Dans la programmation évènementielle, les "réactions" du programme sont définies par différents évènements.
Programmation séquentielle	La programmation séquentielle laquelle la séquence d'instructions exécutée est déterminée ou modifiée en permanence par les différents événements extérieurs .
Programmation interruptible	Programmation interruptible c'est un programme est briser la continuité de, rompre la continuation de (bloquer)
Programmation concurrente	La programmation concurrente est l'existence de plusieurs piles sémantiques qui peuvent être appelées threads, processus

	ou tâches.
Programmation orientée aspect	La POA n'est pas un language de programmation, mais elle est une extension de language. Elle peut effectivement être appliqué au Java, C, C++, C#
Programmation par contrat	La programmation par contrat est une technique de conception ,L'objectif de cette méthode est d'arriver à écrire du code plus robuste et plus facilement réutilisable.
Programmation chimique	c'est un programmes sont vus comme des solutions chimiques abstraites. Les données sont des molécules dont les réactions chimiques représentent les opérations.
Programmation orientée agent	Dans cette approche, les agents sont les éléments centraux du langage, de la même façon que les objets sont centraux pour les langages orientés objets.
Programmation non-déterministe	Un langage de programmation non-déterministe est un langage n'est possible pas de prévoir en fonction de l'état actuel
Programmation réactive	la programmation réactive visant à conserver une cohérence d'ensemble en propageant les modifications d'une source réactive (modification d'une variable, entrée utilisateur, etc.)
Programmation synchrone	La programmation asynchrone permet de lancer plusieurs processus simultanément. Cela permet dans certains cas un gain important de temps.
Programmation par annotations	En programmation, une annotation est un élément permettant d'ajouter des méta-données à un code source.
Programmation par messages	Programmation par messages est un programme de communication entre ordinateurs ou entre processus à l'intérieur d'un même ordinateur.
Programmation récursive	La programmation récursive est une autre méthode permettant de répéter un nombre indéterminé de fois une action. on crée une fonction qui va effectuer une action
Programmation réflexive	la réflexion est la capacité d'un programme à examiner, et éventuellement à modifier, ses propres structures internes de haut niveau lors de son exécution.

Programmation scalaire	Un processeur est dit scalaire s'il ne traite qu'une seule donnée à la fois.

Table 02 : Autres types de programmations

6- Conclusion

Dans ce chapitre , nous avons mis en relief le concept de paradigme de programmation, sa relation avec les langages, et son évolution dans le temps. Il est clair que cette évolution se rapproche de plus en plus de la pensée naturelle de l'homme. L'exposition à plusieurs paradigmes de programmation permet d'avoir un panel d'approches et une multitude de solutions à chaque situation

Chapitre02

Des travaux et des Outils existants

1- Introduction

Des différentes spécifications bases sur des multiples modèles sont obtenues pour décrire les aspects différents d'un système.

Diverses voies, regroupées sous le terme de génie logiciel, ont été explorées pour tenter de fabriquer des logiciels avec une qualité comparable à celle des produits industriels: organisation du processus de fabrication, méthodes de conception (Merise par exemple), spécifications, structuration des programmes (programmation structurée, modulaire puis objet), preuve mathématique, réutilisation (bases de données en premier lieu), techniques de test. Malgré les améliorations partielles qu'elles ont apportées, aucune de ces voies n'a permis de résoudre le problème de la fabrication des logiciels qui reste pour l'essentiel artisanale.

La génération automatique du code source est une voie vers la création de meilleurs logiciels.

Est une opération permettant de générer automatiquement du code source. Son but est d'automatiser la production de code source répétitif afin de minimiser les risques d'erreurs et de permettre au programmeur de se concentrer sur l'écriture de code à plus grande valeur ajoutée.

Il existe de nombreuses sources à partir desquelles générer le code source :

- ✓ Génération à partir de programmes informatiques écrits dans un autre langage de programmation, généralement de plus haut niveau. Le logiciel réalisant cette transformation peut être appelé compilateur ou traducteur. Par exemple, on peut générer du code Java ou C# à partir d'un programme décrit en UML. Un autre exemple serait un compilateur pour Eiffel générant du code C qui est ensuite lui-même compilé.
- ✓ Génération à partir de métadonnées. Par exemple, on peut générer la couche logicielle d'accès à une base de données relationnelle grâce aux métadonnées des tables .

Le résultat est souvent excellent, mais il pourrait être amélioré. De plus, elle peut se révéler nettement plus rapide et plus économique, ce qui n'est pas moins important.

Plusieurs travaux (approches et outils) sont consacrés à la génération automatique des codes sources. Ces travaux peuvent être regroupés en deux types :

- ✓ Des outils de génération de code basé sur les architectures dirigées par modèle (MDA).
- ✓ Des outils de génération automatique du code basé sur des méthodes formelles

Dans ce chapitre nous allons présenter et discuter ces travaux :

2- Outils de génération de code basée sur les architectures dirigées par modèle (MDA) :

Le principe de base du MDA est l'élaboration de différents modèles, en partant d'un modèle métier indépendant de l'informatisation (computation independent model, CIM), la transformation de celui-ci en modèle indépendant de la plate-forme (platform independent model, PIM) et enfin la transformation de ce dernier en modèle spécifique à la plate-forme cible (platform specific model, PSM) pour l'implémentation concrète du système. Les PSMs peuvent utiliser des langages spécifiques à un domaine ou des langages généralistes comme Java, C#, Python… Les techniques employées dans le cadre de l'approche MDA sont donc principalement des techniques de modélisation et des techniques de transformation de modèles.

Un exemple typique de l'approche MDA est la génération automatique de code source à partir d'une modélisation UML, qui suppose de combiner :

- le standard UML, et l'outil de modélisation qui l'implante (ex. : Rose, Enterprise Architect, Together, MagicDraw, RSA, Dia) ;
- des templates de génération UML→code source, et l'outil de génération de code (ex MIA-Studio) ;
- le tout intégré dans une « chaîne » de production.

Il n'est bien sûr pas requis que TOUT le code soit généré automatiquement, mais l'architecture globale du système (ex: squelettes de code) au moins doit être obtenue ainsi.

Les transformations entre le CIM, le PIM et les PSM sont souvent automatisés à l'aide d'outils. Ces transformations sont réalisées avec des outils plus ou moins compatibles avec le standard de l'OMG nommé QVT.

Le passage du PSM à la génération du code est la suite logique de ce traitement. Elle peut être réalisée par des générateurs afin de produire tout type de cibles technologiques.[19]

2-1 Mia-Studio

Une solution de génération de code qui exploite toutes les fonctionnalités offertes par la plateforme standard Eclipse.

Destiné aux architectes et développeurs, Mia-Studio est un atelier de création et d'exécution de générateurs de code exploitant des modèles de type UML ou des modèles conformes à un langage de modélisation de l'entreprise (DSL)

Mia-Studio s'interface avec les modeleurs UML du marché et en particulier MagicDraw et RSA.

Une version communautaire est aussi disponible.

Mia-studio 9 : une révolution pour vos développements d'applications JAVA

- Permettre aux développeurs de produire rapidement un code de qualité
- Faciliter le développement de générateurs de code et leur déploiement
- Utiliser les générateurs dans l'environnement de développement

Mia-Studio 9 est un ensemble de plugins permettant à partir d'un environnement de développement basé Eclipse de créer et d'utiliser des générateurs de code et des transformateurs de modèles. Le développeur accède à toute la puissance des produits Mia-Software intégrés dans son poste de travail Eclipse. [21]

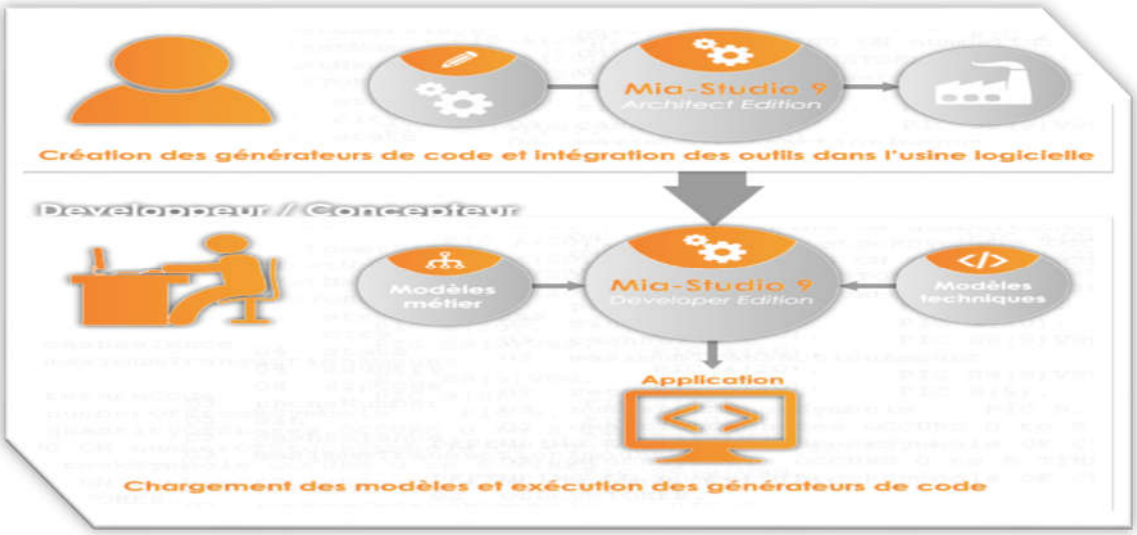


Figure 01 : Architecture utilisant logiciel

2-2 Acceleo :

Le projet Acceleo est né en 2006 autour du site web Acceleo.org. Acceleo 1.0 et 1.1 étaient à l'époque sous licence GPL et compatible avec Eclipse 3.2 et de nombreux modeleurs basés sur EMF ou UML 1.32. Quelques mois plus tard, lors de la sortie d'Acceleo 1.2, le projet Acceleo changea de licence pour adopter la licence EPL3. La version 2 d'Acceleo fut disponible le 5 juin 2007 après l'ouverture du site planet.acceleo.org regroupant les publications de nombreux acteurs de la communauté Acceleo et de la ferme de module regroupant des générateurs de code conçu avec Acceleo.

Acceleo est un générateur de code source de la fondation Eclipse permettant de mettre en œuvre l'approche MDA (Model driven architecture) pour réaliser des applications à partir de modèles basés sur EMF. Il s'agit d'une implémentation de la norme de l'Object Management Group (OMG) pour les transformations de modèle vers texte (M2T) Model to Text.

Fonctinnalité de Acceleo

Acceleo fournit des outils pour la génération de code depuis des modèles. Grâce à ces outils, Acceleo permet notamment de réaliser des générations incrémentales. La génération incrémentale consiste à générer du code puis à pouvoir modifier le code généré librement et à re-générer le code sans pour autant perdre les modifications réalisées à la main sur le code généré précédemment.

Acceleo permet aussi :

- interopérabilité des méta-modèles d'entrée (UML 1 / UML 2 / DSL conformes à EMF).
- syntaxe arborescente dédiée à la manipulation de modèles.
- personnalisation de la génération par templates.
- indépendance du langage généré.

Exemple

Acceleo peut prendre en entrée n'importe quel type de modèles réalisés avec EMF comme des modèles UML ou des modèles représentant des langages dédiés à un domaine. À partir des éléments utilisés par ce modèle, on peut réaliser un template qui permettra la génération de code.

Figure 02 : Exemple de modèles et templates

Ici, on utilise les éléments EClass, EAttribute et EOperation de EMF. Grâce à ce template et à ce modèle fournit en entrée, Acceleo peut générer le code précédent. Les templates présent dans cet exemple sont paramétrés pour générer du Java mais la norme MOFM2T est indépendante du langage généré. Une fois le générateur paramétré, il suffit de modifier le modèle pour générer un code d'apparence similaire mais avec un contenu différent. [11]

2-3 Eclipse Modeling Framework

Le projet Eclipse Modeling Framework (EMF) est un framework de modélisation, une infrastructure de génération de code et des applications basées sur des modèles de données structurées. Partant d'une spécification décrite généralement sous la forme d'un modèle en XMI, EMF fournit des outils permettant de produire des classes Java représentant le modèle avec un ensemble de classes pour adapter les éléments du modèle afin de pouvoir les visualiser, les éditer avec un système de commandes et les manipuler dans un éditeur.

Modélisation EMF

Le projet EMF permet de créer deux types de modèles, d'un côté des modèles définissant des concepts, souvent nommé le méta-modèle, et de l'autre des modèles instanciant ces concepts. À titre d'exemple, on peut définir un méta-modèle définissant des concepts tels que « Classe », « Operation » et « Attribut » et ensuite utiliser ce méta-modèle pour définir des modèles contenant par exemple un élément de type « Classe » nommé « Voiture » avec une « Operation » nommée « rouler » et un « Attribut » nommé « consommation ». Tout modèle EMF est une instance d'un modèle EMF avec pour racine commune le modèle Ecore fourni par EMF. EMF permet non seulement de créer un méta-modèle représentant les concepts désirés par l'utilisateur mais il permet ensuite à l'utilisateur de créer des modèles issus de ce méta-modèle et de les manipuler avec un outillage adapté.

Exemple de modélisation EMF

La première étape dans le cadre d'une utilisation simple d'EMF consiste à créer un projet Java dans Eclipse à définir son méta-modèle en créant un nouveau fichier de type "ecore" avec l'aide des wizards fournit par EMF.

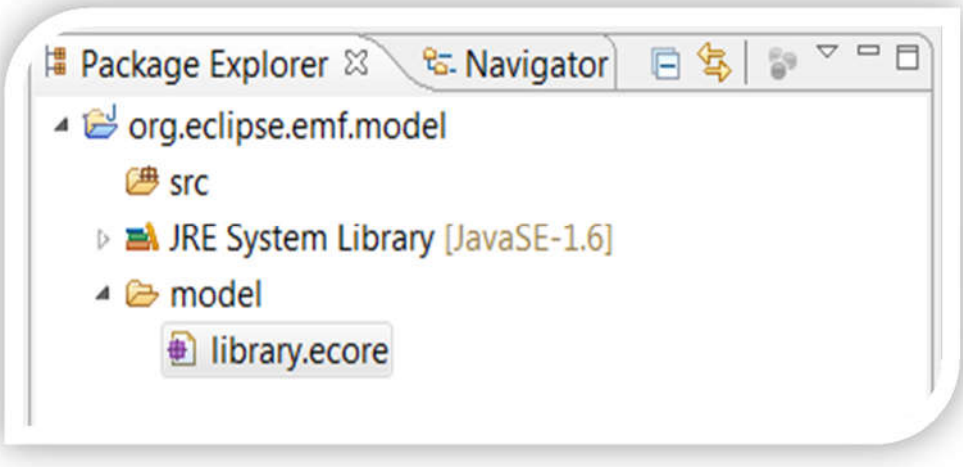


Figure 03 : Un projet Java avec un modèle EMF

Dans ce modèle EMF, on peut ensuite créer un package, racine d'un méta-modèle définit avec EMF. La vue "Properties" d'Eclipse permet de définir le nom de ce package avec son préfixe et son uri. L'uri d'une instance de package venant d'Ecore est un identifiant unique permettant d'identifier un méta-modèle.

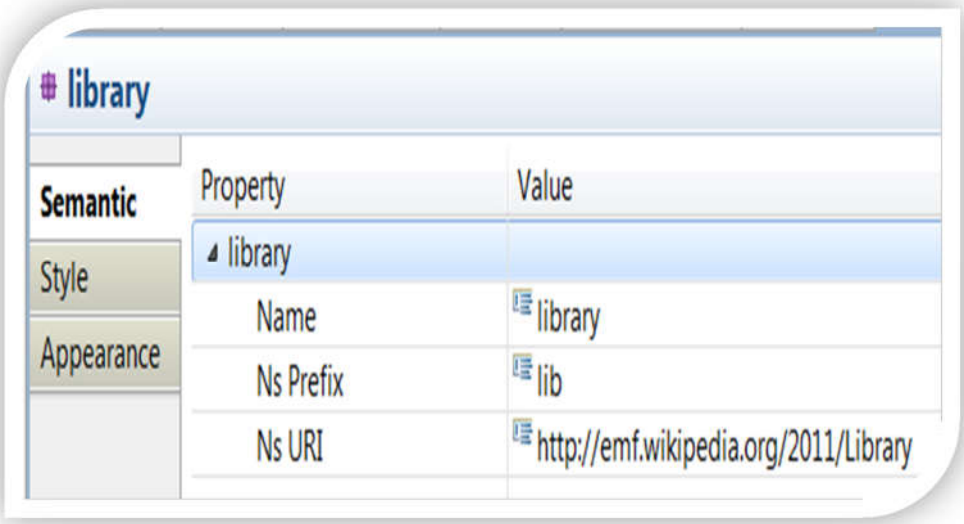


Figure 04 : Capture d'écran de la vue "Properties" pour un package

Au sein de ce package, nous pouvons définir les différents concepts de notre domaine que nous souhaitons manipuler avec EMF. Dans cet exemple, nous pouvons voir des instances de classes venant d'Ecore permettant de définir les concepts d'une bibliothèque simple. L'image ci-dessous montre donc une instance du concept de package, quatre instances du concept de classes, une instance d'énumération et plusieurs instances de références et attributs.

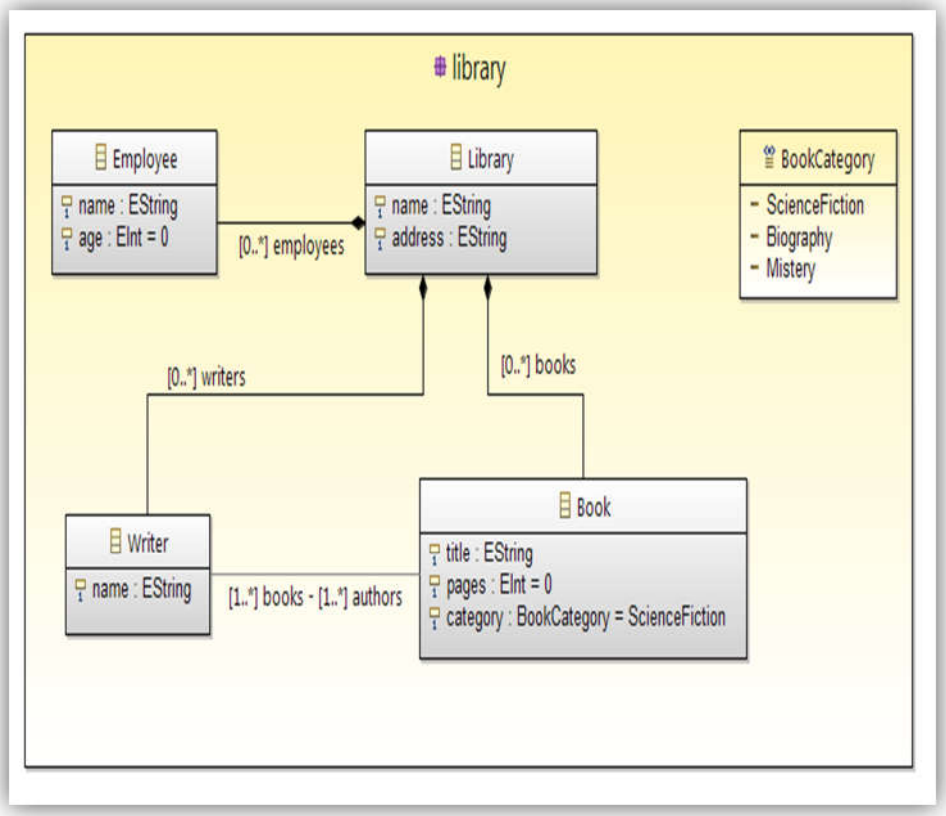


Figure 05 : Le méta-modèle "Library" définit avec EMF.

Une fois ce méta-modèle définit, nous pouvons l'utiliser avec l'éditeur basique pour fichier "ecore" pour créer une instance du concept "Library" dans le méta-modèle. On peut donc voir sur la capture suivante, l'éditeur basique d'EMF avec une instance de nos concepts Library, Book, Employee et Writer. Lorsque l'on sélectionne un élément, on peut voir et éditer ses attributs dans la vue "Properties".

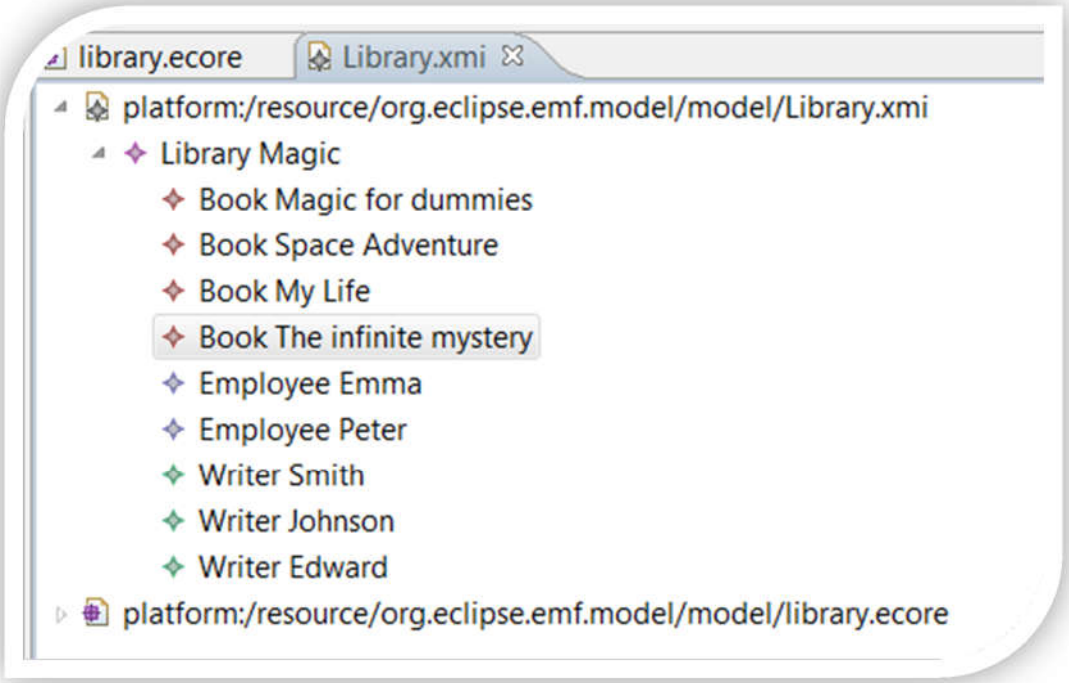


Figure 06 : Un modèle dynamique instance de notre méta-modèle.

Cet exemple représente une utilisation très basique d'EMF. Il est par exemple possible d'initialiser un modèle "ecore" à partir d'une série de classes java ou bien d'un fichier "xsd". EMF permet aussi de choisir la sérialisation à utiliser pour un modèle EMF (par défaut XMI).[13]

2-4 Simulink

Est un environnement de diagramme fonctionnel destiné à la simulation multidomaine et à l'approche de conception par modélisation Model-Based Design. Il prend en charge la conception et la simulation au niveau système, la génération automatique de code, ainsi que le test et la vérification en continu des systèmes embarqués.

Simulink propose un éditeur graphique, un ensemble personnalisable de bibliothèques de blocs et des solveurs pour la modélisation et la simulation de systèmes dynamiques. Il est intégré à MATLAB, ce qui vous permet d'incorporer les algorithmes MATLAB dans les modèles et d'exporter le résultat des simulations vers MATLAB pour compléter les analyses.

Principales fonctionnalités de Simulink

- Éditeur graphique pour la création et la gestion de diagrammes fonctionnels hiérarchiques
- Bibliothèques de blocs prédéfinis pour la modélisation de systèmes en temps continu ou en temps discret
- Moteur de simulation avec des solveurs ODE à pas fixes ou variables
- Scopes et affichage de données pour la consultation du résultat de la simulation
- Outils de gestion de données et de projets pour les données et fichiers des modèles
- Outils d'analyse de modèles pour l'affinage de l'architecture de modèle et l'augmentation de la vitesse de simulation
- Bloc MATLAB Function pour l'import d'algorithmes MATLAB dans les modèles
- Outil Legacy Code Tool pour l'import rapide de code C et C++ existant dans les modèle .[26]

3- Les méthodes formelles

En informatique, les méthodes formelles sont des techniques permettant de raisonner rigoureusement, à l'aide de logique mathématique, sur des programmes informatiques ou du matériel électroniques, afin de démontrer leur validité par rapport à une certaine spécification.

Elles sont basées sur les sémantiques des programmes, c'est-à-dire sur des descriptions mathématiques formelles du sens d'un programme donné par son code source (ou parfois, son code objet).

Ces méthodes permettent d'obtenir une très forte assurance de l'absence de bug dans les logiciels (Evaluation Assurance Level, Safety Integrity Level). Elles sont utilisées dans le développement des logiciels les plus critiques. Leur amélioration et l'élargissement de leurs champs d'application pratique sont la motivation de nombreuses recherches scientifiques en informatique.

3-1 Présentation du système GAP (Générateur Automatique de Programmes) :

Le système GAP utilise une approche qui permet d'industrialiser la fabrication des logiciels de gestion en générant les programmes à partir d'une description abstraite des données.

Principe : produire automatiquement les programmes à partir d'une description abstraite des données

A partir d'un modèle mathématique (ensembliste) décrivant les données, leurs relations et les principales fonctions de calcul, le système GAP produit automatiquement le logiciel (programmes serveurs et écrans). Les fonctions de validation des dossiers (workflow), d'identification des utilisateurs et de restriction de leurs droits d'accès aux données sont intégrées dans le système.

Les calculs complexes et les écrans qui leurs correspondent doivent être programmés manuellement.

Cette approche qui consiste à générer les programmes à partir d'une spécification formelle du résultat attendu a déjà été utilisée avec succès dans le domaine des automates industriels. Elle n'avait que peu été explorée pour les applications de gestion. Le système GAP comble cette lacune.[15]

3-2 La méthode B :

La méthode B est une méthode formelle qui permet le raisonnement sur des systèmes complexes1 ainsi que le développement logiciel.

La méthode B permet de formaliser le système et son environnement de manière abstraite, puis par raffinements successifs, de rajouter les détails au modèle du système. Une activité de preuve formelle permet de vérifier la cohérence du modèle abstrait et la conformité de chaque raffinement avec le modèle supérieur (prouvant ainsi la conformité de l'ensemble des implémentations concrètes avec le modèle abstrait).

On distingue :

- le B classique tel qu'il est défini dans le B Book de 1962. L'outil logiciel de support est l'atelier B3, ou le B-Toolkit 4.
- le B événementiel qui est une évolution utilisant uniquement la notion d'événements pour décrire les actions et non plus les opérations (qui sont proches des routines informatiques). Par conséquent, la méthode peut s'appliquer pour l'étude des systèmes de domaines variés, plus seulement à des programmes. On réalise alors des développements incrémentaux de systèmes prouvés. Pour cela on utilise toujours l'atelier B3
- le B# (B sharp) qui est une reprise du B événementiel avec des éléments de la notation Z. L'atelier logiciel change et s'appelle Rodin5.
-

La méthode B et la génération du code Source

La méthode B permet de modéliser de façon abstraite le comportement et les spécifications d'un logiciel dans le langage de B, puis par raffinements successifs d'aboutir à un modèle concret dans un sous-ensemble du langage B transcodable en Ada ou en C, exécutables par une machine concrète.

Le code produit est issu de la traduction des implémentations écrites dans le langage B0, sous-ensemble du langage B et d'une syntaxe proche d'un quelconque langage structuré. La phase de traduction permet alors de produire un code cible de forme similaire à celle du modèle B0.

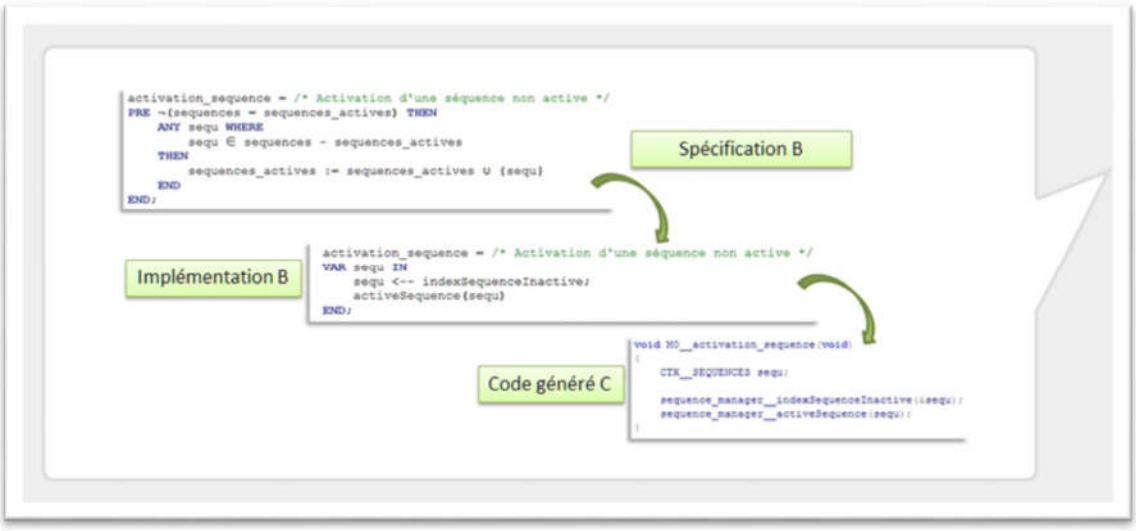


Figure 07 : Le code produit :écrites dans le langage B0

Cette traduction dite **« 1 pour 1 »** permet le cas échéant de vérifier facilement le code en sortie du générateur par comparaison avec le modèle B0.[1]

4- Synthèse et conclusion

Après avoir présenté des différents travaux relatifs à la génération automatique du code, nous avons constaté qu'un certain nombre de points forts et autres faibles n'étaient pas traités, ou tout du moins, pas de manière satisfaisante. :

Points forts

- ✓ Les approches et les outils MDA utilise des standards (diagrammes UML,…) accessibles qui masquent la complexité du système.

- ✓ Ces approches fournissent aussi un moyen visuel de comparer le modèle de conception et le modèle d'implémentation tout en vérifiant l'absence de violation de propriétés générales.

- ✓ Ces approche formalise certains opérateurs utilisés pour le langage de programmation et permet ainsi certaines vérifications.
- ✓ Son haut degré d'abstraction lui permet de décrire une programmation à différents niveaux (module, composant services, orchestration,…).

Points faibles

- ✓ UML étant semi-formel et plusieurs langages de programmation n'ayant aucune sémantique formelle, il est alors impossible de prouver que le code généré correspond exactement à ce qui est décrit. En ce sens, les comparaisons et les vérifications ne peuvent être que partielles.
- ✓ Ces 'approches désolidarise les phases de conception et d'implémentation pour les regrouper, une fois ces dernières indépendamment terminées. En cas de non cohérence (concordance) de traces, l'implémentation est donc totalement à reprendre : la phase de vérification est alors très/trop tardive.
- ✓ Ces approche n'offre pas un moyen d'expression proche de certaine domaine
- ✓ Les approches formelles présentent une syntaxe complexe et difficile à comprendre et à utiliser par n'importe qui.

Ainsi dans notre prochain chapitre nous allons présentons notre approche pour la génération automatique des codes sources basée sur une interaction avec l'utilisateur qui peut être non informaticien, cette interaction de type : Questions –réponses.

Chapitre03

Architecture et Modélisation

I. Introduction

Nous avons abordé ce chapitre, en prenant en considération 02 partie :

Partie 01 : Architecture et Conception Ce partie sera consacré a la architecture et conception qui est l'étape la plus importante d'un notre projet. Elle a pour but à faciliter le concept général du programme pour se faire, nous avons proposé :

- ✓ Architecture général du système
- ✓ Arbre général du système
- ✓ Algorithme général proposé

Partie 02 : Modélisation Conceptuelle & Organisationnelle

Partie 01 : Architecture et Conception

I- Architecture et Arbre

1-Architecture general propose

La figure suivant l'architecture générale de notre application qui explique les tâches

Utilisateur - Système.

- **Système :** Pose de nombreuses questions et des réponses possibles…
- **Utilisateur :** * sélectionne une réponse parmi les Réponses proposées par le système.

  * Proposition (Domaine, ….)

Après chaque sélection fourni par l'utilisateur le système génére une partie du code source

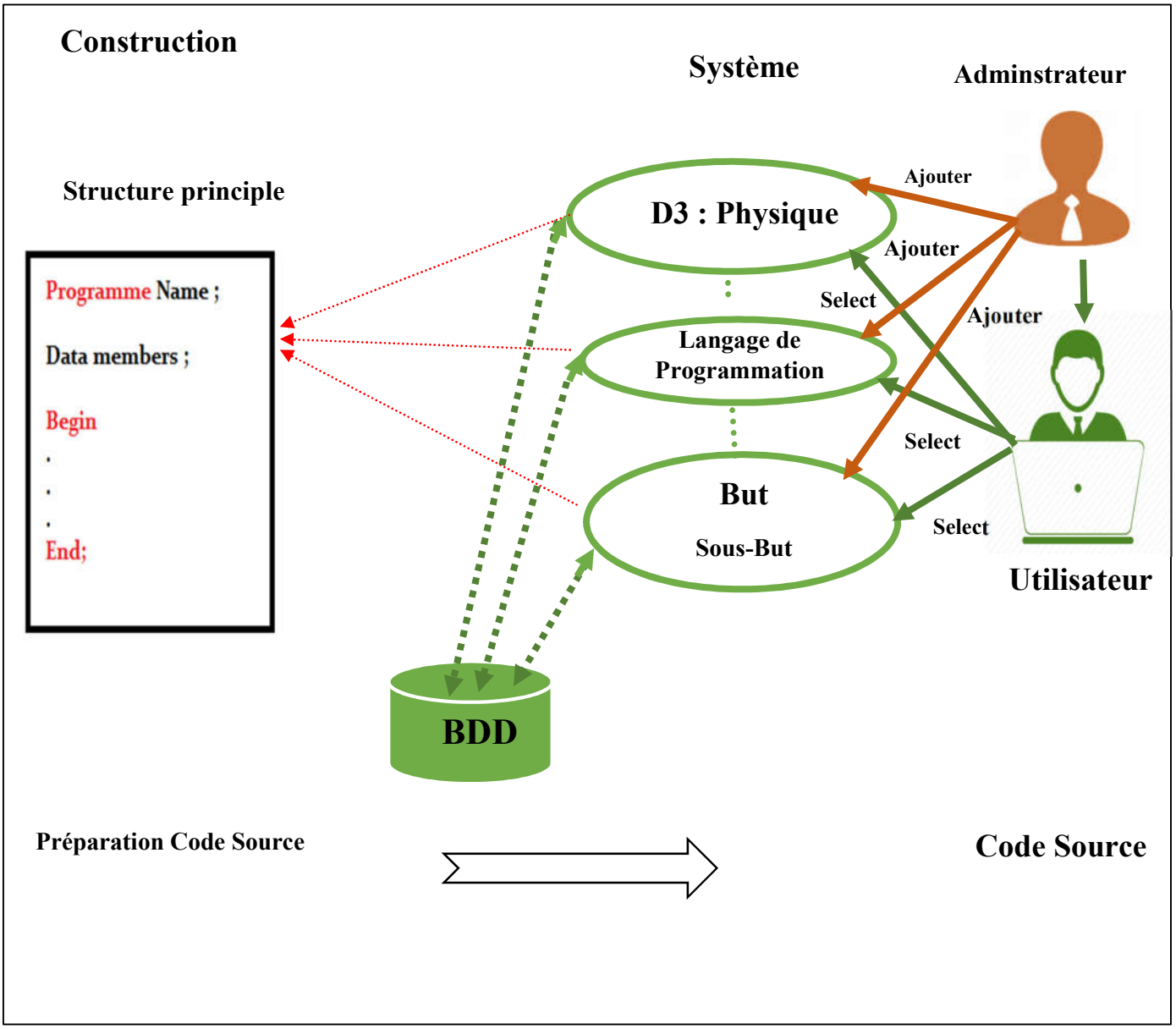
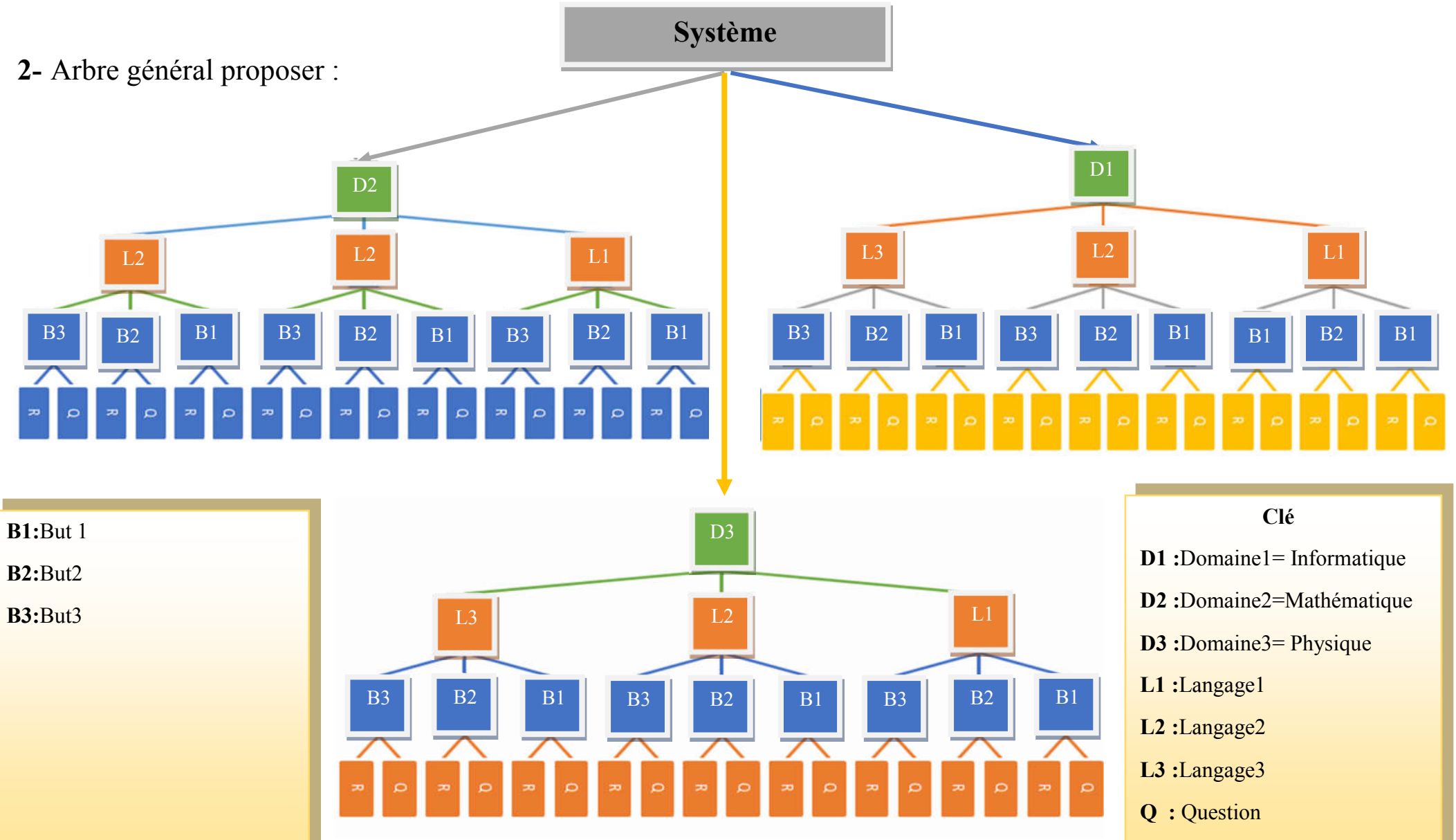


Figure 08 : Architecture general propose

2- Arbre général proposer :



B1: But 1

B2: But2

B3: But3

2/2

Clé

D1 : Domaine1= Informatique

D2 : Domaine2=Mathématique

D3 : Domaine3= Physique

L1 : Langage1

L2 : Langage2

L3 : Langage3

Q : Question

R :Réponse **1/2**

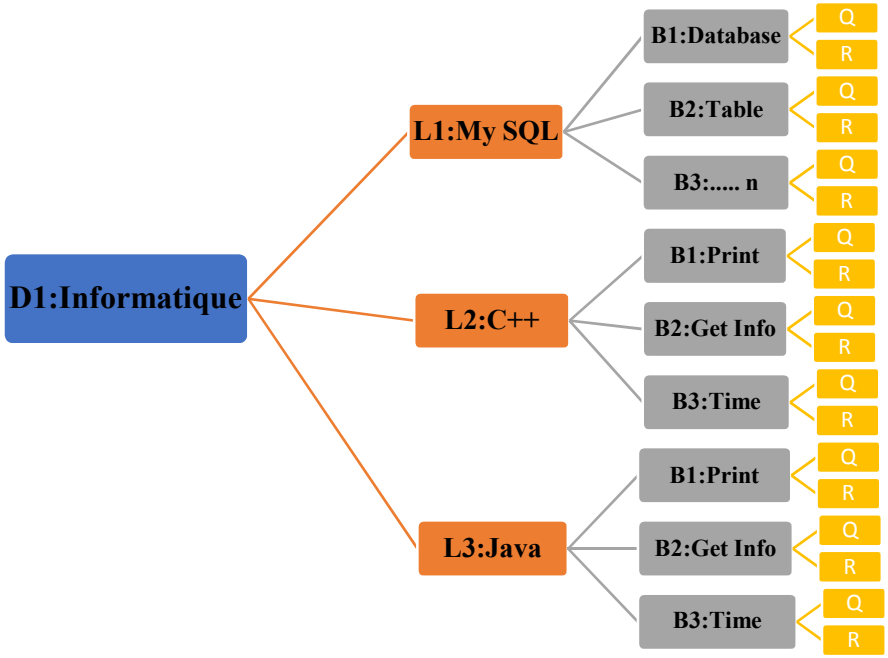
Schema 06 : Arbre général proposer

3- Domaine informatique :

Si l'utilisateur a sélectionné « **Domaine Informatique** »

3-1 Arbre :

Le schéma suivant illustre les opérations dépendantes dans domaine informatique .



Schema 07: Arbre Domaine informatique

3-2 Architecture :

3-2-1 Architecture langage de programmation

Figure 10 représente l'architecture langage de programmation d'un domaine informatique

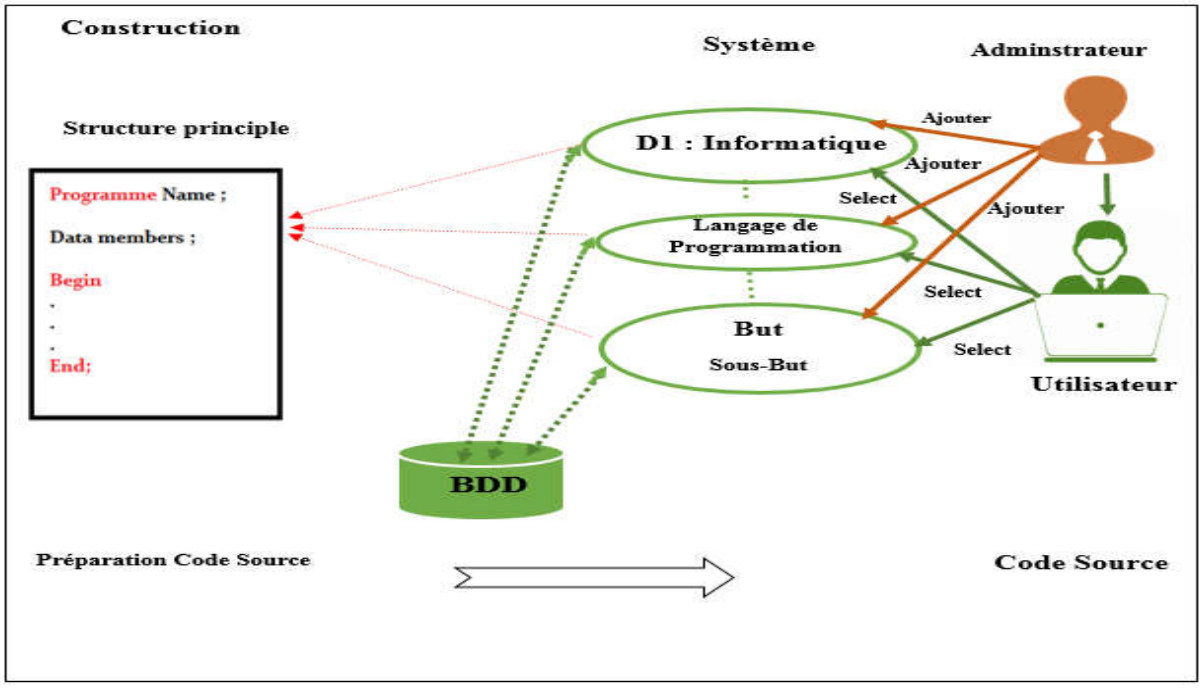


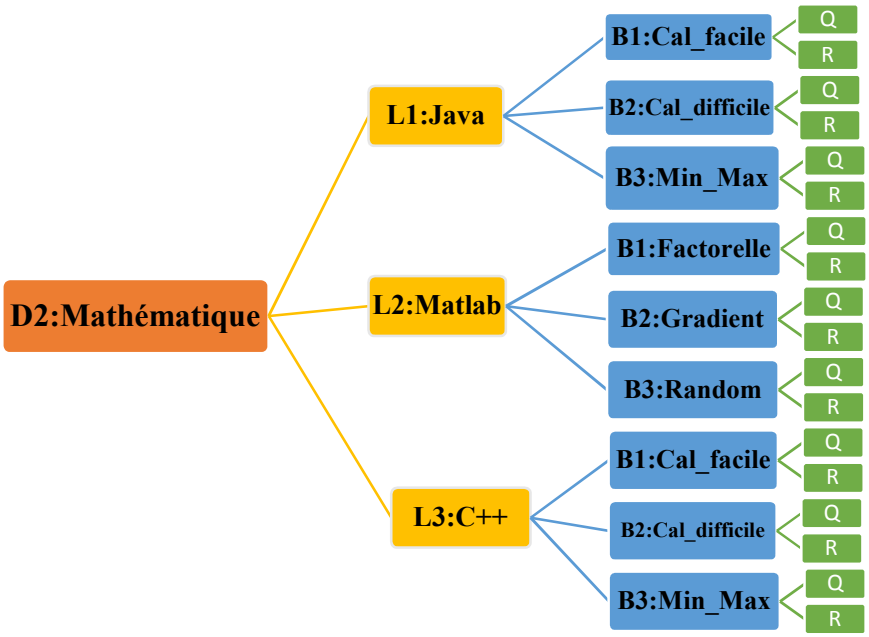
Figure 09 : Architecture le Domaine Informatique d'un Langage

4- Domaine mathématique

Si l'utilisateur a sélectionné « **Domaine Mathématique**»

4-1 Arbre :

Le schéma suivant illustre les opérations dépendantes dans le domaine mathématique



Schema 08: Arbre Domaine mathématique

4-2 Architecture :

4-2-1 Architecture Langage Programmation

Figure 11 représente l'architecture langage de programmation d'un domaine mathématique

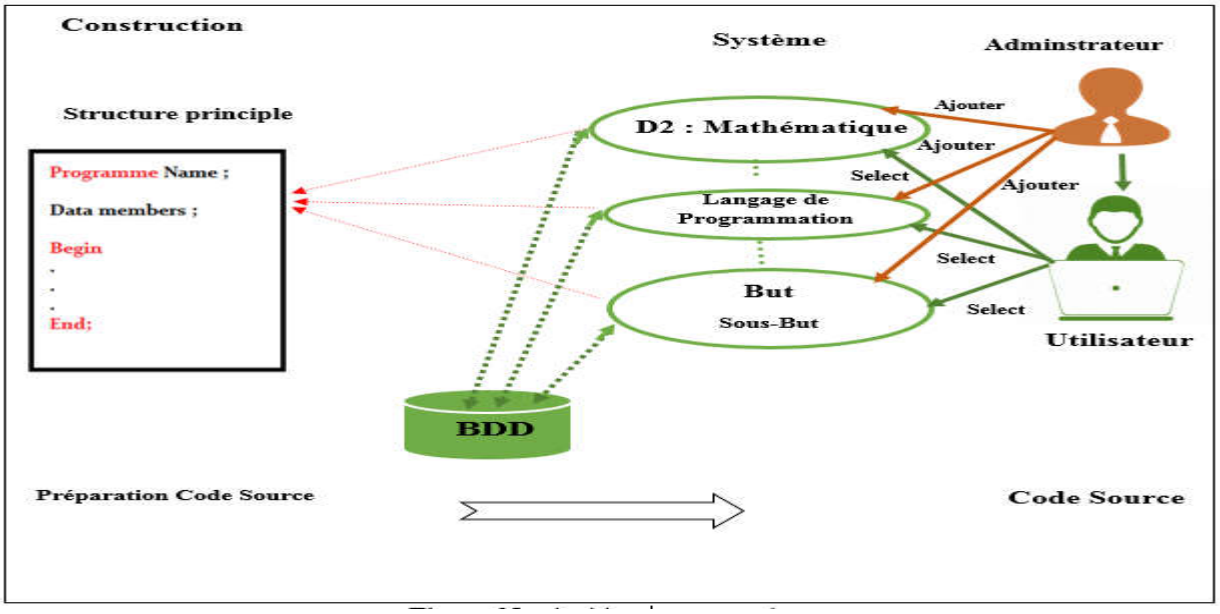


Figure 10 : Architecture le domaine mathématique d'un langage programmation

5- Domaine physique :

Si l'utilisateur a sélectionné « **Domaine Physique** ».

5-1 Arbre :

Le schéma suivant illustre les opérations dépendantes dans le domaine physique .

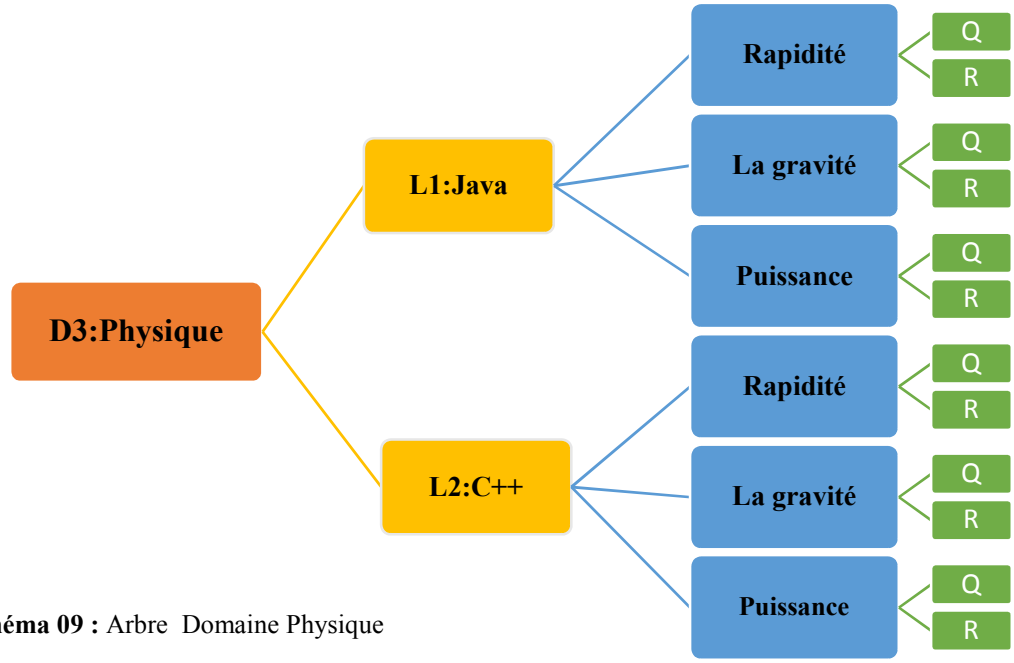


Schéma 09 : Arbre Domaine Physique

5-2 Architecture

5-2-1 Architecture Langage Programmation :

Figure 11 représente l'architecture langage de programmation d'un domaine physique.

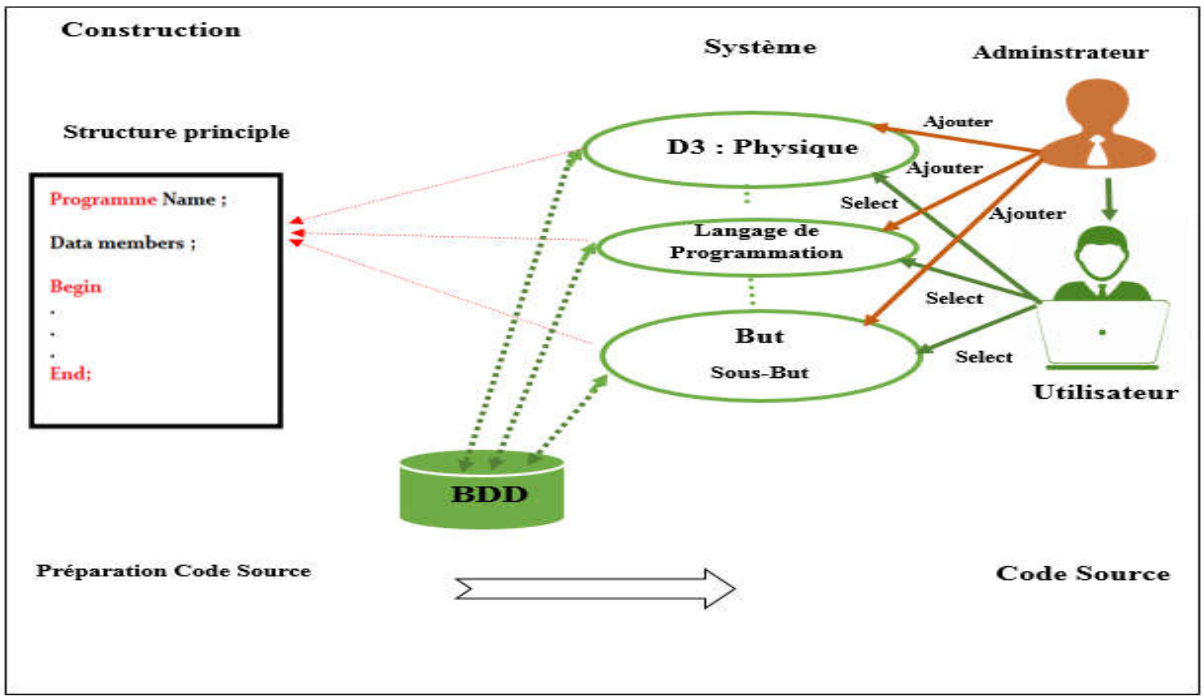


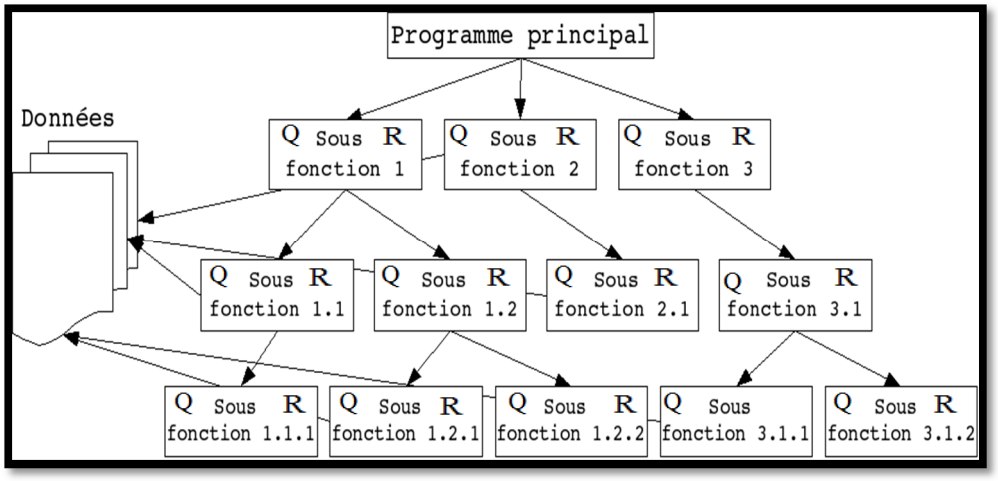
Figure 11 : Architecture le Domaine Physique d'un Langage Programmation

II L'approches construction

Aujourd'hui, en programmation, il existe deux principaux modèles de représentation du monde :

1- Le modèle fonctionnel :

Dans une approche fonctionnelle, les programmes sont composés d'une série de fonctions, qui assurent ensemble certains services. Il s'agit d'une approche logique, cohérente et intuitive de la programmation. Voir le schéma 9 suivant :



Schema 10 : Représentation graphique d'une approche fonctionnelle.

Alors pours cette approche , les questions proposées par notre systeme sont des question autour les fonctions du programme (l'application) cible.

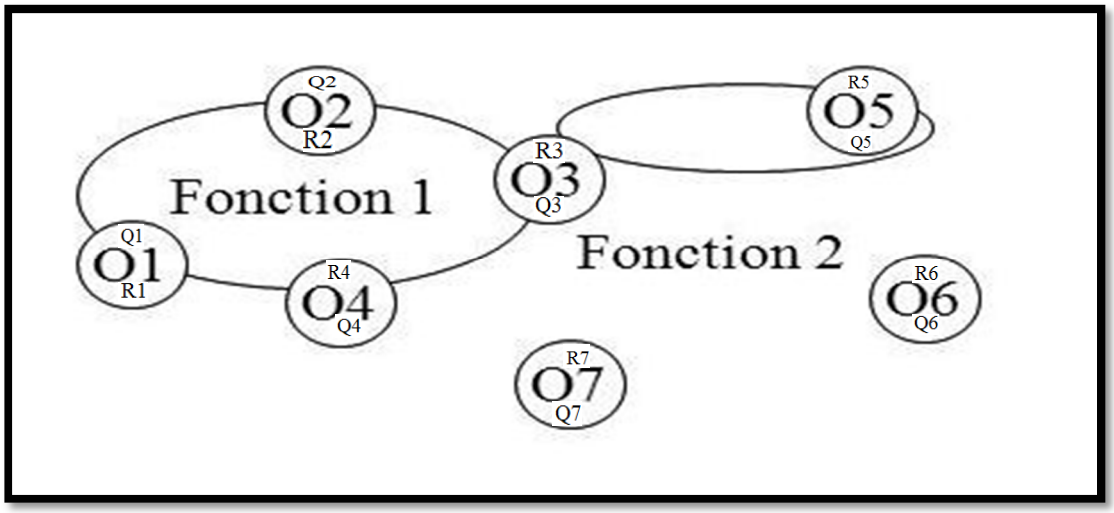
Cette approche a un avantage certain appelé la factorisation des comportements (c'est à dire que pour créer une fonction d'une application, rien ne vous empêche d'utiliser un autre ensemble de fonctions (qui sont donc déjà écrites)).

Mais, l'approche fonctionnelle a aussi ses défauts, comme par exemple une maintenance complexe en cas d'évolution d'une application (une simple mise à jour de l'application à un point donné peut impacter en cascade sur d'autres fonctions de l'application). L'application sera alors retouchée dans sa globalité.

L'approche fonctionnelle n'est pas adaptée au développement d'applications qui évoluent sans cesse et dont la complexité croit continuellement (plusieurs dizaines de milliers de lignes de code).[9]

2- Le modèle objet

on identifie les objets manipulés par le système, avec leurs états et leurs comportements. Voir le schéma 10 suivant :



Schema 11 : Représentation graphique d'une approche objet.

Alors pours cette approche , les questions proposées par notre systeme sont des question autour les objets du programme (l'application) cible.

- la modélisation des objets de l'application (consiste à modéliser informatiquement un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques. Ces entités informatiques sont appelées objet)
- la modularité (la programmation modulaire permet la réutilisation du code, via l'écriture de librairies)
- la réutilisabilité
- l'extensibilité (pour une meilleure productivité des développeurs et une plus grande qualité des applications).

L'approche objet a été donc inventée pour faciliter l'évolution d'applications complexes. Elle apporte l'indépendance entre les programmes, les données et les procédures parce que les programmes peuvent partager les mêmes objets sans avoir à se connaître comme avec le mécanisme d'import/export.

L'approche objet a introduit indépendamment de tout langage de programmation à l'objet trois concepts de base : objet, classe et héritage entre classes. Les concepts objet et classe sont interdépendants, c'est-à-dire qu'un objet est une instance d'une classe et la classe décrit la structure et le comportement communs d'objets (ses instances).[9]

III Construction de code source :

1- Algorithme :

Un algorithme est une séquence bien définie d'opérations (calcul, manipulation de données, etc.) permettant d'accomplir une tache en un nombre fini de pas. En principe un algorithme est indépendant de toute implantation. Cependant dans la pratique de la programmation il s'avère indispensable de tenir compte des capacités du langage de programmation utilisé. La conception d'un algorithme passe par plusieurs étapes:

Analyse : définition du problème en terme de séquences d'opérations de calcul de stockage de données, etc; .

Conception : définition précise des données, des traitements et de leur séquencement ;

Implantation : traduction et réalisation de l'algorithme dans un langage précis;

Test : Vérification du bon fonctionnement de l'algorithme.[10]

2- Structure d'un algorithme

```
Algo test
'déclarations des constantes et variables
Début
'Opérations sur les données
Fin
```

Algo 02 : Structure d'un algorithme Généralement

3- Instruction :

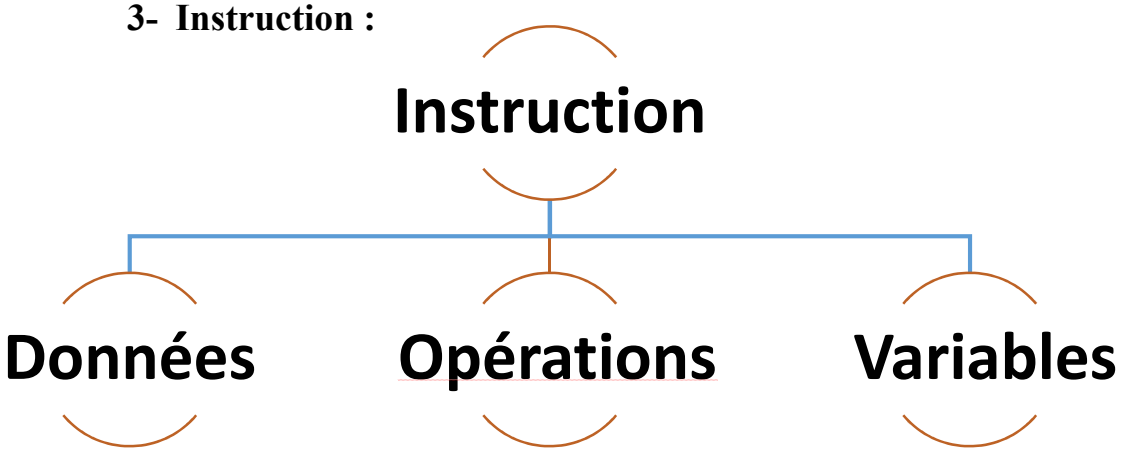


Schéma 12 : composants l'instruction

3-1 Données et types

Un algorithme manipule des données de différents types, par exemple :
Type entier

- ✓ Type réel
- ✓ Type chaîne de caractères. [10]

3-2 Opérations sur les données

Un opérateur est un signe qui relie deux valeurs, pour produire un résultat.

- ❖ **Opérateurs numériques :** Ce sont les quatre opérations arithmétiques :
  + addition
  - soustraction
  * multiplication
  / division
  ^ puissance

Opérateur alphanumérique : Cet opérateur permet de concaténer deux chaînes de caractères.[3]

3-3 Variables

Ceci se fait tout au début de l'algorithme, avant même les instructions proprement dites. C'est ce qu'on appelle la déclaration des variables. La déclaration se fait par la donnée du nom de la variable et du type de la variable.

- **Un nom** Le nom de la variable (l'étiquette de la boîte) obéit à des règles qui changent selon le langage utiliser.
- **Un type** :il existe troie type
  - ✓ Types numériquesclassiques
  - ✓ Type alphanumérique
  - ✓ Type booléen

- **Une valeur :** Une fois la variable définie, une **valeur** lui est **assignée**, ou**affectée.**
- **Une adresse** : C'est l'emplacement dans la mémoire de l'ordinateur, où est stockée la valeur de la variable.[3]

3-4 Instructions d'entrée/ sortie : Lire et Ecrire

- **Ecrire**

L'instruction Ecrire : **Ecrire** (élément1 ; élément2 ; …)

Ecrire permet d'écrire (d'« éditer ») la valeur de variable(s) et/ou de constante(s) sur un **périphérique de sortie**, en général l'**écran**.

Algo 03 : Instructions d'entrée/ sortie

- **Lire**

L'instruction Lire comporte deux syntaxes : **Lire** (variable) ou **Lire** (message ; variable)

Lire est une instruction d'affectation. Ce type d'affectation peut être fait à n'importe quel moment du déroulement du programme

Algo 04 : Instructions Lire

4- Algorithme général proposer :

Algorithme Système

Entrée : Utilisateur U, Systeme M
Domaine D, Langage L
But B, Nombre Domaine i
Question Q, Réponse R
Nombre Langage j, Nombre But r
Select réponse Selec
Sortie : Code Source CS

Début

SI **U** *(problème de code)***ALORS**
U : utilisateur Employerle système

// Utilisateur choisir Domaine
Switch ***(Domaine)*** *{*
Case D1:*(Langage1 ou Langage2 ou Langage j)* ***dans Domain1***
Case D2:*(Langage1 ou Langage2 ou Langage j)* ***dans Domain2***
Case Di:*(Langage1 ou Langage2 ou Langage j)* ***dans Domaine i***
}
// Utilisateur choisir Langage
Switch ***(Langage)*** *{*
Case L1:*(But 1 ou But 2 ou But 3)* ***dans Langage 1***
Case L2 : *(But 1 ou But 2 ou But 3)* ***dans Langage 2***
Case Lj *:(But 1 ou But 2 ou But r)* ***dans Langage j***
}
// Utilisateur choisir But
Switch ***(But)*** *{*
Case B1:*(Q 1 et Q2 ou Q 3)* ***dans But 1***
Case B2:*(Q 1 et Q2 ou Q 3)* ***dans But 2***
Case Br:*(Q 1 et Q2 ou Q 3)* ***dans But r***
}
Q : Systeme poser Question M(Q)
R : Réponses proposées U(R)
Selec :Utilisateur Selectionne des réponse U(selec)
Fin si
CS ← M(Q) +U(R)

Retourner code source par utilisateur

Fin

Algo 05 : Algorithme général proposer

IV- Conclusion

Dans cette partie, nous avons réalisé l'architecture de notre application. Cette architecture nous a permis de dégager conception qui sera exploité lors de l'implémentation. Ce architecture sera transformé en modèle physique de données qui fera l'objet du chapitre suivant.

Partie 02 : Modélisation Conceptuelle & Organisationnelle

I. Modélisation Conceptuelle

1. Introduction

Le Modèle conceptuel de données est une représentation statique du système d'information. Il a comme objectif de constituer une représentation claire et cohérente des données manipulées dans le système d'information. Cette section, sera présentée comme suit : nous commençons par le choix de la méthodologie de conception et justification. Ensuite nous identifions les acteurs et les diagrammes des cas d'utilisation, puis nous présentons le diagramme de classe, diagramme de collaboration et enfin les diagrammes d'état transition.

2. Choix de la méthodologie de conception :

Dans la cadre de notre projet, nous avons opté pour le langage UML comme une approche de conception. Ci-dessous, nous présentons ce langage puis nous justifions notre choix.

2.1 Présentation d'UML:

UML (Unified Modeling Language) est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait lui un langage universel. En plus UML est essentiellement un support de communication, qui facilite la représentation et la compréhension de solution objet. Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions. L'aspect de sa notation, limite l'ambigüité et les incompréhensions. UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues.[6]

2.2 Les Diagrammes d'UML :

UML définit 9 diagrammes. Ceux-ci permettent de visualiser et de manipuler les éléments dits "de modélisation". Chaque diagramme UML ci-dessous possède une structure précise.

- **Diagrammes d'activité :** représentation du comportement d'une opération en terme d'action .

- **Diagrammes de cas d'utilisation :** représentation des fonctions du système d'un point de vue de l'utilisateur .

- **Diagrammes de classes :** représentation de la structure statique en terme de classes et de relations .

- **Diagrammes de collaboration :** représentation spatiale des objets, des liens et des interactions .

- **Diagrammes de déploiement :** représentation du déploiement des composants sur les dispositifs Matériels .

- **Diagrammes d'états-transitions :** représentation du comportement d'une classe en terme d'état .

- **Diagrammes d'objet :** représentation des objets et de leurs relations, correspond à un diagramme de collaboration simplifié, sans représentation des envois de message ;

- **Diagrammes de séquence :** représentation temporelle des objets et de leurs interactions.[2]

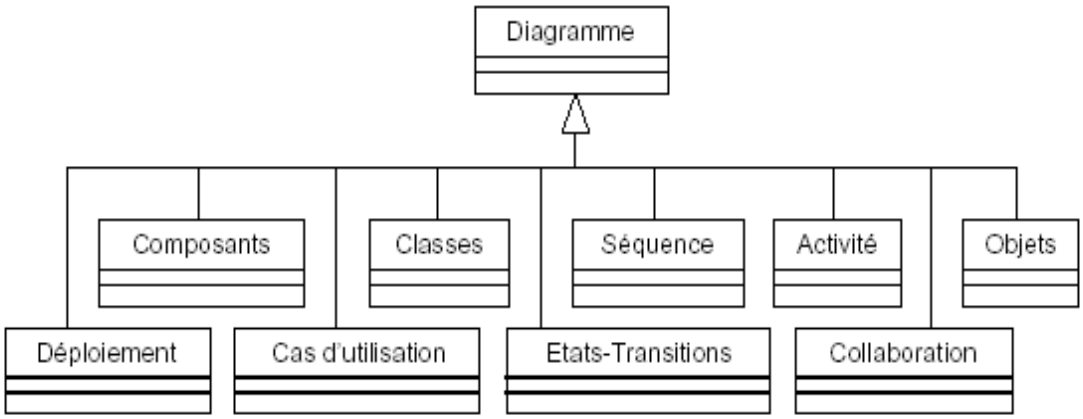


Schéma 05 : les 9 diagrammes UML

3. Diagramme des cas d'utilisation :

Les cas d'utilisation décrient un ensemble d'actions réalisées par le système, en réponse à une action d'un acteur.

3.1 Identification des acteurs

L'Utilisateur et l'administrateur sont les acteurs qui interagissent avec notre système.

- **L'Utilisateur** : utilise un outil pour obtenir le code
- **Administrateur** : c'est le responsable de l'administration du Outil Programmation.

3.2 Identification des cas d'utilisation

Nous décrivons pour chaque acteur les cas d'utilisation. On distingue les cas d'utilisation suivants :

L'Utilisateur : considéré comme une personne qui peut

- choisir un réponse : sélectionnée (domaine, opération, variable type, langage)
- Génération code.

Administrateur : personne qui a pour rôle principal de gérer le système .

- Ajouter (domaine, langages ,opérations, Codes souces)
- choisir un réponse : sélectionnée (domaine, opération, variable type, langage)
- Génération de codes source

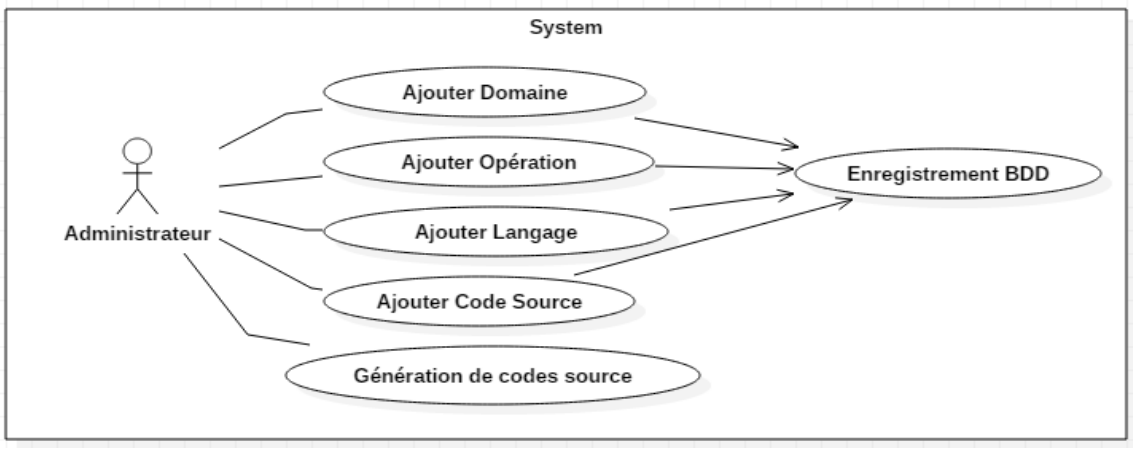


Figure 12 : Diagramme de cas d'utilisation pour Administrateur

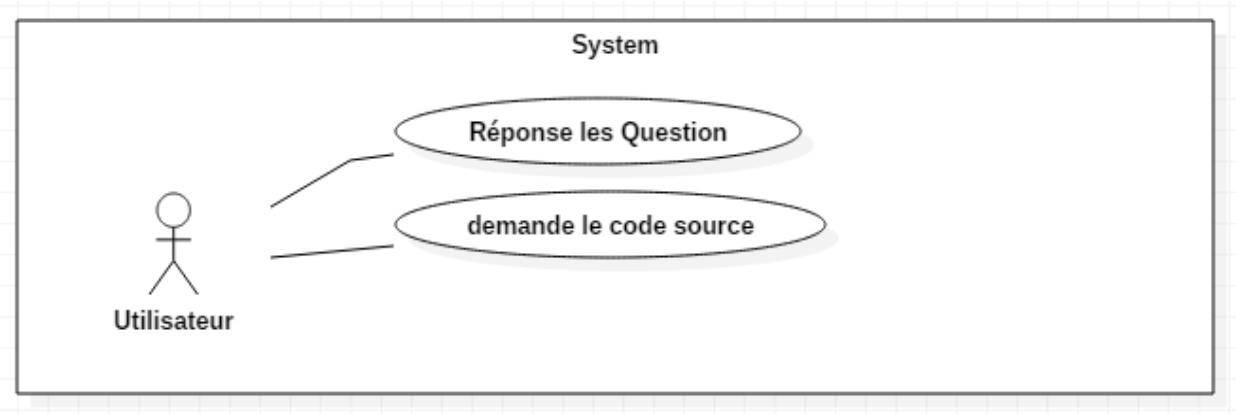


Figure 13 : Diagramme de cas d'utilisation pour l'utilisateur

3.3 Description textuelle des principaux cas d'utilisation

Dans le but de mieux comprendre notre système et les interactions avec les utilisateurs, dans cette partie nous allons détailler les scenarios de principaux cas d'utilisation

CU1: Ajouter les question
Acteurs : Administrateur
Résumé: le but de cette action pour trouver code source.
Scénario nominal
DESCRIPTION DU SCENARIO NOMINAL **« DEBUT»** **01 :** Ajouter nombreuse domaine **02 :** Ajouter nombreuse opération **03 :** Ajouter nombreuse langage **« FIN»**

CU2: génération code source
Acteurs : Administrateur
Pré-Condition : l'utilisateur sélectionner tous les réponse parmi les réponse proposes par le outil programmation
But : admin générer un code source à utilisateur
Scénario nominal
DESCRIPTION DU SCENARIO NOMINAL **« DEBUT»** **01 :** l'utilisateur sélectionner :domaine ,opération ,langage ….

02 : parmi les réponses de utilisateur l'Administrateur qui génération le code source **« FIN»**

CU3: Sélection de Réponses
Acteurs : Utilisateur
But : choisir le réponse de la proposition
Résumé : l'utilisateur sélectionné une réponse parmi les réponse proposer par le système
Scénario nominal
DESCRIPTION DU SCENARIO NOMINAL « DEBUT» 01 : le système poser de nombreuses question et des réponses possibles 02 : l'utilisateur sélectionné une réponse . « FIN»

CU4: Enregistrement BDD
Acteurs : Administrateur
Résumé: objectif cette action Save les données qui insert par Adminstrateur .
Scénario nominal
DESCRIPTION DU SCENARIO NOMINAL **« DEBUT»** **01 :** Admin qui ajoute :(Domaine,Opération ,Variable ,Langage) **02 :** Enregistrement des ajouts dans BDD **03 :** transformation les donneé à forme code source **« FIN»**

4. Modélisation conceptuelle des données

La modélisation conceptuelle des données permet de dégager l'ensemble des données manipulées en vue d'élaborer le diagramme de classes. En effet, ce dernier donne une vue statique du système. Il décrit les types et les objets du système. Il s'agit donc d'une représentation des données du champ de l'étude ainsi que le lien sémantique entre ces données, facilement compréhensible, permettant de décrire le système d'information à l'aide des concepts proposés par le modèle UML.

4.1 Dictionnaire des donnése

Le tableau ci-dessous représente la liste des attributs composants toutes les classes notre système ainsi que leur description.

Classe	Champs	Description
Domaine	ID	Identifiant de domaine
	Name	Name le domaine
Langage	ID	Identifiant du langage
	Langage	Name de langage
Sujet	ID	Identifiant de sujet
	Type	Type de opération
	Règle	Règle de opération
	Numéro Variables	Numéro Variables de opération

Table 04 : Dictionnaire des données

4.2 Représentation des classes

Les objets constituent la base de l'architecture des applications, ils peuvent être réutilisés à travers des domaines d'application ou encore être identifiés et dérivés directement

des cas d'utilisation ou des domaines d'application. Une classe est composée :

- **Attributs** : représentant des données dont les valeurs représentent l'état de l'objet.
- La **méthode** : il s'agit des opérations applicables aux objets.

Après avoir dégagé le dictionnaire de données épuré, nous pouvons dégager les classes ainsi

leurs méthodes et leurs attributs qui sont présentés dans le tableau suivant :

Classe	Attributs	Type /Taille	Méthodes
Domaine	ID	Entier long	• Choisir Domaine ( )
	Name	Varchar(50)	• Ajouter Domaine( )
Langage	ID	Entier long	• Choisir Langage( )
	Langage	Varchar(50)	
Sujet	ID	Entier long	• Choisir opération ( )
	Type	Varchar(50)	• Choisir type ( )
	Règle	Varchar(50)	• Choisir Numéro Variables ( )
	Numéro Variables	Varchar(50)	

Table 05 : Liste des classes

4.3 Diagramme de classes

La figure ci-dessous récapitule les tableaux précédents dans un diagramme de classes qui Contient toutes les informations telles que les classes, les méthodes, les associations et les propriétés.

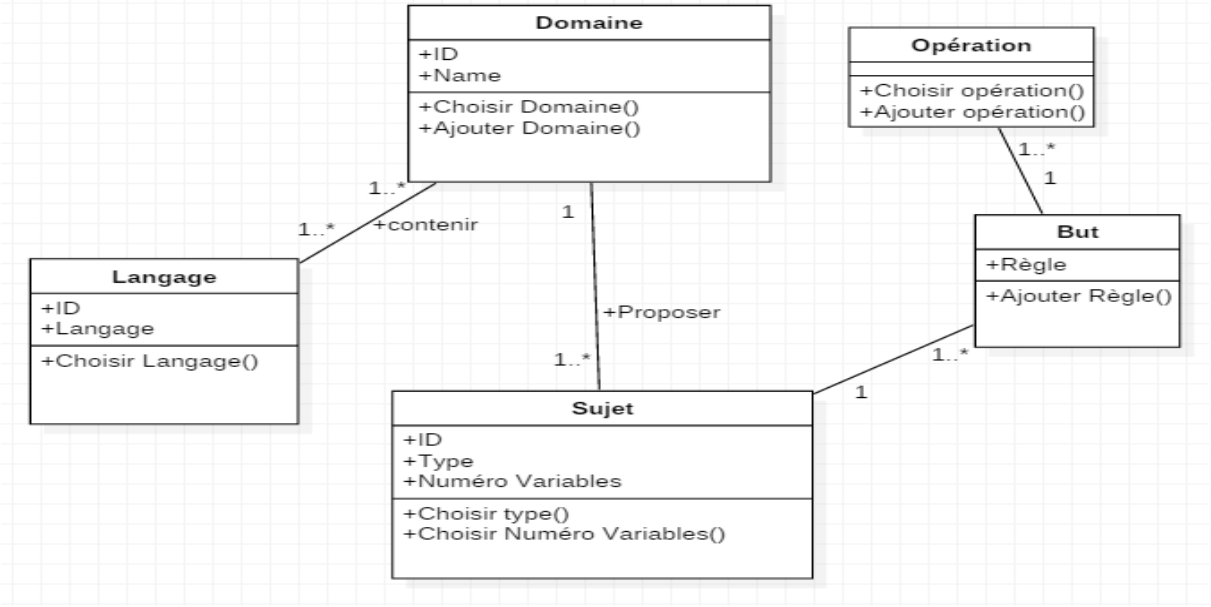


Figure 14 : Diagramme de classes

4.4 Diagrammes de séquences

Les diagrammes de séquences représentent les interactions entre les objets en indiquant la chronologie des séquences.

Le diagramme de séquence «Ajouter les question » présente le séquencement des interactions entre Administrateur , l'interface de système.

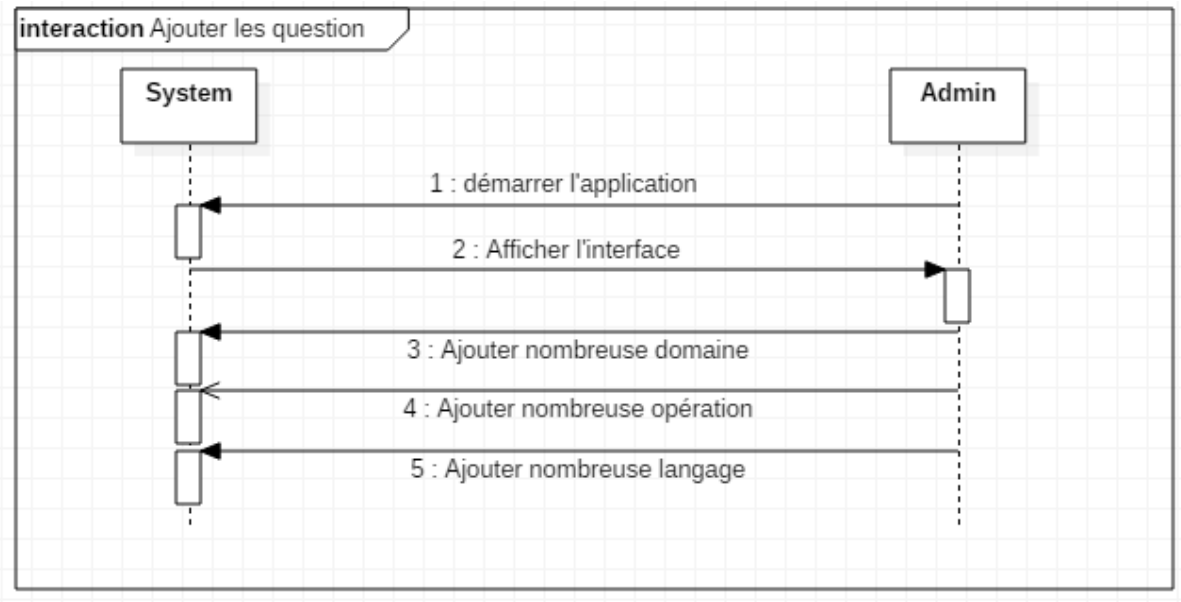


Figure 15 : diagramme de séquence «Ajouter les question »

- Le diagramme de séquence «génération le code source »

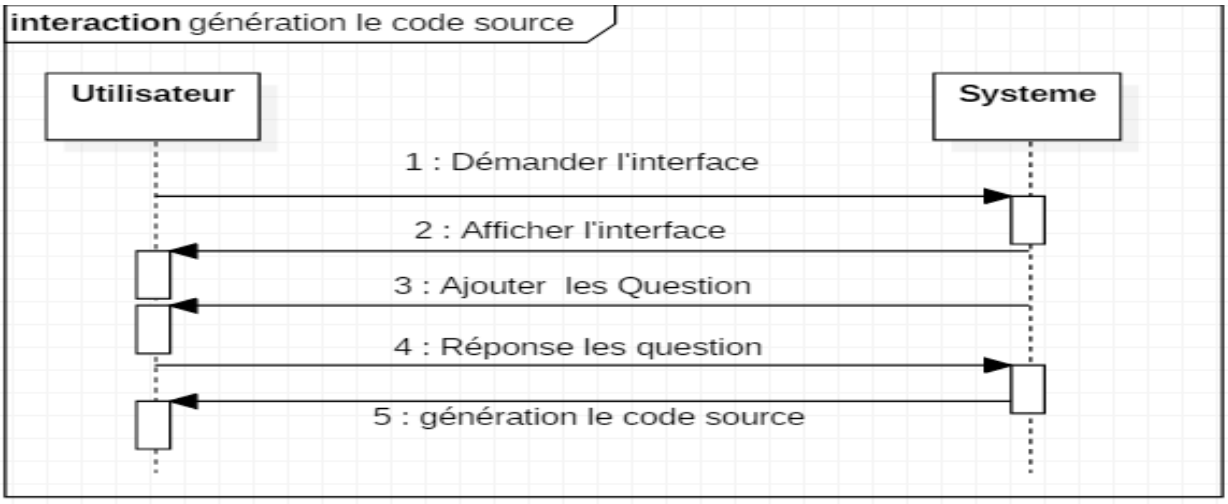


Figure 16 : diagramme de séquence «génération le code source»

- Le diagramme de séquence « Sélection de Réponses»

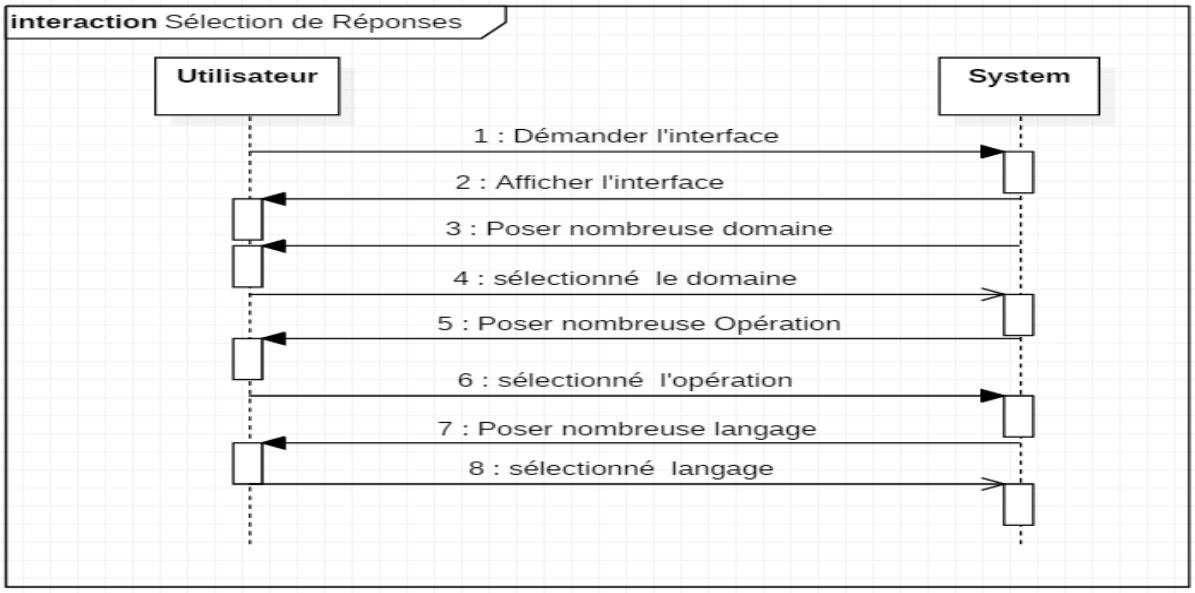


Figure 17 :diagramme de séquence « Sélection de Réponses»

- Le diagramme de séquence «Enregistrement BDD»

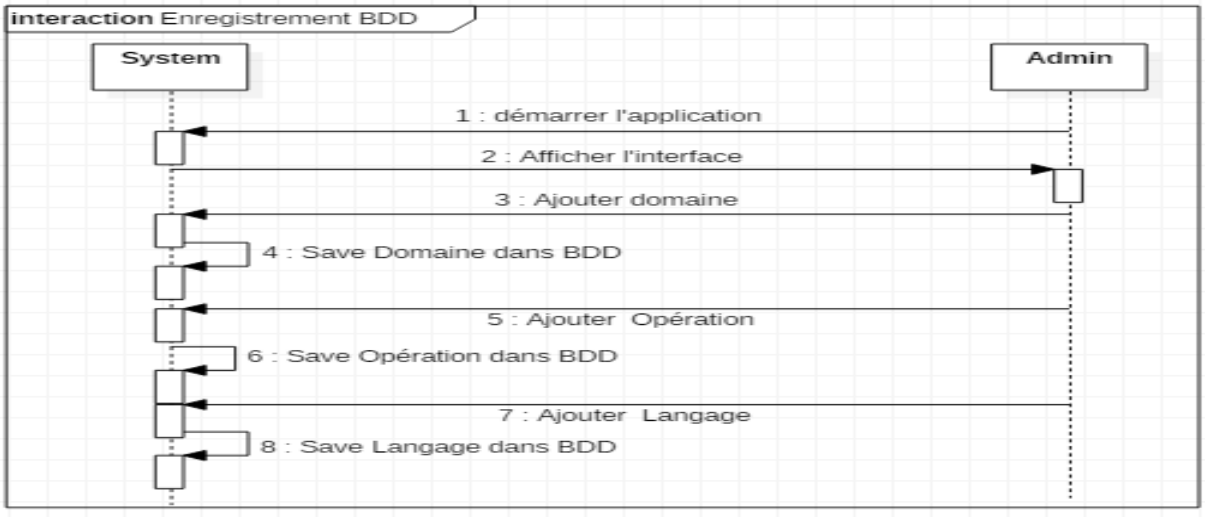


Figure 18 :diagramme de séquence « Enregistrement BDD»

II. Modélisation organisationnelle et logique

Dans la section précédente nous avons proposé une modélisation conceptuelle des données et des traitements en se basant sur l'approche objet UML qui représente l'état de l'art des langages de modélisation objet, il permet de modéliser la structure et le comportement d'un système indépendamment de toute méthode ou langage de programmation.

1. Diagramme d'activités:

Les diagrammes d'activités sont relativement proches des diagrammes d'états transitions dans leur présentation, mais leur interprétation est différente.

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles.

Dans ce qui suit, nous présentons notre diagramme d'activité pour créer le outil programmation

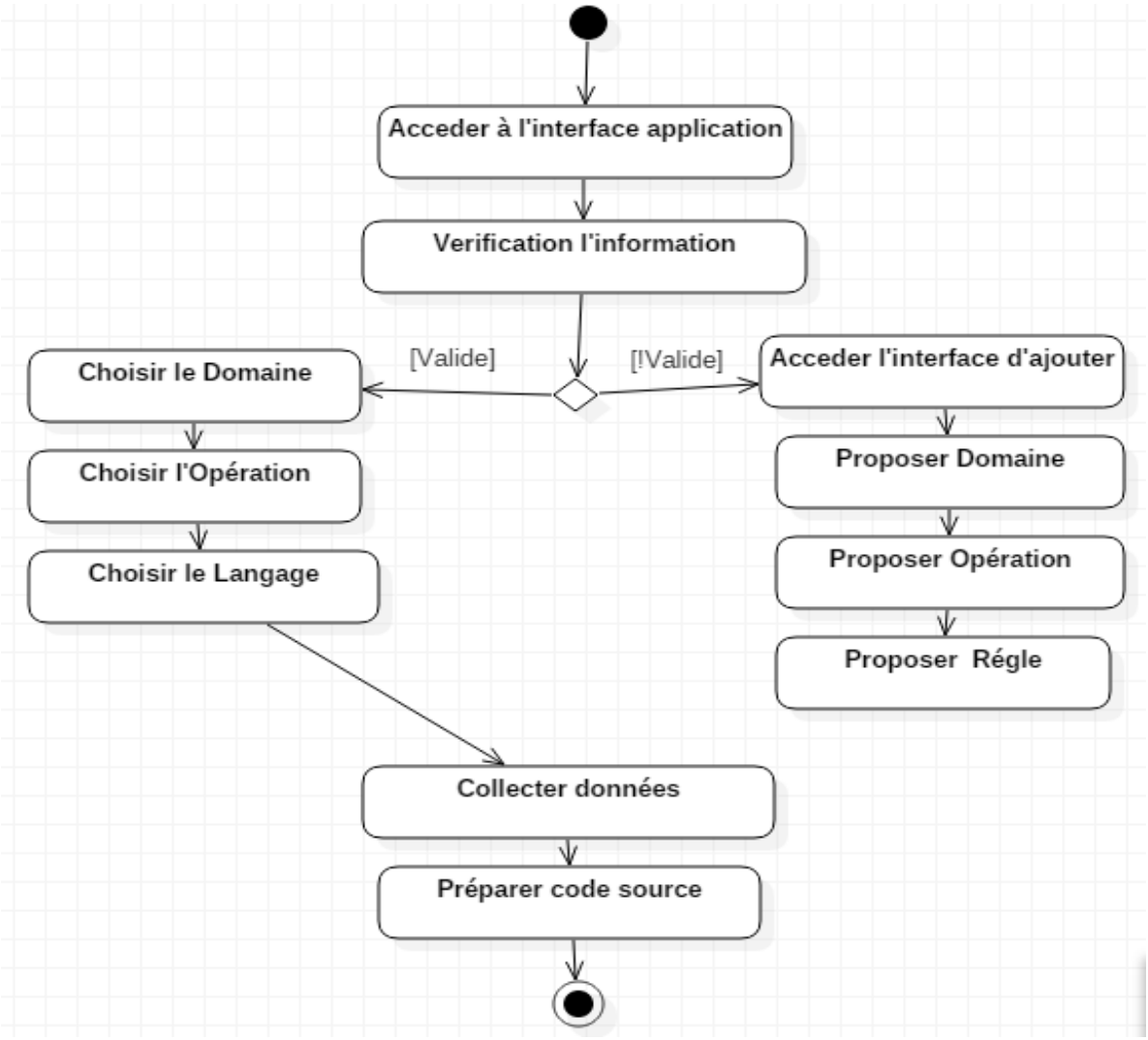


Figure 19 : diagrammes d'activités

III. Conclusion :

Dans cette partie, nous avons réalisé la modélisation organisationnelle et logique de notre application. Cette modélisation nous a permis de dégager le modèle logique des données qui sera exploité lors de l'implémentation. Ce modèle sera transformé en modèle physique de données qui fera l'objet du chapitre suivant.

Chapitre04

Réalisation

1- Introduction

Dans ce chapitre, nous allons aborder le côté pratique de notre projet. Nous définirons les étapes de réalisation, et la mise en œuvre de notre système l'outil de programmation automatique. Nous commenceront par une brève présentation des outils de développement que nous avons utilisé, et nous terminerons par l'implémentation de notre système.

2- Environnement du développement:

Avant de commencer l'implémentation de l'architecture conceptuelle de notre système, nous allons tout d'abord spécifier les outils utilisés qui nous ont semblés être un bon choix de par les avantages qu'ils offrent.

2-1 Choix du système d'exploitation (Windows)

Notre application a été développée sous le système d'exploitation Windows 7, mais comme elle est développée en langage C++, elle peut être intégrée dans n'importe quel autre système d'exploitation supportant la machine virtuelle C++ (Windows 98/00, Linux, …).

3- Outils de développement

3-1 Choix du langage de programmation(C++):

Le langage C++ est un langage de programmation crée par Bjarne Stroustrup et normalisé en 1998. Le ++ est l'opérateur d'incrémentation (qui fais +1) : le C++ est donc une amélioration de l'ancestral langage C.

Il apporte notamment la programmation orientée objet (qui est l'apport majeur), la gestion des exceptions, la gestion des références (remplaçant partiellement l'usage quelque peu délicat des pointeurs), la surcharge des opérateurs et les templates (liste non exhaustive). Enfin, une rétrocompatiblité a été gardée ; les programmes en C compilent sans difficulté avec un compilateur C++. [12]

Structure code general :

```
#include <iostream>

int main()
{
std::cout << "Hello, new world!" << std::endl;
}
```

Algo 06 : Structure code general d'un porogramme c++

3-2 Choix d'éditeur (Microsoft Visual Studio):

Microsoft visual studio est une suite de logiciels de développement pour windows conçue par microsoft. Visual studio est un ensemble complet d'outils de développement permettant de générer des applications web asp.net, des services web xml, des applications bureautiques et des applications mobiles. visual basic, visual c++, visual c# utilisent tous le même environnement de développement intégré (ide), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages.[18]

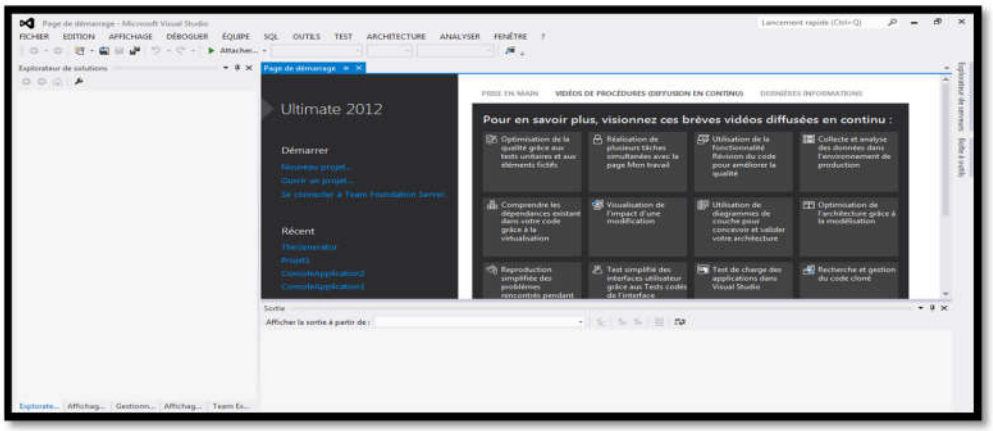


Figure 19 : Interface de Microsoft Visual Studio

Nous avons mise en œuvre de notre travaille par Windows application MFC C++Win 32

3-3 Choix BDD (MS Access Database) :

Microsoft Access (officiellement Microsoft Office Access) est un SGBD relationnel édité par Microsoft. Il fait partie de la suite bureautique MS Office Pro.MS Access est composé de plusieurs programmes : le moteur de base de données Microsoft Jet, un éditeur graphique, une interface de type Query by Exemple pour manipuler les bases de données, et le langage de programmation Visual Basic for Applications.[5]

Les avantages d'Access	les Inconvénients d'Access
Pouvoir questionner une base de données.	Je désire développer une application : je dois alors maitriser la conceptualisation des données
Possibilité de verrouiller certaines parties de l'application en définissant des autorisations	Je désire travailler simplement sur une base de données : formation obligatoire.

Table 06 : Table comparaison avec les avantages entre les inconvénients d'access

3-3-1 Création des bases de données

En **premièrement**, nous avons créé une base de données à l'aide du programme ms access .

nom de la base de données : system.

En **second lieu** ,nous avons créé trois tables.

En **finalement:** le tableau 04 suivant explique bien le contenu de base de données

3-3-1-1 Modélisation physique des données

Nom Table	Nom Du Champs	Type et Taille
Domaine	ID	Number
	Name	varchar(50)
Langauge	ID	Number
	Langauge	varchar(50)
Subject	ID	Number
	Subject	varchar(50)
	Type	varchar(50)
	Rule	varchar(50)
	VariablesNumber	varchar(50)

Table 07 : Modélisation physique des données

3-3-2 Développement de l'application

Pour créer un programme constructif nous avons une étape qui permis l'établissement de connexion entre Application et Base Données (MS Accès) , Pour terminer cette opération on utilise la connexion suivante :

3-3-2-1 Les méthodes de connexion de Visual Studio vers MS Access

La connexion à la base de données MS Access est rendue possible par l'utilisation de la classe Microsoft Access Driver et de l'interface Connection.

La connexion s'effectue par la méthode OpenConnection , Il est préférable de mettre l'ouverture de connexion dans un Try… Catch .

Illustration dans le code suivant :

```
CString sDriver = L"MICROSOFT ACCESS DRIVER (*.mdb)";
CString sFile = L".\\Data\\system.mdb";
//setup main database variables
                    sDsn.Format(L"ODBC;DRIVER={%s};DSN='';DBQ=%s",sDriver,sFile);// Build ODBC
connection string
                    g_OutputText = L"" ;
        VarNum = 1 ;
                    //Open connection over the database
                    OpenConnection();
```

Algo 07 : Partie d'un programme expliquer (déclaration variable Database)

- **OpenConnection()**

```
void OpenConnection()
{
	TRY
	{
		// Open the database
		database.Open(NULL,false,false,sDsn);
	}
	CATCH(CDBException, e)
	{
			// If a database exception occured, show error msg
			AfxMessageBox("Database error: " + e->m_strError);
	}
	END_CATCH;
}
```

Algo 08 : Partie d'un programme expliqué (Ouvrir la connexion)

3-3-2-2 Fermeture d'une connexion

La connexion est fermée en appliquant la méthode close() à tous les objets Connection ouvert.

Illustration dans le code suivant :

```
void Disconnect()
{
	//close
	recset.Close();
}
```

Algo 09 : Partie d'un programme expliquer (Fermeture la connexion)

4-Présentation de l'application

4-1 Structure générale:

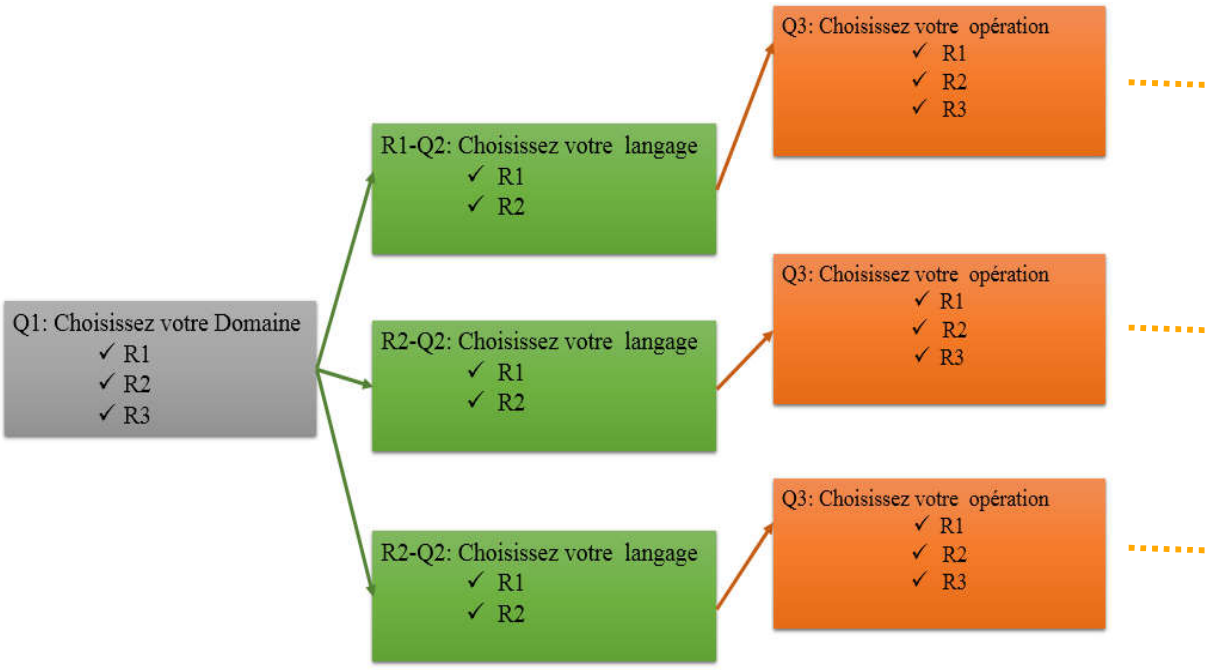


Schéma 14 : Structure général d'application

4-2 Les interfaces de l'application:

Notre programme a une seule interface qui consiste dans un espace contenant un système de question/réponse. Il s'agit de nombreuse question que pose le programme et qui permet de former le code. Ces questions consistent dans un ensemble des domaines ou plutôt des langages de programmation ou …ou…L'utilisateur doit en choisir un.

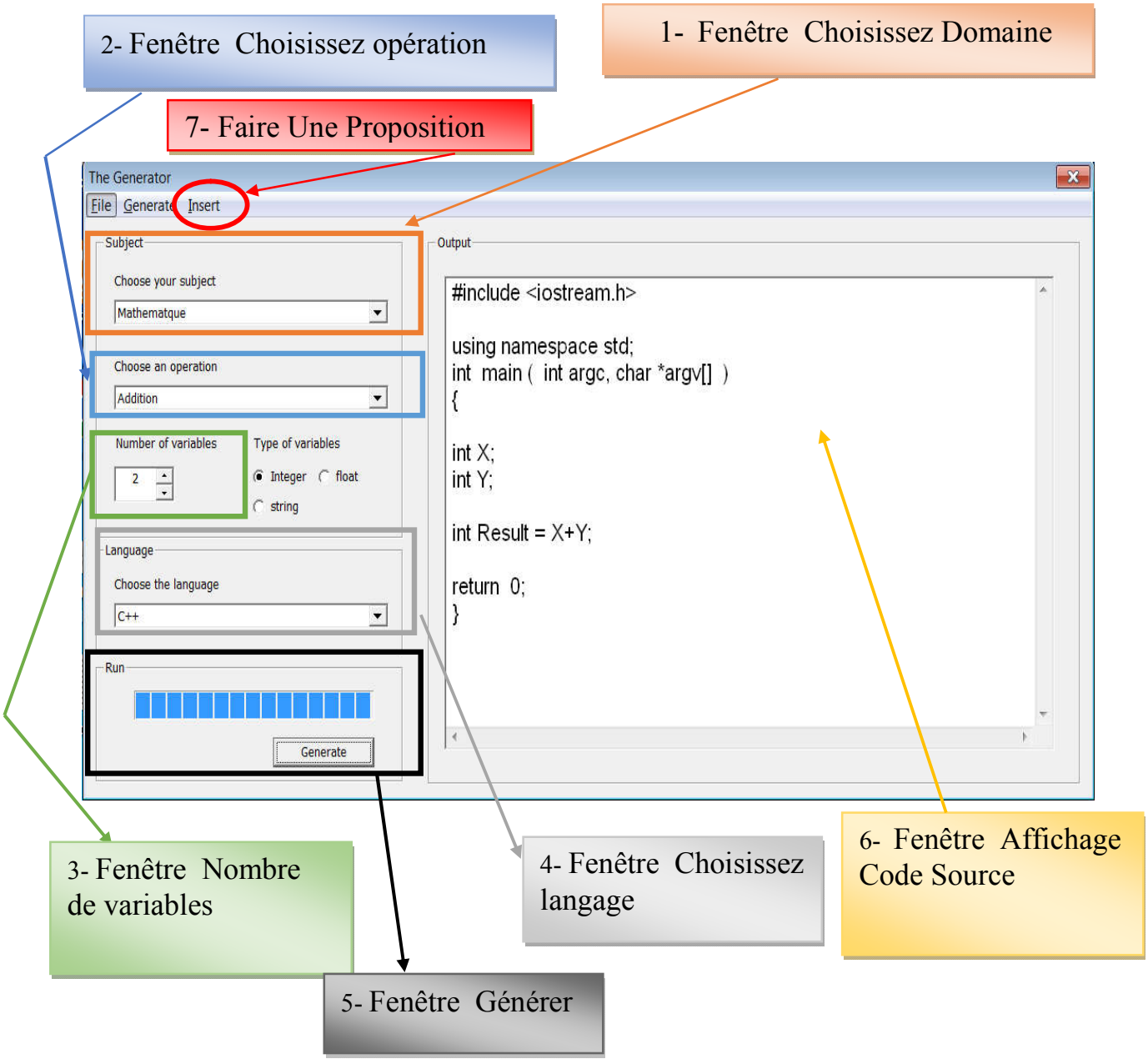


Figure 20 : Les interfaces de l'application

4-2-1 Fenêtre Choisissez Domaine

Cette partie permet de choisir l'un des domaines proposés, l'utilisateur doit en désigner un qui convient à son besoin où il doit préciser le domaine du code de son propre programme que nous avons formulé sous forme de :

- Programme (poser des questions) : cela propose de nombreux domaines tels : Informatique, mathématique, physique, etc.
- L'utilisateur (choix de réponse) où il choisit l'un des domaines recherché selon le besoin.

Le figure 20 ci-dessous présente Partie Domaine :

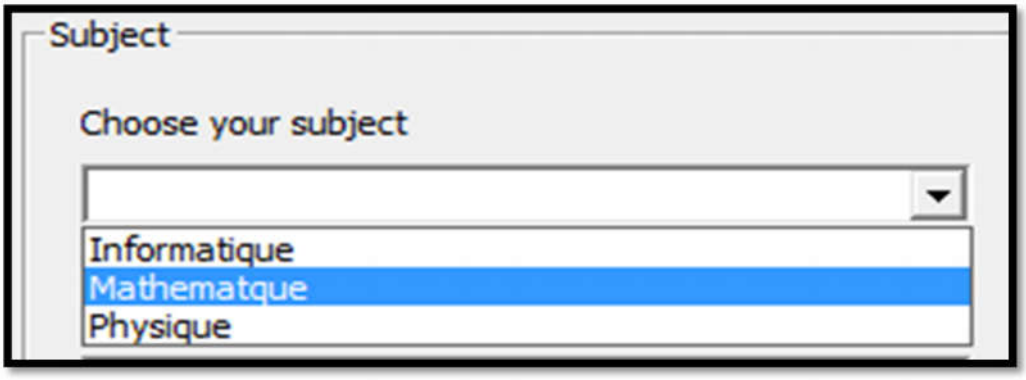


Figure 21 : Les Fenêtre Choisissez Domaine

4-2-2 Fenêtre Choisissez opération

Nous abordons cette section immédiatement après avoir sélectionné la zone où chacune des zones précédente a un grand nombre d'opérations particulières qui s'inscrivent dans 'un domaine particulier ; de telle sorte que chaque zone a ses propres opérations. Nous les avons reliés en utilisant la base de données que nous avons formulé sous forme de:

Lorsque l'utilisateur choisit une zone des zones prévues par le programme, les opérations changent automatiquement selon de modifications selon le domaine dont elles dépendent.

Le figure 21 ci-dessous présente d'une partie de l'opération

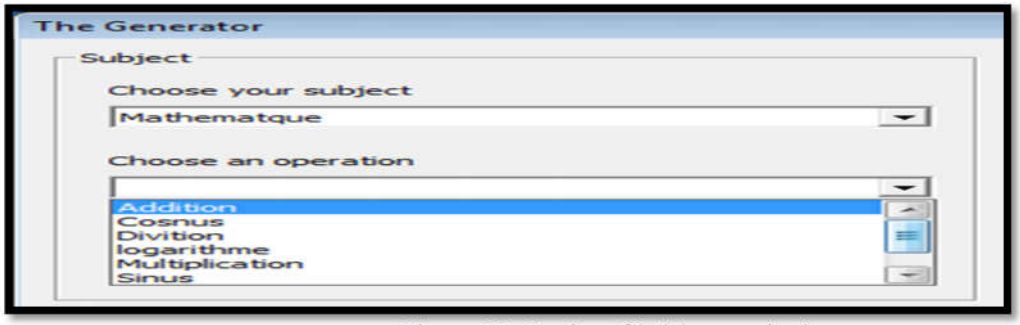


Figure 22 : Fenêtre Choisissez opération

Par exemple, le tableau suivant présente chaque domaine et les opération dont elles dépendent

Domaine	**Opération**	
Informatique	Get System Information	/
	Get Time Information	/
	Print	/
Mathématique	Calcul Facile	Addition
		Soustraction
		Multiplication
		Divition
	Calcul Difficile	Logarithme
		Cosnus
		Sinus
		Tangent
Physique	Velocity	/
	Gravity	/
	Power	/
	Print	/

Table 08 : Tableau Choisissez opération

4-2-3 Fenêtre Nombre de variables

Cette partie concerne seulement le domaine mathématique, et en particulier les opérations de calcul où nous avons présenté à l'utilisateur un outil pour le choix d'un plus de variable pour l'opération de calcul. Ce qui veut dire que : s'il choisit le domaine de mathématique, puis il choisit l'opération de calcul, il devra introduire le nombre des variables qu'il veut calculer dans son programme.

Le figure 22 ci-dessous présente Partie Selection Variable :

Figure 23 : Fenêtre Nombre de variables

4-2-4 Fenêtre Choisissez langage

Notre programme a deux types du langage : c++ و java. A partir du choix du langage par l'utilisateur, le programme génère le code en se basant sur les donnés précédentes. Le figure 23 ci-dessous présente Partie Langage

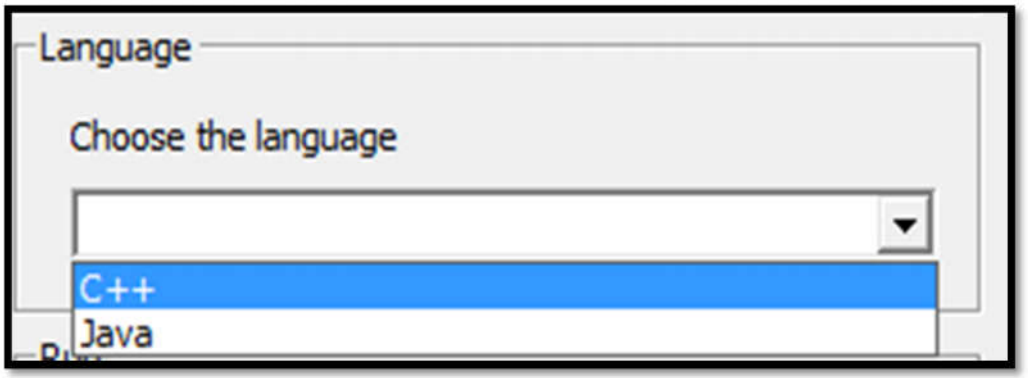


Figure 24 : Fenêtre Choisissez langage

4-2-5 Fenêtre ou Bouton Générer

On utilise ce bouton après terminer l'opération de question/réponse . ce qui veut dire que la rôle de ce bouton est d'indiquer la fin de l'opération question/réponse dans le programme. L'utilisateur doit donc l'utiliser pour lui apparaitre le code de son programme. Le document ci-dessous présente le bouton de génération. Le figure 24 ci-dessous présente Partie Générer d'un programme

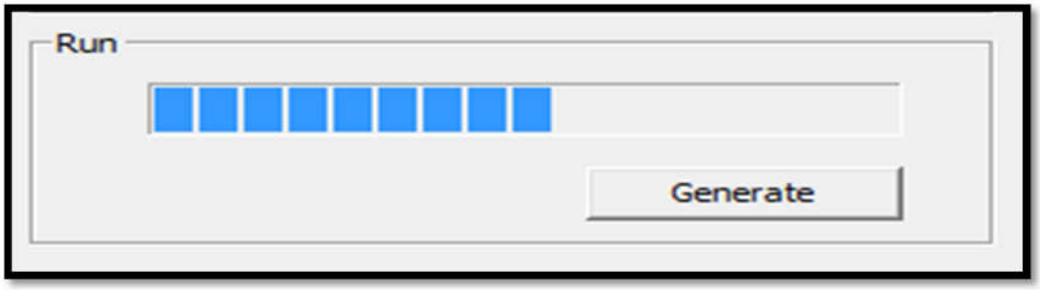


Figure 25 : Bouton Générer

4-2-6 Fenêtre Affichage Code Source

Zone de présentation de code du programme Il s'agit d'une zone blanche où l'application montre le code de programme complet prêt et exécutable contenant les données que l'utilisateur à déjà présentées. Le figure 25 ci-dessous présente Partie Affiche Code Source

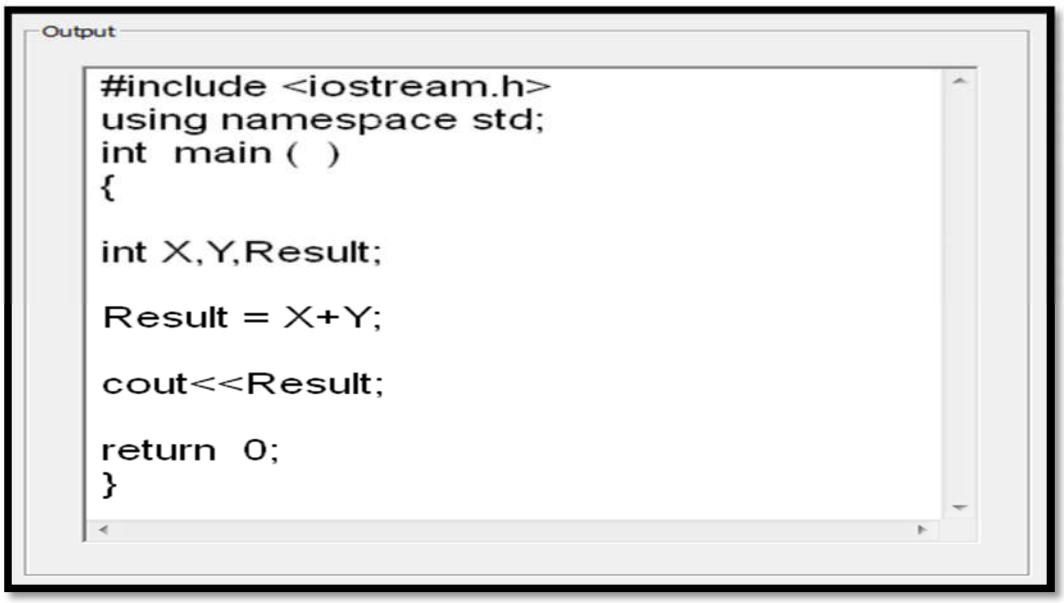


Figure 26 : Fenêtre Affichage Code Source

4-2-7 Faire Une Proposition :

Fenêtre d'insertion disponible dans la barre des tache dans notre application, comme indique sur la figure suivant :

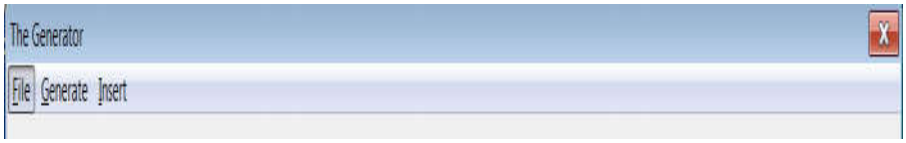


Figure 27: barre des tache dans application

L'utilisateur utilise si sa requête n'est pas disponible dans la proposition du système. Donc cet espace lui permet d'insérer ce dont il a besoins avec précision. En appuyant sur le bouton OK ceci aura pour effet d'ajouter sa proposition dans la réponse existant dans l'application. Enfin, il obtiendra un code source du programme .

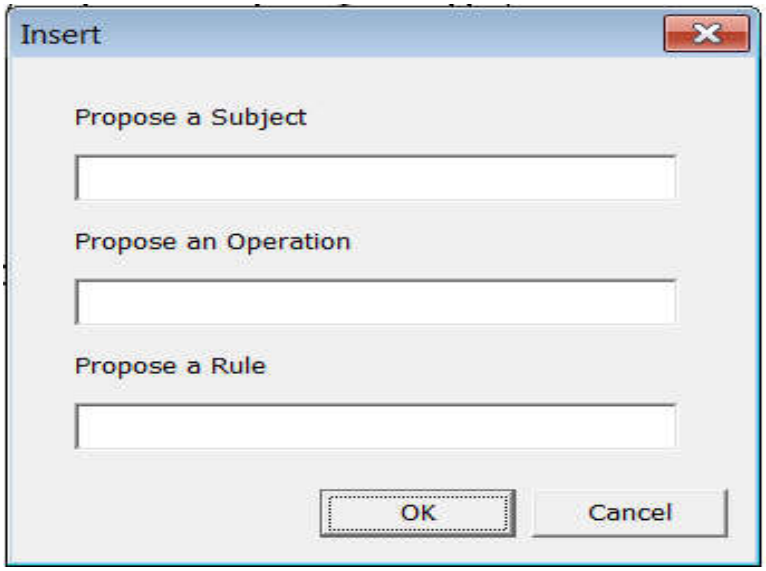


Figure 28 : Espace d'ajouter sa proposition

5- Conclusion

Dans cette dernière partie nous avons présenté la réalisation de notre outil programmation automatique . Ce dernier a été commencé par l'exploration des outils utilisés, ensuite la description du système à travers les captures d'écran .

Conclusion

Conclusion Générale

Conclusion

Ce projet intitulé « Conception et Réalisation de l'outil de programmation automatique qui se base sur le principe des questions et des réponses, » a pour but de créer une application d'aide pour ceux qui ont du mal à écrire leurs propres codes de programme; de sorte que cet outil, que nous visons réaliser, vise à fournir le code requis en posant, à ses utilisateurs, des questions pour déterminer le besoin.

Pour ce faire, nous avons suivi trois étapes :

D'abord, nous avons procédé à une étude générale de quelques exemples et des modèles des langages de programmation ; où nous en avons eu un regard général suffisant sur les types de langages et leurs modèles de différences et leurs évolutions selon le temps.

Nous avons présenté, par la suite, une conception de l'application à travers l'élaboration d'une structure servant de modèle illustratif sur lequel va se baser notre pratique d'application.

Enfin, une dernière étape est l'étape exécutive de la conception que nous avons proposée dans l'étape précédente (la deuxième);et ce, en utilisant: le langage de programmation C++, l'environnement de développement Visual studio , base de données MS Access.

Nous avons rencontré plusieurs difficultés lors de la conception et la mise en œuvre de ce projet. Le plus important du manque de références adéquates et des travaux similaires à notre sujet.

En réalisant le présent travail, il s'est avéré l'importance de l'outil de programmation qui serait :

- Outil de programmation didactique du premier degré.
- Il permet à son utilisateur de créer son propre programme.
- Il s'agit d'un outil qui permet d'établir et de mettre en œuvre le programme dans le plus bref temps possible.

Cependant, notre travail peut être amélioré par l'ajout d'autres fonctionnalités telles que:

- L'espace d'utilisateur ; afin de lui donner la possibilité de proposer un nouveau champ.
- La possibilité d'élargir cette application (proposée) le pus maximum possible à travers l'ajout de zones et des langages de programmation.

Bibliographie

Bibliographie

Livres électroniques

[1]. Abrial.J.R : The B-Book, Combridge University Press.

Date: 1996.

[2]. Bernard espinasse , Professeur à l'université d'Aix Marseille « Présentation générale du langage de modélisation objet (UML) » .

[3]. Bernard Maurin , « Algorithmique et programmation Visual Basic » , université lumière lyon 2 faculté de sciences économiques et de gestion
Date : 2005-2006

[4]. Christian CARREZ , Professeur des Universités au CNAM, « Les Systèmes Informatiques » Vision cohérente et utilization
Date : 2002-2003.

[5]. DJEBALI Hadjer, KENOUZE Ilyas « Un Outil d'administration du Système de Gestion de Base de données Oracle (SGBD Oracle) », UNIVERSITE KASDI MERBAH OUARGLA ,Faculté des Nouvelles Technologies de l'Information et de la Communication,Département d'Informatique et Technologie de l'information
Date : 2013 / 2014

[6]. halima chabchoub et rim abdelhedi « conception et réalisation d'une plateforme d'e-learning » , **universite de sfax institut superieur d'informatique et de multimedia**

Date 14 juin 2012

[7]. les étudiants de l'option informatique , « Vers une carte d'identité des langages de programmation » , projet de veille technologique en TIC à Centrale Nantes,
Date:23/02/2011.

[8]. Ludovic DENOYER « Architecture Orientée Service, JSON et API REST» , L'université Pierre-et-Marie-Curie à Paris
Date :03/02/2015

[9]. Nahimana Chantal Et Mbanje Jean Vernack , « Conception Et Realisation D'une Application Web D'aide A La Supervision Des Activites D'impression Dans Une Imprimerie 'Cas D'imprilac' », Faculté: Informatique, Option: Génie Logiciel, Bujumbura
Date : Novembre 2014

[10]. Pr. Mohamed El Marraki, «Algorithmique» , Université Mohammed V-Agdal, Faculté des Sciences Rabat Département Mathématiques et Informatique
Date : 15/02/2007

Web électroniques

[11]. Acceleo , wikipedia, [En ligne] :
https://fr.wikipedia.org/wiki/Acceleo
Date :03/09/2013

[12]. Définition : Langage C++ , [En ligne] :
http://www.depannetonpc.net/lexique/lire_74_langage-c.html
Date :07/10/2007

[13]. Eclipse Modeling Framework, wikipedia, [En ligne] :
https://fr.wikipedia.org/wiki/Eclipse_Modeling_Framework
Date :27/08/2013

[14]. Générateurs de code , , [En ligne] :
http://www.rankspirit.com/generateurs.php

[15]. Jean-Paul , Automates Intelligents , [En ligne] :
www.jean-paul-baquiast.fr
Date : 2002

[16]. Langage procedural et fonctionnel et imperative et structuré , commentcamarche [En ligne] :
http://www.commentcamarche.net/forum/affich-30448497-langage-procedural-fonctionnel-imperatif-structure#q=langage+proc%C3%A9dural&cur=1&url=%2F

[17]. le Générateur automatique du code du balisage , Fonctionnement de JSON-LD Schema , Arobasenet , [En ligne] : http://www.arobasenet.com/2016/07/generateur-balisage-donnees-structurees- 3154.html
Date :05/07/2016

[18]. Microsoft Visual Studio , wikipedia, [En ligne] :
https://fr.wikipedia.org/wiki/Microsoft_Visual_Studio

[19]. Model driven architecture , wikipedia, [En ligne] :
https://fr.wikipedia.org/wiki/Model_driven_architecture
Date :21/02/2017

[20]. Pierre Baron , La programmation fonctionnelle [En ligne] :
http://pierrebaron.fr/blog/la-programmation-fonctionnelle
Date : 23/08/2014

[21]. Produits Mia-Studio , [En ligne] :
http://www.mia-software.com/produits/filiere-java/mia-studio/
Date : 2004 – 2014

[22]. Programmation orientée objet , wikipedia, [En ligne] :
https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet

[23]. Introduction à JavaScript orienté objet , Technologies web pour développeurs: JavaScript , developer Mozilla [En ligne] : https://developer.mozilla.org/fr/docs/Web/JavaScript/Introduction_%C3%A0_JavaScript_orient%C3%A9_objet#Programmation_orientée_prototype

[24]. Programmation/Programmation orientée objet/Classes et objets , wikibooks [En ligne] : https://fr.wikibooks.org/wiki/Programmation/Programmation_orient%C3%A9e_objet/Classes_et_objets

[25]. Techniques de programmation [En ligne] : https://sites.google.com/site/programmationpourlesnuls/techniques-de-programmation

[26]. Simulation et approche Model-Based Design , math works , [En ligne] : https://fr.mathworks.com/products/simulink.html