# Guillaume Lay

Mémoire

# En quoi l’apparition de langages de programmation est-elle nécessaire ?

2020-2021

# Remerciements

Table des matières

[Guillaume Lay 1](#_Toc48817832)

[En quoi l’apparition de langages de programmation est-elle nécessaire ? 1](#_Toc48817833)

[Remerciements 2](#_Toc48817834)

[Intro, comment parler à une machine 5](#_Toc48817835)

[L’origine du bit 5](#_Toc48817836)

[Notion de valeur 5](#_Toc48817837)

[Les couches applicatives 5](#_Toc48817838)

[De plus en plus accessible 5](#_Toc48817839)

[Partie 1 : Qu’est-ce qu’un langage de prog 6](#_Toc48817840)

[Utilité 6](#_Toc48817841)

[Plateforme d’utilisation/Ressources 6](#_Toc48817842)

[Mode d’exécution 6](#_Toc48817843)

[Syntaxe 6](#_Toc48817844)

[POO 6](#_Toc48817845)

[Librairies 6](#_Toc48817846)

[Partie 2 : L’évolution de la technologie et des langages de prog 7](#_Toc48817847)

[Notions d’OS 7](#_Toc48817848)

[Evolution des supports 7](#_Toc48817849)

[Simplification des langages 7](#_Toc48817850)

[Partie 3 : Comment créer un langage de prog 8](#_Toc48817851)

[Compilateur 8](#_Toc48817852)

[Esolangs 8](#_Toc48817853)

[Partie 4 : Pourquoi le besoin de nouveaux langages 9](#_Toc48817854)

[Langages qui meurent 9](#_Toc48817855)

[Nouvelles performances 9](#_Toc48817856)

[Partie 5 : L’impact d’un choix de technologie 10](#_Toc48817857)

[Apprentissage 10](#_Toc48817858)

[Limitations 10](#_Toc48817859)

[Résistances 10](#_Toc48817860)

[Partie 6 : Un esperanto informatique 11](#_Toc48817861)

[Caractéristiques 11](#_Toc48817862)

[Implications 11](#_Toc48817863)

[Partie 7 : Les langages du futur 12](#_Toc48817864)

[Vers du no code ? 12](#_Toc48817865)

[Accessible à tous ? 12](#_Toc48817866)

[Ordinateur quantique ? 12](#_Toc48817867)

[Conclusion 13](#_Toc48817868)

[Bibliographie 14](#_Toc48817869)

[Livre 1 14](#_Toc48817870)

[Livre 2 14](#_Toc48817871)

[Livre 3 14](#_Toc48817872)

[Déroulé méthodologique 15](#_Toc48817873)

[Présentation du thème : 15](#_Toc48817874)

[Liste des personnes rencontrées : 15](#_Toc48817875)

[Calendrier M2 15](#_Toc48817876)

# Intro, comment parler à une machine

Logique, interprétation par l’ordinateur etc…

### L’origine du bit

Vrai ou faux ? Voici la question qui dirige le monde moderne. A l’heure ou l’informatique s’est installée dans notre quotidien, il n’est plus l’heure des « peut-être ».

La langue maternelle de tout matériel informatique est le binaire. C’est l’interprétation directe de ce qui est physiquement écrit dans la machine. Que ce soit un trou dans une carte perforée ou la charge d’un électron, le fondement de l’information est un état physique à deux issues. Cela forme le dialecte le plus basique formé de 1 et de 0.

Jusqu’au temps où l’humain fusionnera avec la technologie, nous ne sommes actuellement pas en mesure de discuter directement avec un ordinateur. Nous utilisons toute sortes de langages afin de communiquer avec ces amas de silicium et de circuits électroniques. JavaScript, PHP, C++… il y en a des centaines et de nouveaux apparaissent tous les ans. Pourquoi il y en a-t-il autant ? Pourquoi sont-ils si différents ? Et surtout, pourquoi en invente-t-on de nouveaux ? C’est à ces interrogations que nous allons tenter de porter des réponses.

### Notion de valeur

Lorsqu’on associe des bits ensemble, on construit ainsi toutes sortes de valeurs pour passer du simple état booléen à quelque chose de plus complexe.

[…]

### Les couches applicatives

Traduire de l’ordinateur à l’humain

[…]

### De plus en plus accessible

Les tâches répétitives sont confiées à la machine

[…]

# Partie 1 : Qu’est-ce qu’un langage de prog

Les caractéristiques d’un langage de prog

### Utilité

Cadre d’utilisation

Diagramme d’utilisation des langages -> Comparer avec une table à manger

Le SI est né dans les domaines de l'informatique et des télécommunications, le concept de SI s'applique maintenant à l'ensemble des organisations, privées ou publiques. Le terme système d'information (ou SI) possède les significations suivantes :

Un ensemble organisé de ressources (personnel, données, procédures, matériel, logiciel, etc.) permettant d'acquérir, de stocker, de structurer et de communiquer des informations sous forme de textes, images, sons, ou de données codées dans des organisations. Selon leur finalité principale, on distingue des systèmes d'information supports d'opérations (traitement de transaction, contrôle de processus industriels, supports d'opérations de bureau et de communication) et des systèmes d'information supports de gestion (aide à la production de rapports, aide à la décision, etc.)4.

Un système ou sous-système d'équipements, d'informatique ou de télécommunication, interconnectés dans le but de l'acquisition, du stockage, de la structuration, de la gestion, du déplacement, du contrôle, de l'affichage, de l'échange (transmission ou réception) de données sous forme de textes, d'images, de sons, et/ou, faisant intervenir du matériel et des logiciels.

Un SI est un réseau complexe de relations structurées où interviennent hommes, machines et procédures, qui a pour but d’engendrer des flux ordonnés d’informations pertinentes provenant de différentes sources et destinées à servir de base aux décisions selon Hugues Angot.

Un SI est un ensemble d'éléments matériels ou immatériels (hommes, machines, méthodes, règles) en interaction transformant en processus des éléments (les entrées) en d'autres éléments (les sorties).

### Plateforme d’utilisation/Ressources

Support, Machine, Propriétaire…

Les langages informatiques employés diffèrent souvent selon chacune de ces catégories, et à l'intérieur des catégories. Par exemple, les systèmes d'information de gestion emploient du Cobol, du langage C, du C++, du Java, du Visual Basic.NET, du WinDev (WLangage), SQL, etc.

Aujourd'hui, la généralisation des applications web rend possible une très forte interopérabilité des systèmes, qui transcende ces catégories traditionnelles. Les langages de balisage (HTML, XML, etc.) s'imposent comme des standards. Ces langages sont souvent associés à des frameworks. Le framework le plus communément employé est actuellement RDF (Resource Description Framework). RDF s'appuie sur des normes d'interopérabilité et l'utilisation massive de métadonnées, données élémentaires communes à toutes les ressources et tous les systèmes quelles que soient leurs utilisations, qui facilitent les accès et les échanges.

Un paradigme est une façon d'approcher la programmation12. Chaque paradigme amène sa philosophie de la programmation ; une fois qu'une solution a été imaginée par un programmeur selon un certain paradigme, un langage de programmation qui suit ce paradigme permettra de l'exprimer13. Impératif, déclaratif, fonctionnel, logique, orienté objet, concurrent, visuel, événementiel, et basé web sont des paradigmes de programmation12. Chaque langage de programmation reflète un ou plusieurs paradigmes, apportant un ensemble de notions qui peuvent être utilisées pour exprimer une solution à un problème de programmation13. Au cours de l'histoire, les scientifiques et les programmeurs ont identifié les avantages et les limitations d'un style de programmation et apporté de nouveaux styles12. La plupart des langages de programmation contemporains de 2013 permettent d'adopter plusieurs paradigmes de programmation12 à condition que ceux-ci soient compatibles.

Le paradigme impératif ou procédural est basé sur le principe de l'exécution étape par étape des instructions tout comme on réalise une recette de cuisine. Il est basé sur le principe de la machine de Von Neumann. Un ensemble d'instructions de contrôle de flux d'exécution permet de contrôler l'ordre dans lequel sont exécutées les instructions qui décrivent les étapes. Le C, le Pascal, le Fortran et le COBOL sont des exemples de langage de programmation qui implémentent le paradigme impératif13.

Il y a essentiellement deux paradigmes déclaratifs ; ce sont le paradigme fonctionnel et le paradigme logique. En paradigme fonctionnel le programme décrit des fonctions mathématiques. En paradigme logique il décrit des prédicats : c'est-à-dire des déclarations qui, une fois instanciées, peuvent être vraies ou fausses ou ne pas recevoir de valeur de vérité (quand l'évaluation du prédicat ne se termine pas)12. Dans un modèle d'implantation, une machine abstraite effectue les opérations nécessaires pour calculer le résultat de chaque fonction14 ou chaque prédicat. Dans ces paradigmes une variable n'est pas modifiée par affectation12. Une des caractéristiques principales15 est la transparence référentielle, qui fait qu'une expression peut être remplacée par son résultat sans changer le comportement du programme.

Le paradigme fonctionnel a pour principe l'évaluation de formules, afin d'utiliser le résultat pour d'autre calculs ; il s'appuie sur la récursivité et il a pour modèle le lambda-calcul, plus précisément la réduction en forme normale de tête. Tous les calculs évaluent des expressions ou font appel à des fonctions. Pour simplifier16, le résultat d'un calcul sert pour le calcul ou les calculs qui ont besoin de son résultat jusqu'à ce que la fonction qui produit le résultat du programme ait été évaluée13. Le paradigme fonctionnel a été introduit par les langages Lisp et ISWIM ainsi qu'en ce qui concerne les fonctions récursives par Algol 60, dans les années 1960. Des langages tels que Ruby et Scala supportent plusieurs paradigmes dont le paradigme fonctionnel12, tandis qu'Haskell ne supporte que le paradigme fonctionnel et OCaml privilégie le paradigme fonctionnel qu'il partage avec le paradigme objet et une petite dose d'impératif.

Le paradigme logique vise à répondre à une question par des recherches dans un ensemble, en utilisant des axiomes, des requêtes et des règles de déduction. L'exécution d'un programme est une cascade de recherches de faits dans un ensemble, en invoquant des règles de déduction. Les données obtenues, peuvent être associées à un autre ensemble de règles et peuvent alors être utilisées dans le cadre d'une autre recherche. L'exécution du programme se fait par évaluation, le système effectue une recherche de toutes les affirmations qui, par déduction, correspondent à au moins un élément de l'ensemble. Le programmeur exprime les règles, et le système pilote le processus13. Le paradigme logique a été introduit par le langage Prolog en 197012.

Le paradigme orienté objet est destiné à faciliter le découpage d'un grand programme en plusieurs modules isolés les uns des autres. Il introduit les notions d'objet et d'héritage. Un objet contient les variables et les fonctions en rapport avec un sujet. Les variables peuvent être privées, c'est-à-dire qu'elles peuvent être manipulées uniquement par l'objet qui les contient. Un objet contient implicitement les variables et les fonctions de ses ancêtres, et cet héritage aide à réutiliser du code12. Le paradigme orienté objet permet d'associer fortement les données avec les procédures13. Il a été introduit par le langage Simula dans les années 1960, et est devenu populaire dans les années 1980, quand l'augmentation de la puissance de calcul des ordinateurs a permis d'exécuter des grands programmes12. Divers langages de programmation ont été enrichis en vue de permettre la programmation orientée objet ; c'est le cas de C++ (dérivé du langage C12), Simula, Smalltalk, Swift et Java sont des langages de programmation en paradigme orienté objet13.

Système de gestion de données techniques (SGDT), ou PDM pour product data management : fonctions d'aide au stockage et à la gestion des données techniques. Surtout utilisé par les bureaux d'études. En fait le PDM est l'évolution de la fonction SGDT, jusqu'à de nouvelles manières de gérer le cycle de vie des données.

### Mode d’exécution

Compilé, interprété

En paradigme concurrent un programme peut effectuer plusieurs tâches en même temps. Ce paradigme introduit les notions de thread, d'attente active et d'appel de fonction à distance12. Ces notions ont été introduites dans les années 1980 lorsque, à la suite de l'évolution technologique, un ordinateur est devenu une machine comportant plusieurs processeurs et capable d'effectuer plusieurs tâches simultanément. Les langages de programmation contemporains de 2013 tels que C++ et Java sont adaptés aux microprocesseurs multi-cœur et permettent de créer et manipuler des threads12. Plus récemment, on a vu apparaître des langages intégralement orientés vers la gestion de la concurrence, comme le langage Go.

Dans la grande majorité des langages de programmation, le code source est un texte, ce qui rend difficile l'expression des objets bidimensionnels12. Un langage de programmation tel que Delphi ou C# permet de manipuler des objets par glisser-déposer et le dessin ainsi obtenu est ensuite traduit en une représentation textuelle orientée objet et événementielle. Le paradigme visuel a été introduit à la fin des années 1980 par Alan Kay dans le langage Smalltalk, dans le but de faciliter la programmation des interfaces graphiques12.

Alors qu'un programme interactif pose une question et effectue des actions en fonction de la réponse, en style événementiel le programme n'attend rien et est exécuté lorsque quelque chose s'est passé12. Par exemple, l'utilisateur déplace la souris ou presse sur un bouton. Dans ce paradigme, la programmation consiste à décrire les actions à prendre en réponse aux événements. Et une action peut en cascade déclencher une autre action correspondant à un autre événement12. Le paradigme événementiel a été introduit par le langage Simula dans les années 1970. Il est devenu populaire à la suite de l'avènement des interfaces graphiques et des applications web12.

Dans les œuvres des années 1980-1990, la composition « classique » des systèmes de l'information d'une entreprise était comme une pyramide des systèmes d'information qui reflétait la hiérarchie de l'entreprise5.

Les systèmes qui traitent les transactions fondamentales (TPS) au fond de la pyramide, suivis par les systèmes pour la gestion de l'information (MIS), et après les systèmes de soutien des décisions (DSS) et se terminant par les systèmes d'information utilisés par la direction la plus supérieure (EIS), au sommet.

Bien que le modèle pyramidal reste utile, un certain nombre de nouvelles technologies ont été développées et certaines nouvelles catégories de systèmes d'information sont apparues et ne correspondent plus aux différentes parties du modèle pyramidal.

### Syntaxe

Avec l’avènement de l'Internet dans les années 1990, les données, les images ainsi que le code s'échangent entre ordinateurs. Si un résultat est demandé à un ordinateur, celui-ci peut exécuter le programme nécessaire, et envoyer le résultat. Il peut également envoyer le code nécessaire à l'ordinateur client pour qu'il calcule le résultat lui-même12. Le programme est rarement traduit en langage machine, mais plutôt interprété ou traduit en une forme intermédiaire, le bytecode, qui sera exécuté par une machine virtuelle, ou traduit en langage machine au moment de l'exécution (just-in-time). Java, PHP et Javascript sont des langages de programmation basée web12.

Chaque appareil informatique a un ensemble d'instructions qui peuvent être utilisées pour effectuer des opérations. Les instructions permettent d'effectuer des calculs arithmétiques ou logiques, déplacer ou copier des données, ou bifurquer vers l'exécution d'autres instructions. Ces instructions sont enregistrées sous forme de séquences de bits, où chaque séquence correspond au code de l'opération à effectuer et aux opérandes, c'est-à-dire aux données concernées ; c'est le langage machine17.

La traduction s'effectue en plusieurs étapes. En premier lieu, le traducteur effectue une analyse lexicale où il identifie les éléments du langage utilisés dans le programme. Dans l'étape suivante, l'analyse syntaxique, le traducteur construit un diagramme en arbre qui reflète la manière dont les éléments du langage ont été combinés dans le programme, pour former des instructions. Puis, lors de l'analyse sémantique, le traducteur détermine s'il est possible de réaliser l'opération, et les instructions qui seront nécessaires dans le langage cible18.

Dans le langage de programmation assembleur, des mots aide-mémoire (mnémonique) sont utilisés pour référer aux instructions de la machine. Les instructions diffèrent en fonction des constructeurs et il en va de même pour les mnémoniques. Un programme assembleur traduit chaque mnémonique en la séquence de bits correspondante19.

Un runtime (traduction : exécuteur) est un ensemble de bibliothèques logicielles qui mettent en œuvre le langage de programmation, permettant d'effectuer des opérations simples telles que copier des données, voire les opérations beaucoup plus complexes20.

Lors de la traduction d'un programme vers le langage machine, les opérations simples sont traduites en les instructions correspondantes en langage machine tandis que les opérations complexes sont traduites en des utilisations des fonctions du runtime. Dans certains langages de programmation, la totalité des instructions sont traduites en des utilisations du runtime20 qui sert alors d'intermédiaire entre les possibilités offertes par la plateforme informatique et les constructions propre au langage de programmation21.

Chaque langage de programmation a une manière conventionnelle de traduire l'exécution de procédures ou de fonctions, de placer les variables en mémoire et de transmettre des paramètres. Ces conventions sont appliquées par le runtime22. Les runtime servent également à mettre en œuvre certaines fonctionnalités avancées des langages de programmation telles que le ramasse-miettes, ou la réflexion20.

Les langages de programmation sont couramment auto-implémentés, c'est-à-dire que le compilateur pour ce langage de programmation est mis en œuvre dans le langage lui-même. Exemple : un compilateur pour le langage Pascal peut être écrit en langage Pascal23.

Les langages de programmation fonctionnent souvent à l'aide d'un runtime.

### POO

Rapide aperçu (La corde à linge)

### Librairies

Choses mises à disposition par le langage, JavaDoc…

# Partie 2 : L’évolution de la technologie et des langages de prog

Des appareils en mise à jour perpétuelle

Des langages de plus en plus riches

## Notions d’OS

## Evolution des supports

## Simplification des langages

# Partie 3 : Comment créer un langage de prog

Créer un compilateur de langage,

Esolangs, cette mine d’or

## Compilateur

## Esolangs

# Partie 4 : Pourquoi le besoin de nouveaux langages

Nouveau matériel, nouvelles performances

8-16-32-64 bits

## Langages qui meurent

## Nouvelles performances

# Partie 5 : L’impact d’un choix de technologie

Partie gestion de projet

Formation de l’équipe

## Apprentissage

## Limitations

## Résistances

# Partie 6 : Un esperanto informatique

Et si on inventait un langage universel ?

## Caractéristiques

## Implications

# Partie 7 : Les langages du futur

Programmer sur autre chose ?

## Vers du no code ?

## Accessible à tous ?

## Ordinateur quantique ?

# Conclusion

# Bibliographie

## Livre 1

## Livre 2

## Livre 3

Déroulé méthodologique :

### Présentation du thème :

Durant tout mon parcours informatique, dès que je parlais de langages de programmation à des non-initiés, j’ai toujours eu droit à cette interrogation : « Mais pourquoi y en a-t-il autant ? ». Même si j’ai une très vague idée de cette raison, j’ai trouvé intéressant de répondre une bonne fois pour toutes à cette question. C’est un sujet qui me passionne et en l’abordant, il me permettra de faire un bilan complet de mes compétences acquises jusqu’à présent.

Ma thématique est assez vague ce qui me permettra d’aborder un large panel de sujets différents.

Liste des personnes rencontrées : 0

### Calendrier M2