

# **L'architecture en appels et retours**

# Table des matières

<b>I. Architecture en appels et retours : une architecture logicielle</b>	<b>3</b>
<b>II. Exercice : Quiz</b>	<b>6</b>
<b>III. Utilité de l'architecture en appels et retours</b>	<b>7</b>
<b>IV. Exercice : Quiz</b>	<b>10</b>
<b>V. Essentiel</b>	<b>11</b>
<b>VI. Auto-évaluation</b>	<b>11</b>
A. Exercice .....	11
B. Test .....	11
<b>Solutions des exercices</b>	<b>12</b>

# I. Architecture en appels et retours : une architecture logicielle

**Durée :** 1 h

**Environnement de travail :** un PC / une tablette / un smartphone

**Prérequis :** aucun

## Contexte

Dans ce cours, vous allez voir ce qu'est l'architecture en appels et retours. Il s'agit d'une des architectures les plus utilisées pour faciliter le développement. Il est important de la connaître pour réaliser de nouveaux projets organisés correctement, faciles à comprendre et à mettre en place pour l'équipe pédagogique. En utilisant les méthodes adaptées, vous vous assurez que votre projet soit réalisable. En tant que développeur, vous serez amené à réaliser des projets. Il vous faudra donc appliquer la méthode la plus adaptée pour la réalisation de votre projet. Le choix de l'architecture doit être déterminant dans votre façon de travailler et de collaborer sur ce projet.

Mettre en place une architecture logicielle, ce n'est pas compliqué. En effet, il s'agit simplement de l'organisation des différents composants de votre logiciel. Ils seront séparés et pourront communiquer entre eux grâce à des liens. Il est donc important, pour le bon déroulement du développement et la mise en production du logiciel au plus vite, de mettre en place une architecture logicielle. Trop souvent laissée de côté par les développeurs, cette étape, si elle est oubliée, peut engendrer de nombreux problèmes. La première question à se poser avant de travailler sur l'architecture logicielle est celle sur le type d'architecture logicielle à mettre en place. Il en existe plusieurs et nous allons voir dans ce cours l'architecture en appels et retours.

L'architecture en appels et retours est un type d'architecture logicielle créée par Niklaus Wirth, un professeur d'informatique suisse, connu pour avoir inventé notamment le langage PASCAL. Elle fait partie de la phase de conception logicielle, c'est-à-dire la phase pendant laquelle le logiciel va être entièrement conçu sous la forme abstraite de schémas. Pour être réalisée, cette architecture logicielle pourra être faite avec des outils CASE (*Computer-Aided Software Engineering*) qui vous permettront de mettre en évidence le lien entre votre architecture et le logiciel final.

## Définition

Une **architecture logicielle** est une manière de schématiser et symboliser les interactions de l'ensemble des éléments du système d'information. Un modèle d'architecture est produit lors d'une phase de conception (c'est-à-dire avant la création logicielle) au contraire de l'analyse fonctionnelle. Il décrit comment l'architecture doit être conçue pour répondre aux mieux aux spécificités logicielles.

## Définition

Les **outils CASE** (*Computer-Aided Software Engineering*) sont des outils d'ingénierie de système assisté par ordinateur. Il s'agit de logiciels ayant pour but d'aider les développeurs en leur facilitant la gestion et la production du code.

## Définition

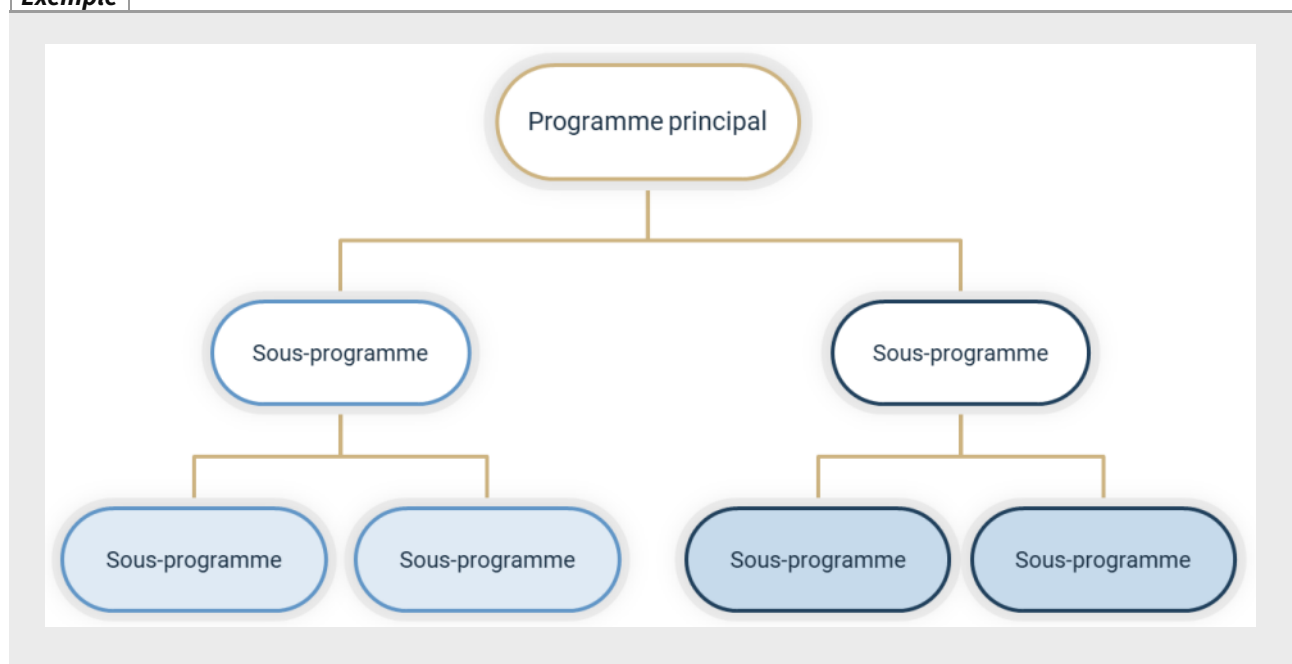
Un **logiciel** correspond à un ensemble de programmes, de procédés et de règles permettant le fonctionnement d'un système informatique.

### Définition

Une **analyse fonctionnelle** correspond à une démarche qui permet de trouver les fonctions d'un produit qui permettront de satisfaire le besoin d'un utilisateur. Il s'agit d'une démarche qui est conduite en mode projet et qui est utilisée lors de la conception et de l'amélioration d'un produit. Cette analyse peut être utilisée tant pour un objet, du matériel, un processus matériel, un processus vivant, une organisation, un logiciel, etc. Ses besoins seront exprimés de façon individuelle et collective, mais aussi objective et subjective. Elle pourra étudier différentes fonctions comme les fonctions de service, d'évaluation et de traitement. Elle prendra également en compte les contraintes et variables de l'environnement étudié.

L'un des plus gros livrables du projet est l'architecture logicielle. Elle représente en moyenne 40 % de l'effort demandé au développement complet du logiciel. Ce taux peut varier en fonction des différents éléments mis à la disposition des développeurs pour la réalisation du projet.

### Exemple

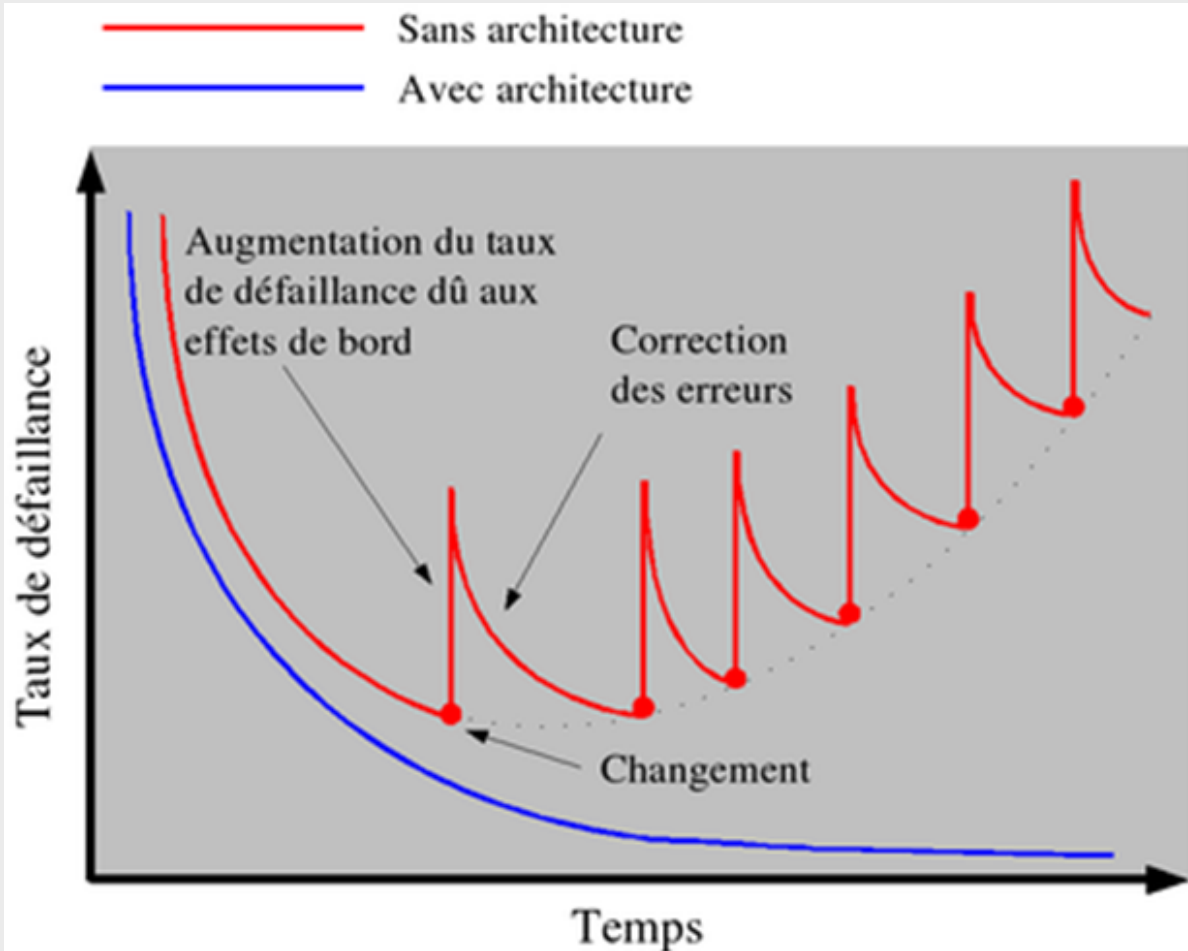


Dans l'exemple ci-dessus, l'on constate bien qu'un programme se décompose en sous-programmes qui eux-mêmes se décomposent en sous-programmes, etc. Un programme invoque des composants qui invoquent eux-mêmes d'autres composants. Comme dans la plupart des schémas de ce type, les éléments de l'architecture sont représentés par des rectangles (ils sont parfois représentés par des formes ovales) et ils sont reliés par des lignes droites (ils peuvent également être reliés par des flèches).

L'approche utilisée pour cette architecture est basée sur le raffinement, c'est-à-dire un détail de la conception permettant d'arriver à l'implémentation finale par itération. Le nom de « *raffinement* » vient du fait que le niveau de granularité sera de plus en plus fin. C'est une technique qui peut tout aussi bien être utilisée pour du code-source que pour un modèle de données.

L'objectif principal de l'architecture logicielle est de réduire les coûts et d'augmenter la qualité du logiciel. En effet, grâce à ce processus, il y aura moins de maintenance à réaliser (correction de *bugs*, modifications logicielles) et il y aura moins de code à produire puisque certains éléments d'une fonction pourront être utilisés plusieurs fois. Tous les ajouts et les modifications qui seront réalisés pourront être faits simplement et rapidement. En ayant une bonne architecture logicielle, vous vous assurez que votre logiciel va fonctionner correctement. Un logiciel pourrait fonctionner de façon correcte et être créé rapidement sans avoir à passer par une architecture logicielle, pourtant, il s'avère que dans la plupart des cas, ces programmes vont rencontrer des problèmes lorsqu'il va falloir faire évoluer le produit. C'est à ce moment-là que l'entreprise aura le plus gros prix à payer puisque les développeurs passeront beaucoup plus de temps sur les différents bugs du programme.

Attention, les architectures logicielles ne doivent pas être confondues avec d'autres concepts et ne doivent pas intégrer les éléments suivants : choix du *framework*, langage, base de données. Du point de vue de l'architecture, ces éléments ne sont que secondaires puisque le principal objectif d'une architecture est de faire ressortir les problèmes de l'application pour les résoudre de façon simple par la suite.

**Exemple**

Source : Techno-Science.net<sup>1</sup>

**Rappel**

Une architecture logicielle au sens strict est définie comme un ensemble de structures qui composent un système, comprenant des éléments logiciels, leurs relations, ainsi que les attributs et leurs relations. En d'autres termes, l'architecture logicielle définit tous les composants du système, leurs interfaces de communication et la manière dont ces composants communiquent entre eux à l'aide de ces interfaces. Les organisations dépendent des systèmes logiciels qui prennent en charge leurs opérations, et leur bon fonctionnement dépend de leur architecture. L'architecture d'un système logiciel est conçue pour répondre aux exigences fonctionnelles et non fonctionnelles établies par les personnes intéressées par le système (les utilisateurs, les clients, les fournisseurs, etc.). La capacité fonctionnelle fait référence aux tâches que le système doit effectuer, tandis que la capacité non fonctionnelle fait référence à la qualité avec laquelle le système doit effectuer ces tâches. Dans le cas d'une entreprise de commerce électronique, un exemple d'exigence fonctionnelle pourrait être : « *Service aux consommateurs X qui permet de facturer le coût des achats en ligne sur la carte de crédit du client* ». Dans ce cas,

<sup>1</sup> <https://www.techno-science.net/glossaire-definition/Architecture-logicielle-page-2.html>

l'exigence non fonctionnelle peut être : « *Utiliser le service X et obtenir une réponse en moins de trois millisecondes* ». Si ce système est mal conçu, l'entreprise de commerce électronique peut perdre beaucoup d'argent, soit parce que le système ne peut pas communiquer correctement avec le système X pour débiter la carte de crédit du client pour l'achat, soit parce que le processus prend plus de temps que le client ne le souhaite. Dans l'un ou l'autre de ces deux scénarios, l'entreprise ne pourra pas réaliser de ventes et perdra beaucoup de clients et d'argent. Une architecture bien conçue est essentielle pour que les entreprises évoluent avec le dynamisme nécessaire dans l'économie d'aujourd'hui. Pour reprendre l'exemple de l'entreprise de e-commerce, si la conception de l'architecture du système n'est pas basée sur de bonnes pratiques d'ingénierie logicielle, alors le système a le potentiel de devenir un obstacle à l'évolution requise par l'entreprise. Si cette structure n'est pas conçue correctement, le bâtiment peut s'effondrer. De même, les systèmes logiciels qui manquent d'une conception architecturale de haute qualité peuvent fonctionner mal ou pas du tout. Pire encore, cela peut avoir des conséquences désastreuses pour l'organisation ou les utilisateurs de ses services.

## Exercice : Quiz

[solution n°1 p.13]

### Question 1

L'architecture logicielle est un type d'architecture en appels et retours.

- ☐ Vrai
- ☐ Faux

### Question 2

L'architecture en appels et retours a été créée par Niklaus Wirth.

- ☐ Vrai
- ☐ Faux

### Question 3

L'architecture logicielle doit être réalisée après les premiers essais de codage.

- ☐ Vrai
- ☐ Faux

### Question 4

Le terme « *outils CASE* » signifie : « *Computer-Aided Software Engineering* ».

- ☐ Vrai
- ☐ Faux

### Question 5

L'idée de l'architecture en appels et retours est qu'un programme se décompose en sous-programmes qui eux-mêmes se décomposent en sous-programmes.

- ☐ Vrai
- ☐ Faux

### III. Utilité de l'architecture en appels et retours

Aussi appelée décomposition fonctionnelle, l'architecture en appels et retours permet de prendre une tâche complexe et de la diviser en plusieurs étapes afin d'avoir une vision globale et détaillée du projet. Ces tâches sont appelées fonctions et pourront être réutilisées plusieurs fois dans un même code. Ces fonctions originales ont différentes structures de sous-programmes qui permettent la découpe d'un problème en sous-tâches. Ce principe a pour but de simplifier un projet complexe. Une fonction sera divisée en sous-fonctions qui seront elles-mêmes divisées en sous-sous-fonctions, etc.

Il ne faut donc pas oublier qu'il est conseillé de créer des fonctions dans votre code et que chaque fonction doit avoir un rôle particulier. Leur création facilitera le travail en équipe puisqu'il sera plus facile pour une personne même externe au projet de comprendre le code.

Une fonction doit contenir quatre informations :

- Le nom (un nom adapté à la fonction),
- Un corps : c'est ici que sera décrit le contenu de la fonction (ce qu'elle devra exécuter),
- Le retour (le type et la valeur résultat de la fonction),
- Les paramètres (facultatif) : ce sont les valeurs que la fonction reçoit lors de son appel.

La **syntaxe type** est la suivante :

```
1 type nom(paramètres)
2 {
3     /* Corps de la fonction */
4     retour
5 }
```

Par exemple, en python, pour définir une fonction, on utilisera cette syntaxe :

```
1 def Fonction_nom(paramètres formels) :
2     #bloc d'instructions...
3     return valeur_resultat
```

En python, le mot « *def* » désigne la définition d'une fonction. Il s'ensuivra le nom de la fonction, puis des parenthèses qui contiendront les paramètres. Ensuite il y aura le corps de la fonction c'est-à-dire un bloc de code qui définit les instructions de la fonction, puis il y aura un retour qui retournera un résultat.

Les **fonctions** sont des concepts essentiels pour programmer, et elles se rapprochent des fonctions en mathématiques. Elles s'appliquent à toute sorte de langage. Dans une fonction, il y a d'abord les données qui arrivent en entrée et qui, grâce au code créé dans la fonction, retournent un résultat en sortie qui traitera les données de la fonction. Il existe deux types de fonctions. Tout d'abord, il y a les fonctions natives. Ce sont des fonctions qui ont été créées par les auteurs du langage et qui sont mises à disposition des programmes. Puis il y a les fonctions définies par l'utilisateur : ces fonctions sont créées par le développeur afin d'ajouter des fonctionnalités spécifiques au programme qu'il est en train de créer. Leur intérêt est de décomposer des tâches complexes en tâches beaucoup plus simples. Elles pourront être appelées plusieurs fois et éviteront ainsi aux développeurs de répéter plusieurs fois les mêmes morceaux du code.

#### Définition

Les **fonctions mathématiques** correspondent à une relation  $F$  entre deux valeurs. C'est cette relation qui est appelée fonction. Une valeur de la variable sera associée à une seule autre valeur de la variable dépendante.

### Définition

Les deux grands paradigmes de programmation :

- **Paradigmes de programmation impératifs** : ils font référence aux premiers langages de programmation où le code était généralement plus long. Parmi eux, on retrouve Pascal, C, Cobol, C++, etc.
- **Paradigmes de programmation déclaratifs** : plus courts et plus précis (mais évidemment moins compréhensibles en raison du manque de spécification). Certains de ces langages sont SQL, ML, Prolog, etc.

### Rappel

Le code d'un programme écrit en C est divisé en fonctions. Bien que similaires aux « méthodes » de Java, les fonctions ne sont pas affectées à des classes ou à des objets. Les fonctions en C ne se distinguent que par leurs noms. Deux fonctions avec le même nom et des nombres et types d'arguments différents sont considérées comme des définitions multiples et sont donc fausses. Une fonction encapsule généralement une opération plus ou moins complexe à partir de laquelle un résultat est dérivé. Pour ce faire, les fonctions peuvent avoir besoin d'appeler d'autres fonctions (ou même des fonctions récursives). La fonction d'un programme est l'entité, compte tenu d'un ensemble de données (paramètres), responsable de l'exécution d'une tâche spécifique et attendant d'obtenir le résultat. L'idéal est de diviser les tâches complexes en morceaux plus simples, qui sont implémentés en tant que fonctions. Diviser et regrouper les tâches en fonctions est l'un des aspects les plus importants de la conception d'un programme.

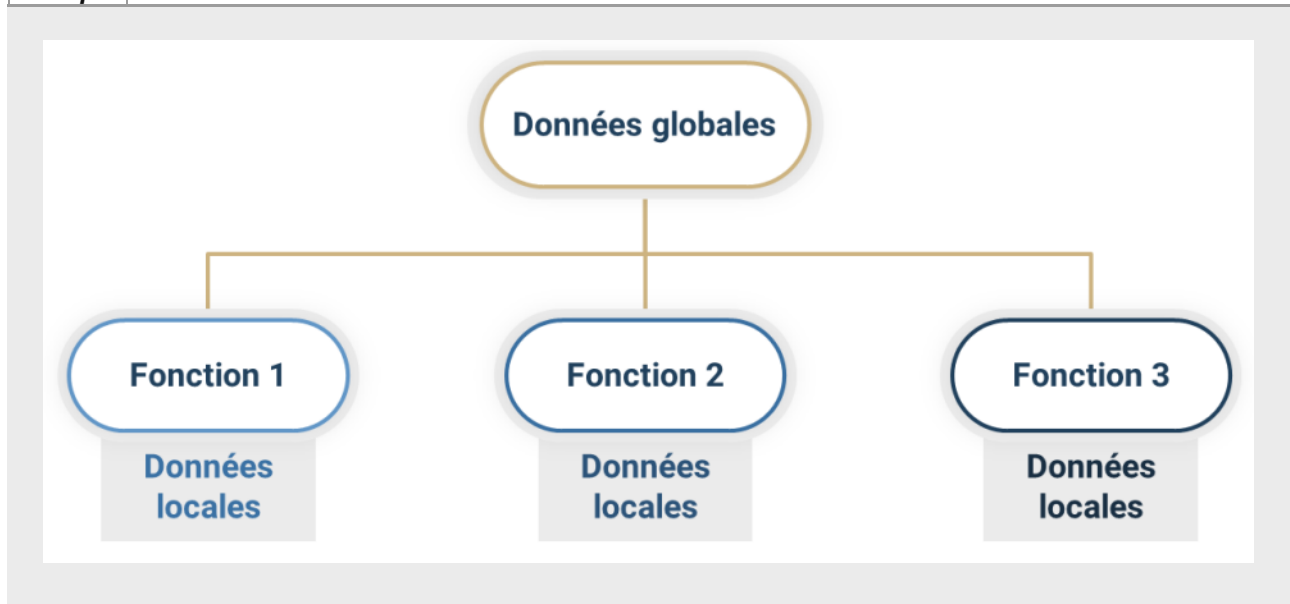
Lorsqu'une fonction est appelée, elle attribue des valeurs à ses paramètres et commence à exécuter le corps jusqu'à ce que la fin soit atteinte ou que l'instruction de retour soit trouvée. Si la fonction renvoie un résultat, l'instruction doit être suivie des données à renvoyer.

Il est également possible de parler de programmation procédurale. Un programme va être divisé en sous-parties appelées fonctions. La procédure utilisée est celle de « étape par étape ». Pour résoudre chaque étape, des fonctions spécifiques seront utilisées.

### Définition Programmation procédurale

Le principe de la **programmation procédurale** est qu'un programme va être divisé en plusieurs sous-parties qui seront appelées procédures ou fonctions. Dans la programmation procédurale, c'est la procédure étape par étape qui va être exécutée. Les différents problèmes rencontrés lors de la mise en place de l'architecture seront décomposés en différentes parties, ce qui permettra une résolution plus facile et plus rapide. Chaque sous-partie sera représentée par une fonction. Une fois ce travail effectué, le code sera plus facilement lisible et les développeurs gagneront du temps, ce qui réduira également les coûts de l'entreprise. Bien sûr la programmation procédurale n'est pas sans faille, et pour corriger les défauts de ce type de programmation, il est possible d'utiliser la programmation orientée objet, qui a introduit le concept d'objets et de classe.



**Exemple**

Ce type d'architecture présente certains avantages :

- En séparant le code en plusieurs fonctions, cela augmente les possibilités d'obtenir un code complexe mais facile à comprendre pour le développer.
- La séparation en différentes fonctions permet également un meilleur travail d'équipe.
- On peut parler de calcul partagé, c'est-à-dire que chaque fonction représente un petit bout du code. Ensemble, elle forme une entité.
- Il s'agit d'une méthode pouvant être utilisée pour tous les projets divisés en plusieurs tâches.
- Meilleure lisibilité du programme.

La première étape de la mise en place de l'architecture d'appels et retours est la création d'un diagramme de décomposition fonctionnelle. Il permettra de mettre en évidence les relations entre les différentes fonctions et les éléments individuels. Il donnera une vision globale du projet et permettra donc une compréhension plus aisée des différentes actions à réaliser. Il facilitera le travail d'équipe et facilitera l'intégration de nouveaux développeurs sur le projet. Il permettra également de faciliter la correction de bugs.

Les architectures de type appels et retours sont utilisées principalement pour programmer facilement et apporter toutes les modifications nécessaires. Il existe des sous-catégories à cette architecture :

- L'architecture d'appel et de procédure à distance : permet de présenter un programme principal ou une architecture de sous-programmes divisés entre plusieurs ordinateurs sur un même réseau.
- L'architecture du programme principal ou des sous-programmes : il y a un programme principal qui est divisé en sous-programmes ou fonctions permettant d'appeler d'autres composants.

### **L'architecture distribuée**

L'architecture distribuée est une architecture dérivée de l'architecture en appels et retours. Comprendre ce concept d'architecture distribuée peut vous aider à mieux comprendre le principe de l'architecture en appels et retours puisque le fonctionnement reste le même. Il s'agit de deux techniques similaires, à la différence que l'une s'applique pour le développement et l'autre pour le réseau.

L'architecture distribuée correspond à un réseau dont l'ensemble des ressources est distribué sur différentes machines. Il s'agit d'un concept opposé à celui de l'informatique centralisée. L'architecture centralisée correspond à un ensemble de ressources disponibles sur un seul ordinateur central. L'un des exemples les plus connus d'architecture distribuée est celui d'Internet. En effet, sur Internet, il n'y a pas qu'un seul ordinateur central, mais plusieurs machines séparées les unes des autres et qui peuvent communiquer entre elles au travers du réseau.

Ce type d'architecture représente de nombreux avantages :

- En distribuant des traitements sur diverses machines présentes sur le réseau, les ressources disponibles pourront être multipliées. Pour reprendre l'exemple d'Internet, chaque ordinateur constitue un élément de tout le réseau sur lequel peuvent être présentes des ressources.
- Cette méthode peut être utilisée pour tous les projets permettant des calculs parallélisables.

### L'avenir de ce type d'architecture

Ce type d'architecture pourrait changer notre façon de consommer des logiciels informatiques. En effet, un logiciel pour être complété de fonctionnalités provenant d'autres logiciels qui peuvent être faits pour se compléter les uns avec les autres selon les besoins de l'utilisateur. Un utilisateur pourrait acheter un logiciel principal et louer les fonctionnalités qui peuvent l'intéresser. Il ne serait donc pas obligé d'acheter un logiciel complet avec des fonctionnalités qui ne lui seraient pas utiles, mais pourrait réellement acheter un logiciel qui correspond parfaitement à ses besoins puisqu'il n'aurait qu'à choisir les fonctionnalités qui lui sont nécessaires. Cela permettrait aux éditeurs de proposer des logiciels, des programmes, des applications toujours plus proches des besoins réels d'un client.

#### Rappel

La programmation procédurale est un paradigme de programmation dérivé de la programmation impérative et basé sur le concept d'appels de procédure. Une procédure consiste simplement en une série d'étapes de calcul à effectuer. Toute procédure donnée peut être appelée par d'autres procédures ou par elle-même à tout moment pendant l'exécution du programme. Les premiers grands langages de programmation procédurale sont apparus vers 1957-1964, dont Fortran, ALGOL, COBOL, PL/I. Pascal et C ont été publiés vers 1970-1972. Le processeur de l'ordinateur fournit un support matériel pour le processus de programmation via des registres de pile et des instructions pour appeler et renvoyer des procédures.

La modularité est souvent souhaitable, en particulier dans les programmes vastes et complexes. Les entrées sont généralement spécifiées syntaxiquement en tant qu'arguments et les sorties sont données en tant que valeurs de retour. Les cadres sont une autre technique qui aide à garder les programmes modulaires. Cette procédure empêche que plusieurs procédures accèdent aux variables qui ne sont pas les siennes sans autorisation explicite. Les programmes moins modulaires, généralement pour les programmes petits ou écrits rapidement, ont tendance à interagir avec un grand nombre de variables dans l'environnement d'exécution qui peuvent également être modifiées par d'autres programmes. En raison de leur capacité à spécifier une interface simple, autonome et réutilisable, les procédures constituent un moyen pratique de créer des extraits de code écrits par différentes personnes ou groupes, même via des bibliothèques de programmation.

### Exercice : Quiz

[solution n°2 p.13]

#### Question 1

L'architecture en appels et retours est aussi appelée décomposition programmée.

- ☐ Vrai
- ☐ Faux

#### Question 2

Les différentes tâches d'un programme peuvent être divisées en fonctions.

- ☐ Vrai
- ☐ Faux

## Question 3

Ce principe a pour but de simplifier un projet complexe.

- ☐ Vrai
- ☐ Faux

## Question 4

Plusieurs fonctions peuvent avoir le même rôle.

- ☐ Vrai
- ☐ Faux

## Question 5

L'usage de ces fonctions et de ce découpage dans un code peut être appelé programmation procédurale.

- ☐ Vrai
- ☐ Faux

## V. Essentiel

L'architecture en appels et retours est un type d'architecture logicielle créée par Niklaus Wirth. Pour fonctionner, cette architecture fonctionne comme cela : un programme se décompose en sous-programmes qui eux-mêmes se décomposent en sous-programmes, etc. Comme dans la plupart des schémas de ce type, les éléments de l'architecture sont représentés par des rectangles et ils sont reliés par des lignes droites. Aussi appelée décomposition fonctionnelle, l'architecture en appels et retours permet de prendre une tâche complexe et de la diviser en plusieurs étapes afin d'avoir une vision globale et détaillée du projet. Ces tâches sont appelées fonctions et pourront être utilisées plusieurs fois dans un même code. Il est également possible de parler de programmation procédurale. Un programme va être divisé en sous-parties appelées fonctions. La procédure utilisée est celle de « l'étape par étape ». Pour résoudre chaque étape, des fonctions spécifiques seront utilisées.

## VI. Auto-évaluation

### A. Exercice

Vous travaillez dans une entreprise qui a un nouveau projet d'application. Vous êtes désigné pour réaliser la conception de l'application. Vous décidez de mettre en place une architecture en appels et retours.

#### Question 1

[solution n°3 p.15]

Comment pouvez-vous argumenter, expliquer ce choix ?

#### Question 2

[solution n°4 p.15]

En quoi l'architecture en appels et retours est-elle comparable à l'architecture d'Internet ?

### B. Test

#### Exercice 1 : Quiz

[solution n°5 p.15]

## Question 1

L'approche utilisée pour cette architecture est basée sur l'amplification.

- ☐ Vrai
- ☐ Faux

## Question 2

L'objectif principal de l'architecture logicielle est de réduire les coûts et d'augmenter la qualité du logiciel.

- ☐ Vrai
- ☐ Faux

Question 3

L'un des avantages de la mise en place de cette architecture est que le code sera davantage protégé des menaces.

- ☐ Vrai
- ☐ Faux

Question 4

La première étape de la mise en place de l'architecture d'appels et retours est la création d'un diagramme de décomposition fonctionnelle.

- ☐ Vrai
- ☐ Faux

Question 5

L'architecture distribuée est une architecture dérivée de l'architecture en appels et retours.

- ☐ Vrai
- ☐ Faux


## Solutions des exercices

**Exercice p. 6 Solution n°1****Question 1**

L'architecture logicielle est un type d'architecture en appels et retours.

☐ Vrai

☒ Faux


 L'architecture en appels et retours est un type d'architecture logicielle.

**Question 2**

L'architecture en appels et retours a été créée par Niklaus Wirth.

☒ Vrai

☐ Faux


 L'architecture en appels et retours est un type d'architecture logicielle créée par Niklaus Wirth, un professeur d'informatique suisse connu pour avoir inventé notamment le langage PASCAL.

**Question 3**

L'architecture logicielle doit être réalisée après les premiers essais de codage.

☐ Vrai

☒ Faux


 Elle fait partie de la phase de conception logicielle, c'est-à-dire la phase pendant laquelle le logiciel va être entièrement conçu sous la forme abstraite de schémas.

**Question 4**

Le terme « *outils CASE* » signifie : « *Computer-Aided Software Engineering* ».

☒ Vrai

☐ Faux


 Les outils CASE (Computer-Aided Software Engineering) sont des outils d'ingénierie de système assisté par ordinateur. Il s'agit de logiciels ayant pour but d'aider les développeurs en leur facilitant la gestion et la production du code.

**Question 5**

L'idée de l'architecture en appels et retours est qu'un programme se décompose en sous-programmes qui eux-mêmes se décomposent en sous-programmes.

☒ Vrai

☐ Faux

 Un programme se décompose en sous-programmes qui eux-mêmes se décomposent en sous-programmes, etc. Un programme invoque des composants qui invoquent eux-mêmes d'autres composants.


**Exercice p. 10 Solution n°2**

### Question 1

L'architecture en appels et retours est aussi appelée décomposition programmée.

☐ Vrai

☒ Faux


 Aussi appelée décomposition fonctionnelle, l'architecture en appels et retours permet de prendre une tâche complexe et de la diviser en plusieurs étapes.

### Question 2

Les différentes tâches d'un programme peuvent être divisées en fonctions.

☒ Vrai

☐ Faux


 Ces tâches sont appelées fonctions et pourront être réutilisées plusieurs fois dans un même code.

### Question 3

Ce principe a pour but de simplifier un projet complexe.

☒ Vrai

☐ Faux


 Ce principe a pour but de simplifier un projet complexe. Une fonction sera divisée en sous-fonctions qui seront elles-mêmes divisées en sous-sous-fonctions, etc.

### Question 4

Plusieurs fonctions peuvent avoir le même rôle.

☐ Vrai

☒ Faux


 Il est conseillé de créer des fonctions dans votre code et chaque fonction doit avoir un rôle particulier.

### Question 5

L'usage de ces fonctions et de ce découpage dans un code peut être appelé programmation procédurale.

☒ Vrai

☐ Faux

 Il est également possible de parler de programmation procédurale. Un programme va être divisé en sous-parties appelées fonctions.

**p. 11 Solution n°3**

En choisissant une architecture en appels et retours, vous faites un choix qui sera économique pour votre entreprise et demandera moins de travail à vos collègues pour la correction de bugs. Le principe de cette architecture est de rendre ce projet complexe simple à comprendre et à réaliser pour vos collaborateurs. Pour cela, vous allez diviser le programme en plusieurs fonctions qui seront elles-mêmes séparées en autres fonctions, etc.

**p. 11 Solution n°4**


Internet fonctionne de telle manière que chaque ordinateur peut potentiellement représenter un morceau d'Internet. Les éléments sont reliés et peuvent se retrouver en communiquant sur le réseau. Internet n'est pas une seule entité mais de nombreuses entités partout dans le monde qui forment un grand ensemble. En utilisant l'architecture en appels et retours dans votre code, vous divisez votre code en plusieurs sous-parties (les fonctions) qui feront fonctionner votre code en communiquant entre elles.

**Exercice p. 11 Solution n°5****Question 1**

L'approche utilisée pour cette architecture est basée sur l'amplification.

☐ Vrai

☒ Faux


 L'approche utilisée pour cette architecture est basée sur le raffinement, c'est-à-dire un détail de la conception permettant d'arriver à l'implémentation finale par itération.

**Question 2**

L'objectif principal de l'architecture logicielle est de réduire les coûts et d'augmenter la qualité du logiciel.

☒ Vrai

☐ Faux

 L'objectif principal de l'architecture logicielle est de réduire les coûts et d'augmenter la qualité du logiciel. En effet, grâce à ce processus, il y aura moins de maintenance à réaliser et il y aura moins de code à produire.

**Question 3**

L'un des avantages de la mise en place de cette architecture est que le code sera davantage protégé des menaces.

☐ Vrai

☒ Faux

 Ce n'est pas un des avantages à cette architecture. Elle ne vise pas à protéger le code.

**Question 4**

La première étape de la mise en place de l'architecture d'appels et retours est la création d'un diagramme de décomposition fonctionnelle.

☒ Vrai

☐ Faux

Q Il permet de mettre en évidence les relations entre les différentes fonctions et les éléments individuels.

### Question 5

---

L'architecture distribuée est une architecture dérivée de l'architecture en appels et retours.

☒ Vrai

☐ Faux

Q Il s'agit de deux techniques similaires, à la différence que l'une s'applique pour le développement et l'autre pour le réseau.