MASTER EN SCIENCES INFORMATIQUES

Identification des manquements dans l'application des modèles mentaux des programmeurs novices et experts

LEYMAN, Jean-Marc

Award date:
2022

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2021–2022

Identification des manquements dans l'application des modèles mentaux des programmeurs novices et experts

Jean-Marc Leyman

Promoteur :  (Signature pour approbation du dépôt - REE art. 40)
Patrick Heymans

Co-promoteur : Julie Henry

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

Ce mémoire réalise un état de l'art traitant des modèles mentaux et, plus particulièrement, des modèles mentaux entrant dans le processus de compréhension de programme. Il met en évidence l'importance des modèles mentaux dans la conception logicielle, mais également la difficulté de traiter ce sujet dans ce domaine et le désintérêt progressif du monde scientifique à son encontre. Les cinq modèles mentaux identifiés dans cet état de l'art, (1) "Text-based" ; (2) "Situation" ; (3) "Program" ; (4) "Top-down" et (5) "Bottom-up", sont ensuite utilisés dans une étude visant à mettre en évidence des éventuels manquements dans leur utilisation et à identifier le type de programmeur présentant ces manquements. Cette identification est rendue possible grâce au profilage des participants réalisé durant l'étude et qui va proposer une définition des types novices et experts sur base du niveau d'expertise du programmeur.

Mots-clès : Modèles mentaux ; Conception ; Intégration ; Compréhension ; Programme ; Novice ; Expert ;

Remerciements

Je tiens à remercier toutes les personnes qui ont participé à la réalisation de ce mémoire.

Premièrement, je remercie Julie Henry, co-promotrice du mémoire, pour son aide précieuse dans la rédaction des ces lignes. Son expertise sur le sujet des modèles mentaux, ainsi que sur les techniques de rédaction d'un tel ouvrage, m'a permis de rationaliser mes idées et présenter les résultats de manière intelligible.

Je remercie Marianne Bruyninckx, ma compagne, pour son soutien et ses encouragements durant l'ensemble de mon parcours scolaire. Son aide indispensable dans la relecture de cet ouvrage et ses conseils lorsque l'inspiration me manquait m'ont permis de garder le cap. Son analyse critique sur mon style de rédaction m'a permis non seulement d'améliorer mon orthographe, mais également d'avancer l'esprit serein.

Je remercie également tous les participants au sondage, pour le temps qu'ils ont bien voulu y consacrer.

Je remercie l'ensemble du corps professoral de l'Université de Namur, pour m'avoir transmis une partie de leur savoir et m'avoir enseigné les compétences nécessaires pour arriver à sans encombre à cette étape finale de mon parcours scolaire.

Enfin, je remercie l'ensemble de mes collègues étudiants qui, par leur bonne humeur et leur camaraderie, m'ont donné l'impression que ces efforts et ce travail n'étaient qu'un jeu, d'où tout le monde sort gagnant dès qu'on y mettait un peu du sien.

Table des matières

Table des figures

Liste des tableaux

1 Introduction

1.1 Objet du mémoire

Ce mémoire fait suite à l'observation d'une tendance de plus en plus présente dans le milieu des entreprises de développement de solutions informatiques : la mise en place de formules d'accompagnement des jeunes recrues et des employés en reconversion, afin de les préparer au mieux à leur futur travail. Le principe de ces séances de formation n'est évidemment pas nouveau, mais innove dans leur principe. Quand les formations standards s'orientent sur une technologie précise, les nouvelles formules vont beaucoup plus loin en proposant une formation sur un ensemble de technologies. De plus, quand les formations standards mettent l'accent sur la théorie durant quelques jours de cours donné par un expert dans le domaine, le gros de la pratique se faisant en auto-formation à postériori, les nouvelles formules sont données sur plusieurs semaines, voire plusieurs mois, en mélangeant théorie et pratique de manière dirigée et continue. A l'origine de cette tendance, le besoin toujours plus présent des entreprises à trouver des collaborateurs efficaces dès leur entrée en fonction, et ce indifféremment des technologies utilisées dans l'entreprise. Le but derrière ces formules d'accompagnement n'est donc pas de former les participants à un ensemble de technologies en particulier, mais de tenter de détecter les éventuels manquements des participants, au niveau du processus d'apprentissage ou de mise en place de solutions, et de tenter de combler ces manquements avant leur entrée en fonction effective.

Pour identifier et, par la suite, corriger ces manquements, il faut déterminer quels facteurs sont à prendre en considération :
Le processus d'apprentissage : Il répond à des codes connus et appliqués dès le début de la scolarité. Ces codes font écho aux opérations cognitives utilisées par Johnson-Laird [Johnson-Laird, 1980 ; Johnson-Laird, 1983] lorsqu'il donne la première définition du concept de modèles mentaux : *"Les modèles mentaux sont la tentative d'expliquer comment l'esprit est capable de traiter une grande quantité d'informations, souvent de manière efficace et appropriée, sans passer par les déductions formelles."* (traduction). Il définit également ces opérations cognitives comme des déductions logiques plus rapides et moins contraignantes que ces dernières. Les manquements pourraient donc être assimilés à des déduction erronées lors de la construction de modèles mentaux. De plus, les modèles mentaux étant un ensemble de déductions logiques, il est possible d'imaginer que, lors de l'application de ces modèles mentaux, des raccourcis ou des chemins détournés soient à l'origine de ces manquements.
La mise en place de solutions : Ces solutions dépendent de l'identification et de l'application des modèles mentaux les plus appropriés au processus d'apprentissage lors de la conception de solutions informatiques.

L'objectif de ce mémoire est donc :

— (1) L'identification des modèles mentaux entrant dans le processus d'apprentissage, qui sont agnostiques à la fois de la technologie et du contexte dans lequel ils seront appliqués, afin de faciliter la conception de solutions informatiques

— (2) Ces modèles mentaux, identifiés au travers d'une revue de la littérature, seront soumis à une étude comparative permettant de distinguer quel(s) profil(s), novice ou expert, applique ces modèles mentaux, dans quelles conditions, et de tenter de déterminer si l'un des profils, ou les deux, les appliquent de manière erronée.

1.2 Méthodologie

L'objectif de ce mémoire étant double, (1) L'identification des modèles mentaux au travers d'un état de l'art et (2) L'étude comparative de l'application de ces modèles mentaux au travers l'analyse des réponses à un questionnaire, ce mémoire se compose de deux phases.

La première phase est consacrée à la revue de la littérature concernant les modèles mentaux. Les écrits ont permis, dans un premier temps, de définir le concept de modèle mental. Ce concept touchant beaucoup de domaines, il a été nécessaire de réduire le domaine de recherche au domaine des sciences de l'informatique, mais également de se restreindre aux sujets traitant du développement et de la programmation en général. Ensuite, étant donné que ce mémoire se concentre sur l'application des modèles mentaux dans un contexte d'entreprise, une catégorisation des résultats a permis d'écarter la documentation concernant l'enseignement de ces modèles mentaux, ainsi que les études à finalité sociologique ne prenant en compte le domaine des sciences de l'informatique que pour illustrer le propos. Enfin, l'étude de la documentation restante a permis de mettre en évidence les modèles mentaux entrant dans le processus de compréhension de programmes.

La seconde phase, l'étude comparative, est, quant à elle, divisée en trois parties. La première partie consiste à définir un questionnaire permettant de mettre en évidence les modèles mentaux identifiés lors de la première phase, au travers de situations classiques rencontrées en entreprise, et de soumettre le questionnaire à un public-cible le plus large possible. Le public-cible n'étant pas restreint, le questionnaire comporte également une section permettant l'identification du profil des répondants. La seconde partie présente les résultats obtenus, sans considérer le profil des sondés, et propose un système de profilage sur base de l'identification des répondants. Enfin, la troisième partie réinterprète les résultats en appliquant le profilage effectué à la partie précédente, pour tenter d'identifier des différences dans l'application des modèles mentaux entre les profils novices et experts, et tenter de déterminer si des manquements sont déjà perceptibles ou si une étude plus détaillée est nécessaire.

2 État de l'art

2.1 Modèles mentaux

2.1.1 Définition & Origines

C'est au début des années 80 que le concept de modèle mental commence à faire l'objet d'études scientifiques, plus particulièrement dans les domaines psychologique et social. Johnson-Laird [Johnson-Laird, 1980 ; Johnson-Laird, 1983] est le premier à donner une définition générale du concept : *"Cette théorie met en évidence certaines opérations cognitives qui, d'une part, semblent fonctionner comme une déduction logique, mais, d'autre part, sont plus rapides et moins contraignantes que celles-ci. Les modèles mentaux sont la tentative d'expliquer comment l'esprit est capable de traiter une grande quantité d'informations, souvent de manière efficace et appropriée, sans passer par les déductions formelles."* (traduction). Selon lui, imaginer une situation revient à construire un modèle mental. Un modèle mental fonctionne comme une déduction logique rapide présentant moins de contraintes que les représentations mentales. Les déductions formelles, souvent liées à une grammaire complexe inhérente à toute représentation mentale, sont réduites à leur strict minimum pour permettre à l'esprit de traiter une grande quantité d'informations de manière efficace. Plusieurs études et réflexions, dont celles de Meunier et Berten [Meunier et Berten, 1995], vont affiner le concept et donner des précisions sur le mécanisme de construction d'un modèle mental. Meunier et Berten identifient deux conséquences à la théorie de Johnson-Laird : (1) *"Les modèles [...] doivent donc reposer sur des éléments situés à un niveau inférieur aussi bien aux mots qu'aux scènes considérées"* et (2) *"On ne construit qu'un seul modèle mental, un modèle unique et provisoire, mais qui, une fois construit, peut être révisé, au fil des énoncés successifs au moyen d'une procédure récursive"*.

En parallèle à ces réflexions, Kintsch et Van Dijk [Van Dijk, Kintsch et al., 1983] proposent de catégoriser les modèles mentaux en se basant sur la compréhension du discours. Selon eux, les modèles de compréhension de texte sont basés sur une représentation mentale multicouche du texte [Corritore et Wiedenbeck, 1999] :

- (1) La représentation textuelle (ou la forme de surface) correspond à la forme littérale du texte. C'est une représentation incomplète, sauf si le texte est déjà connu.
- (2) La base textuelle propositionnelle est composée des propositions dans le texte et des relations. C'est une représentation linguistique isomorphe à la structure du texte, supposée être construite automatiquement lors

de la lecture.

— (3) Le modèle de situation (ou domaine) représentant la situation dans le monde que décrit le texte. Ce n'est pas une représentation du texte mais du sens du texte. Le modèle de situation n'est pas construit automatiquement et sa construction dépend du lecteur qui fait des inférences à partir du texte et de ses connaissances sur le domaine du texte [Ehrlich et Tardieu, 1993 ; Mills et al., 1995].

Ce modèle de représentation multicouche est le point de départ de plusieurs études empiriques appliquant la théorie des modèles mentaux dans le domaine de l'informatique et de la compréhension de programme Alardawi et Agil, 2015 ; Burkhardt et al., 1997 ; Butcher, 2006 ; Schwamb, 1990.

Les modèles mentaux sont omniprésents et s'appliquent à tous les domaines. Ainsi, la loi de l'offre et de la demande en économie est un modèle mental, au même titre que le rasoir d'Ockham en philosophie ou encore la régression vers la moyenne en statistique. Certains modèles peuvent être transversaux et s'appliquer à plusieurs domaines. C'est le cas de l'ordre de grandeur qui peut être appliqué aux domaines scientifiques et économiques. Le principe de Pareto, plus communément connu sous le nom de la loi des 80/20, est à l'origine lié au domaine économique. Il est pourtant également présent dans les domaines du marketing, de la productivité et de la conception logicielle.

De par sa nature, un modèle mental n'est pas fixe. Il évolue dans le temps et diffère suivant les personnes qui l'utilisent et le contexte dans lequel il va être utilisé. Une définition rendant mieux la nature évolutive des modèles mentaux a été proposée par Furlough dans son étude philosophique "Mental Model Structures : Differences at Multiple Levels of Experience" [Furlough et al., 2017]. Selon lui, *"les modèles mentaux sont des représentations mentales du monde extérieur que les humains utilisent constamment lorsqu'ils interagissent avec l'environnement et les systèmes qu'il contient. Ces modèles mentaux sont en partie constitués par une structure sous-jacente de concepts associés qui sont modifiés à mesure qu'une personne acquiert de l'expérience avec un système ou un domaine."* (traduction). Furlough met en évidence le fait que les modèles mentaux évoluent au fur et à mesure de l'expertise d'un individu dans un domaine particulier. Il semble également sous-entendre qu'un modèle mental sur un domaine particulier peut diverger suivant les individus, si ceux-ci n'évoluent pas dans le même environnement.

Au regard de ces définitions, il est légitime de se demander en quoi un modèle mental peut être utile, si ce dernier est volatile et dépendant de facteurs extérieurs ? Un exemple va permettre de mettre en évidence l'importance d'identifier les modèles mentaux qui permettent de se représenter le monde extérieur. Suivant son expérience, chaque individu peut donner une représentation mentale différente de ce qu'est un "avion". Ainsi, les amateurs de voyages pourraient décrire des avions de ligne, qui peuvent différer suivant la compagnie aérienne, la longueur du vol, le nombre de passagers... Les férus de vitesse pourraient avoir tendance à décrire un avion de chasse moderne. Les historiens, spécialistes de la Première Guerre mondiale, pourraient décrire un biplace à hélice. De même que les pilotes amateurs pourraient décrire un ULM, ou encore les milliardaires, leur jet privé. Ces représentations présentent des similitudes intrinsèques qui permettent de les identifier comme étant des représentations mentales désignant le terme "avion". Chaque déduction formelle citée possède un cockpit, des ailes, une queue, un fuselage, des roues et un système de propulsion. Bien que tous ces éléments ne se ressemblent pas et ne sont pas reliés entre eux de la même manière au niveau de chaque déduction, ils constituent l'essence même d'un avion et sont considérés comme

les éléments du modèle mental "avion". Si ces éléments ne sont pas présents dans la déduction formelle d'un individu, quels que soient sa forme et le nom qu'il lui donne (une hélice sous-entend un système de propulsion, un habillage des ailes ou du cockpit sous-entend un fuselage ...), ce n'est plus un avion. Le modèle mental "avion" est alors dénaturé : sans le cockpit, c'est un drone ; sans le système de propulsion, c'est un planeur ; sans ailes et avec des hélices sur le toit, c'est un hélicoptère.

La frontière entre le modèle mental et la déduction formelle, telle que définie par Johnson-Laird, est mince. De plus, l'influence de l'expertise de l'individu, telle qu'exprimée par Furlough, n'aide pas à construire un modèle mental valide, où les éléments du modèle nécessaires pour définir un concept sont tous présents. Un modèle mental permet donc de définir un concept quand ce dernier contient un certain nombre d'éléments de base indissociables du concept. Le modèle mental doit également limiter l'ajout d'éléments non essentiels et la modification, même mineure, d'éléments essentiels au concept. Par exemple, une personne décrivant un avion avec des rotors basculants permettant un décollage vertical ne dénature en rien le modèle mental "avion".

De nombreuses études ont été menées pour identifier et définir les modèles mentaux dans tous les domaines d'application possibles : l'apprentissage du fonctionnement d'un équipement inconnu [Kieras et Bovair, 1984], l'apprentissage par le dessin en biologie [Quillin et Thomas, 2015 ou encore le développement des compétences de joueurs sur des jeu répétitifs [Wasserman et Koban, 2019]. Comme évoqué précédemment, c'est également le cas du domaine informatique : les modèles mentaux permettent de définir un concept de manière précise, tout en laissant assez de libertés pour prendre en compte les évolutions et adaptations des environnements hétérogènes rencontrés dans ce domaine.

Dans ce mémoire, le modèle mental sera considéré comme une manière de voir un concept afin de comprendre comment ce dernier fonctionne. **Un modèle mental doit permettre de répondre à deux questions** :

1. Comment cela fonctionne-t-il ?
2. Pourquoi cela fonctionne-t-il de cette manière ?

2.1.2 Modèles mentaux & informatique

L'un des défis les plus importants dans le domaine informatique est la compréhension des méthodes menant à l'écriture de programmes. Que ce soit aux niveaux fonctionnel, architectural, technique ou encore environnemental, les méthodes possibles sont multiples et ne sont pas toujours adéquates ou ne répondent pas forcément aux challenges auxquels le produit final doit répondre. De nombreuses études ont été menées pour identifier, catégoriser et décrire des procédures-types permettant au monde informatique de disposer d'un catalogue de méthodes simples et adaptables à toutes situations. C'est le cas des études de Pennington [Pennington, 1987b], Letovsky [Letovsky, 1987], Soloway [Soloway et Ehrlich, 1984], Von Mayrhauser [Von Mayrhauser et Vans, 1995] et Canas [Canas et al., 1994], qui définissent les bases de la construction d'un programme informatique soutenu par des modèles mentaux tels que le "Text-based", "Situation", "Program", "Bottom-up" ou "Top-down". Les résultats de ces études sont ensuite utilisés pour tenter d'expliquer les problèmes de conception que peuvent rencontrer les étudiants lors du processus d'apprentissage [Kaczmarczyk et al., 2010] ou les développeurs dans l'utili-

sation de concepts de base, comme la définition d'un objet [Holland et al., 1997], ou encore pour tenter de comprendre l'influence de certaines techniques, comme les méthodes Agile [1] dans les procédures de maintenance d'un projet [Schmidt et al., 2014].

2.2 Approche

Le point de départ de cette étude fait écho à un besoin de plus en plus difficile à satisfaire dans les entreprises : intégrer de nouveaux arrivants, novices dans le domaine ou experts dans d'autres domaines, dans une équipe. Le besoin d'uniformisation des compétences et techniques d'apprentissage est de plus en plus inévitable. Pour tenter de répondre à ce besoin, les entreprises proposent de plus en plus des formations dès l'arrivée de nouvelles recrues, afin de les aider à se préparer au mieux aux attentes des équipes et clients avec qui elles devront travailler. La diversité des domaines d'application des entreprises et des projets au sein de celles-ci demande de plus en plus de compétences et de connaissances dans des domaines de plus en plus variés et complexes. Néanmoins, un postulat peut être émis : **La diversité des compétences et des connaissances nécessaires à l'aboutissement d'un projet peut être vue comme une multiplication de solutions différentes à une série de problèmes déjà connus et identifiés**. Dès lors, il serait logique de privilégier la recherche pour identifier ces problèmes, ainsi que leurs solutions possibles, plutôt que de directement tenter de s'orienter vers une approche répondant aux besoins d'une équipe ou d'un projet en particulier. Pouvoir s'assurer que ces problèmes soient connus et maitrisés par les novices pourrait faciliter leur intégration dans les équipes déjà en place et leur permettre d'évoluer plus rapidement dans leurs domaines respectifs.

Cet état de l'art se veut une revue systématique de la littérature concernant l'étude des modèles mentaux, leur identification et leur application dans des domaines informatiques proches du monde de l'entreprise et plus particulièrement dans le processus de compréhension d'un programme. Les études incluant les modèles mentaux dans leur méthode de recherche vont permettre d'identifier plus facilement les modèles de solution communs aux problèmes identifiés, indifféremment du domaine ciblé.

De cette approche, basée sur le postulat de départ, découlent plusieurs questions :

- (1) Quels sont les problèmes connus et identifiés ?
- (2) Comment identifier les modèles mentaux utilisés lors de l'élaboration d'une solution ?
- (3) Existe-t-il plusieurs modèles mentaux pouvant amener une solution à un problème unique ?
- (4) Existe-t-il des modèles mentaux pouvant amener à un résultat erroné, mais satisfaisant pour le domaine visé ?

Les études apportant des éléments de réponse à ces questions sont légion. Letovsky a tenté d'identifier les modèles mentaux utilisés par les novices et les experts lors de l'ajout de nouvelles fonctionnalités dans un programme donné,

1. https://agilemanifesto.org/

le problème adressé ici étant de la rétro-ingénierie sur base d'une liste de données [Letovsky, 1987]. Von Mayrhauser et Vans ont tenté d'identifier plusieurs modèles mentaux et leur possible application dans différents domaines, en se focalisant principalement sur les modèles mentaux liés à la compréhension d'un problème donné [Von Mayrhauser et Vans, 1995, 1996, 1998 ; von Mayrhauser et Vans, 1994]. Bien que les problèmes étudiés soient assez restreints et ne s'attardent que sur un seul aspect de la programmation (l'ajout ou la correction du comportement d'une méthode), il permet déjà d'identifier plusieurs modèles mentaux tels que le "Top-down" ou le "Bottom-up", qui seront décrits plus loin dans cet état de l'art (voir 2.3, Observations page 14). La nature opposée de ces deux modèles mentaux donne déjà une réponse à la question (3) sur l'unicité d'une solution à un problème donné. Holland, Griffiths et Woodman [Holland et al., 1997] ainsi que Kaczmarczyk, Petrick, East et Herman [Kaczmarczyk et al., 2010] ont, quant à eux, étudié les problèmes de compréhension et d'application de certains modèles mentaux liés au concept d'objet et d'assignation de valeurs à des variables primitives. Ils ont pu déterminer que certains problèmes de compréhension ne provenaient pas de l'application d'un modèle mental en particulier, mais relevaient plus d'un problème de compréhension lié à la sémantique du langage de programmation ou à un manque de connaissance du domaine objet du programme. Enfin, Storey, Fracchia et Müller [Storey et al., 1999] mettent en évidence l'importance d'adresser correctement un problème et la difficulté d'identifier les modèles mentaux qui entreront dans le processus de résolution du problème.

Une grande partie de ces études base leurs résultats sur une catégorisation des participants en novice ou expert du domaine [Bergantz et Hassell, 1991 ; Burkhardt et al., 1997 ; Romero et du Boulay, 2004]. Cette catégorisation aide à identifier d'éventuels problèmes dans la manière d'appréhender les questions posées spécifiques au domaine sujet de l'étude. Cependant, chaque étude définit les termes novice et expert sur base de ce domaine. Huang et Cakmak [Huang et Cakmak, 2015], de même que Ofoleta [Ofoleta, 2015] ou Wiedenbeck, Fix et Scholtz [Wiedenbeck et al., 1993] définissent comme novices des personnes qui n'ont aucune connaissance technique dans l'informatique et les comparent à des experts, identifiés comme des étudiants en fin de cycle de 5 ans en informatique. Ce choix est justifié par le sujet des études qui demandent peu de compétences techniques et plus de réflexion logique sur l'enchaînement des tâches. Dans d'autres études, Burkhardt, Détienne et Wiedenbeck [Burkhardt et al., 2002 ; Burkhardt et al., 1998] définissent les novices comme étant des étudiants en fin de cycle, supposés maitriser la technologie utilisée, et les comparent à des développeurs avec plusieurs années d'expérience dans la technologie et maitrisant également le contexte fonctionnel du projet mis en évidence dans l'étude. Ce choix est justifié par le domaine d'étude qui compare les approches prises par les novices et les experts pour résoudre un problème précis lié à une problématique fonctionnelle et non technique. La même approche est prise par Corritore et Wiedenbeck [Corritore et Wiedenbeck, 1999], mais, dans leur étude, les novices sont définis comme étant des développeurs ne maitrisant pas le paradigme de programmation sujet de l'étude et les comparent à des experts développant depuis plusieurs années dans le paradigme visé.

Bien que les études de Huang et Cakmak, de Ofoleta et de Wiedenbeck, Fix et Scholtz apportent des éclairages intéressants sur la construction des modèles mentaux et leur application, elles sont en dehors du périmètre visé par cet état de l'art, car intégrant un public-cible trop éloigné du monde de l'entreprise. Par contre, les études telles que celles de Burkhardt, Détienne et Wiedenbeck

et Corritore et Wiedenbeck entrent en adéquation avec le domaine visé par cet état de l'art.

Le focus est donc mis sur (1) les études dont le domaine d'application est le monde de l'entreprise ou équivalent, (2) sur une approche plus technique que psychologique, (3) avec un public cible déjà sensibilisé aux techniques de développement de solutions informatiques (des études comme celle de Muhamad [Muhamad, 2012], étudiant les modèles mentaux dans un but de recrutement, sont considérées hors scope, car la dimension des modèles mentaux étudiés se rapproche plus de la psychologie et du social que du technique).

2.3 Observations

Avant de pouvoir étudier les résultats de l'influence des modèles mentaux sur les processus d'apprentissage et de mise à niveau dans les entreprises, il est nécessaire d'identifier quels types de modèles mentaux sont repris dans les différentes études portant sur le processus de compréhension de programmes.

Comme cela a été dit précédemment, de nombreuses études ont permis de mettre en évidence les principaux modèles mentaux aidant à la représentation d'un programme informatique (point 2.1.2 Modèles mentaux & informatique page 11). Ces études se basent principalement sur la représentation mentale multicouche du texte définie par Kintsch et Van Dijk [Van Dijk, Kintsch et al., 1983] et synthétisée par Corritore et Wiedenbeck [Corritore et Wiedenbeck, 1999] :

- (1) **La représentation textuelle** (ou la forme de surface) correspond à la forme littérale du texte. C'est une représentation incomplète sauf si le texte est déjà connu.
- (2) **La base textuelle propositionnelle** est composée des propositions dans le texte et des relations. C'est une représentation linguistique isomorphe à la structure du texte, supposée être construite automatiquement lors de la lecture.
- (3) **Le modèle de situation** (ou domaine) représentant la situation dans le monde que décrit le texte. Ce n'est pas une représentation du texte, mais du sens du texte. Le modèle de situation n'est pas construit automatiquement et sa construction dépend du lecteur qui fait des inférences à partir du texte et de ses connaissances sur le domaine du texte [Ehrlich et Tardieu, 1993 ; Mills et al., 1995].

Les principaux modèles mentaux sont ceux listés ci-dessous :

- **Letovsky** [Letovsky, 1987] définit **un modèle de compréhension de haut niveau** comportant trois éléments principaux : (1) Une base de connaissances : une expertise en programmation, une connaissance du domaine objet du programme, ... ; (2) Un modèle mental (représentation interne) caractérisé par une sous-couche de spécification, une sous-couche d'implémentation et une sous-couche d'annotation et (3) Un processus d'assimilation, soit ascendante, soit descendante, permettant faire les liens entre les deux autres éléments.
- **Soloway** [Soloway et al., 1988 ; Soloway et Ehrlich, 1984] propose **un modèle descendant**, généralement appliqué lorsque le code ou le type de code est familier, et divisé en trois types de plan : (1) Stratégique (stratégie générale de programme ou d'algorithme) ; (2) Tactique

(stratégie locale pour résoudre le problème) et (3) La mise en œuvre (pour mettre en œuvre les plans tactique et stratégique, dépendant du langage de programmation).

— **Pennington** [Pennington, 1987b] présente **la compréhension ascendante** avec deux représentations mentales : (1) Le modèle de programme, qui est une abstraction du programme de contrôle, construit de bas en haut via des balises et qui identifie le premier bloc de code, et (2) Le modèle de situation, qui nécessite une connaissance des domaines du monde réel et est complet une fois que l'objectif du programme est atteint.

— **Von Mayrhauser** [Von Mayrhauser et Vans, 1995, 1996, 1998 ; von Mayrhauser et Vans, 1994] intègre **le modèle descendant de Soloway et le modèle ascendant de Pennington** et base son approche sur le fait que le modèle mental utilisé par les programmeurs dépend de leur connaissance du domaine, de leur connaissance de base et de leur expérience de programmeur.

Von Mayrhauser [Von Mayrhauser et Vans, 1995] répertorie les différentes approches et définit des critères d'évolution des différents modèles mentaux proposés par ses collègues. Il propose ensuite le tableau 2.1 qui résume son approche et sera utilisé dans cette étude comme base pour l'identification des modèles mentaux de compréhension de programme (voir point 2.3.1). Le tableau 2.1 liste les différents modèles mentaux utilisés par les développeurs suivant le niveau d'abstraction du problème à résoudre, en se basant sur les critères suivants :

— Les structures statiques : connaissances préalables non liées au progamme ou représentations mentales du programme actuel.

— Les structures dynamiques : représentations mentales construites à l'aide des connaissances extérieures au programme.

— Le type d'expérimentation : type d'expérimentation menée pour valider chaque modèle étudié.

Model criterion	**Abstraction level**	**Letovsky**	**Shneiderman and Mayer**	**Brooks**	**Soloway, Adelson and Ehrlich**	**Pennington**	**Integrated model**
Static : Knowledge structures	Low	Knowledge base	Syntactic knowledge	Programming domain	Implementation plans	Text-structure knowledge	Program model structures
	Intermediate	Knowledge base	Semantic knowledge	Intermediate domain	Tactical plans	Plan knowledge	Situation structures
	High	Knowledge base	Semantic knowledge	Problem domain	Strategic plans		Top-down structures
Static : Mental representations	Low	Implementation layer	Working memory : Low-level concepts	Hypotheses and subgoals	Pans/schemas	Program model	Program model
	Intermediate			Hypotheses and subgoals	Pans/schemas	Situation model	Situation model
	High	Specification layer	Working memory : Low-level concepts	Hypotheses and subgoals	Pans/schemas		Top-down model
Dynamic : Porcesses	Direction	Top-down or bottom-up	Top-down or bottom-up	Top-down	Top-down	Bottom-up	Top-down or bottom-up
Experimentation	Type	Small-scale code experiments	Small-scale code experiments	Self-observation experiments	Small-scale code experiments	Small-scale code experiments	Large-scale code experiments

Table 2.1 – Model evaluation criteria [Von Mayrhauser et Vans, 1995]

Von Mayrhauser considère les différents points de vue de ses collègues et tente une définition générale du modèle mental d'un logiciel : *"Le modèle men-*

tal est une représentation fonctionnelle interne du logiciel considéré. Il contient des entités statiques, telles que des structures de texte, des blocs, des plans, des hypothèses, des balises et des règles de discours. Les plans de niveaux supérieurs s'affinent en plans ou blocs plus détaillés. Chaque bloc, à son tour, représente une abstraction de niveau supérieur d'autres blocs ou structures de texte. La construction d'un bloc combine plusieurs comportements dynamiques, y compris des stratégies, des actions, des événements et des processus."(traduction). Il va également conclure son étude sur plusieurs observations partagées par de nombreuses autres études :

1. La plupart des études ne se concentrent que sur une compréhension générale d'un programme ou sur une partie du code peu représentative de la réalité du terrain (cette même conclusion est observée dans les études de Burkhardt, Détienne et Wiedenbeck [Burkhardt et al., 1997], Navarro-Prieto et Canas[Navarro-Prieto et Canas, 2001] ou Bidlake, Aubanel, et Voyer [Bidlake et al., 2020]).
2. Les résultats d'études apparaissent comme des composants de résultats plus larges (c'est le cas pour le méta-model intégré, proposé dans le tableau 2.1, et qui fait écho aux études sur les modèles mentaux partagés [Klimoski et Mohammed, 1994 ; Van den Bossche et al., 2011]).
3. Les problématiques de maintenance adaptative et perfective ne sont pas correctement adressées (observation faite également dans les études de Von Mayrhauser et Vans [Von Mayrhauser et Vans, 1995] ou Corritore et Wiedenbeck [Corritore et Wiedenbeck, 1999]).

Enfin, l'étude de Von Mayrhauser permet de mettre en évidence la majorité des modèles mentaux qui seront étudiés au cours des décennies suivantes. Les modèles de compréhension de programme qui sont identifiés dans cette étude sont au nombre de cinq : (1) "Text-based" ; (2) "Situation" ou "Domaine" ; (3) "Program" ; (4) "Top-down" et (5) "Bottom-up". Ces derniers sont tantôt opposés, tantôt complémentaires et ne dépendent pas du problème posé, mais du programmeur qui fait face à ce problème. A contrario, certains modèles mentaux cités dans l'étude de Von Mayrhauser seront vite abandonnés par la communauté scientifique. C'est le cas du modèle mental proposé par Brooks [Brooks, 1983], qui construit son modèle dans une seule direction, du problème vers le programme, mais également se base exclusivement sur des hypothèses de changement, sans donner de structure claire aidant à la reconstruction ou à l'évolution du modèle mental. Le modèle mental de Shneiderman et Mayer [Shneiderman et Mayer, 1979] est également vite écarté, car il ne propose pas assez d'indices sur la construction de la base de connaissance nécessaire à la construction mentale du modèle.

2.3.1 Modèles mentaux de compréhension de programme

Comme cité précédemment, les modèles mentaux de compréhension de programme les plus étudiés sont au nombre de cinq : (1) "Text-based" ; (2) "Situation" ou "Domaine" ; (3) "Program" ; (4) "Top-down" et (5) "Bottom-up". Ces derniers ne sont toutefois pas sur un même pied d'égalité lorsqu'il est question de les appliquer pour résoudre un problème. Cela dépend de différents critères étroitement liés à l'expérience des programmeurs, leurs habitudes, ainsi que l'environnement dans lequel ils évoluent. C'est Letovsky [Soloway et al., 1988 ; Soloway et Ehrlich, 1984] qui est le premier à mettre en évidence l'interdépendance de certains modèles mentaux et l'importance du vécu du programmeur dans le processus de compréhension de programme. Il définit ce processus d'opportuniste : c'est le programmeur qui fait le choix de procéder d'une manière ou d'une autre (de manière ascendante ou descendante

dans les études de Litovsky) en se basant sur le meilleur rendement estimé en termes de prise de connaissance et d'utilisation des acquis. Chaque modèle mental de compréhension de programme ayant ses caractéristiques propres, bien comprendre leur structure permettrait de comprendre ce qui pousse le programmeur à préférer un modèle plutôt qu'un autre et à corriger ce choix quand ce dernier n'est pas identifié comme le meilleur choix dans une situation donnée.

Text-based : Le modèle mental "Text-based" peut être considéré comme un modèle mental de base pour les autres modèles mentaux étudiés dans cette section. Sa définition découle directement de la définition générale des modèles mentaux proposée par Johnson-Laird [Johnson-Laird, 1980 ; Johnson-Laird, 1983] et est affinée par les réflexions de Kintsch et Van Dijk [Van Dijk, Kintsch et al., 1983] et leur représentation mentale multicouche du texte (voir 2.1.1 Définition & Origines page 9]). Cette représentation permet d'identifier, de comprendre et de relier entre eux les flux de contrôles [2] nécessaires à la compréhension du programme [Pennington, 1987a, 1987b].

Ce modèle mental se base donc sur la reconnaissance de structures textuelles, qui peuvent être la structure d'un programme dans son entièreté, des points de syntaxe liés au langage de programmation ou des conventions de nommage ou de programmation. Cette reconnaissance de structures textuelles est donc fortement liée à l'analyse de code et à la documentation associée, mais elle ne permet pas à elle seule d'expliquer complètement les mécanismes permettant aux programmeurs de comprendre un programme. Suivant les préférences du programmeur, démarrer la compréhension d'un programme par le code ou par la documentation liée amène une réflexion et une construction mentale différente. De plus, la direction prise lors de la création de lien entre les différentes structures peut également amener des constructions mentales différentes. C'est la raison pour laquelle le modèle mental "Text-based" est considéré comme un modèle mental de base. Il n'est pas assez précis pour expliquer l'entièreté des mécanismes qui entrent en considération lors de la compréhension de programme et doit être associé à d'autres modèles mentaux, tels que "Program" et "Situation", ainsi que par les modèles mentaux directionnels "Top-down" et "Bottom-up".

Le modèle mental "Text-based" peut être également mis en concurrence avec d'autres modèles mentaux, tels que les modèles mentaux de type visuel, qui semblent être une alternative intéressante au modèle mental "Text-based". Plusieurs études prennent ces modèles mentaux de type visuel comme sujets de base. C'est le cas de l'étude de Navarro-Prieto et Canas [Navarro-Prieto et Canas, 2001] qui met principalement en concurrence le modèle mental proposé par Pennington [Pennington, 1987a, 1987b], basé sur la représentation textuelle, et les recherches sur des aides visuelles dans l'apprentissage du texte proposé par Mayer [R. E. Mayer, 1994 ; R. E. Mayer et Gallini, 1990]. Contrairement au modèle mental "Text-based", qui facilite la détection des flux de contrôle lors du processus de compréhension d'un programme, un modèle mental de type visuel facilite la détection des flux de données [3]. Navarro-Prieto

2. Flux de contrôle : *"[...] la séquence d'instructions dans le programme et certains mots-clés fournissent des informations sur la séquence dans laquelle les instructions du programme seront exécutées. Ce type d'information est appelé le flux de contrôle du programme et comprendre un programme nécessite de comprendre son flux de contrôle"*(traduction) [Pennington, 1987b]

3. Flux de données : *"[...] concerne les changements ou les constances dans la signification ou la valeur associée aux noms d'objets du programme tout au long du programme"* (traduction) [Pennington, 1987b]

et Canas [Navarro-Prieto et Canas, 2001] concluent en partie qu'un modèle mental de type visuel serait plus pertinent pour la conception de logiciels éducatifs et dans l'enseignement de langages de programmation. Ces conclusions semblent partagées par plusieurs études, notamment celles de Butcher [Butcher, 2006] et Quillin et Thomas [Quillin et Thomas, 2015].

Top-down : Le modèle mental "Top-down", comme le modèle mental "Bottom-up", est un modèle mental de soutien au processus de compréhension de programme. Il n'est pas directement lié au type de programme ou au langage de programmation utilisé, ni même au type de modèle mental de base. Il peut donc être appliqué avec le modèle mental "Text-based" ou les modèles mentaux de type visuel. C'est un processus de compréhension descendant : le programmeur utilise une procédure de décomposition de la situation en éléments connus et, dans le meilleur de cas, maitrisés. Ce mécanisme de décomposition est généralement appliqué aux situations familières, car il requiert une identification rapide de structures complexes, structures qui ne nécessitent pas plus de décompositions ou de détails pour en déduire leur finalité.

Soloway, Adelson et Ehrlich [Soloway et al., 1988] identifient trois plans lors du mécanisme de compréhension de programme :

— (1) **Plan stratégique**, à savoir les stratégies globales utilisées dans le programme

— (2) **Plan tactique**, à savoir les stratégies locales pour la résolution d'un problème donné

— (3) **Plan de mise en oeuvre**, dépendant du langage de programmation utilisé (contrairement aux autres plans) et mettant en oeuvre les plans tactiques

Ils établissent donc une hiérarchie d'objectifs et de plans qui sont généralement liés par un raisonnement superficiel. En d'autres termes, le processus de compréhension de programme part de macrostructures constituant le plan considéré et qui vont être décomposées en microstructures. Ces microstructures sont identifiées grâce à l'observation de balises reconnues par le programmeur comme composant fondamental de ces microstructures. De plus, ce processus d'identification peut être récursif, car ces microstructures peuvent être, à leur tour, des macrostructures du plan inférieur du mécanisme de compréhension de programme. Le raisonnement est dit superficiel, car la reconnaissance de balises ne suffit pas à garantir l'identification de microstructures de manière certaine. Le modèle mental "Top-down" ne requiert pas une identification certaine des microstructures constituant les macrostructures des plans hiérarchique stratégique et tactique.

Certaines études, comme celles de Parkin [Parkin, 2004] et Balijepally, Nerur et Mahapatra [Balijepally et al., 2015], montrent que le modèle mental "Top-down" est majoritairement appliqué par les programmeurs ayant un certain niveau d'expertise. De manière générale, les plans de niveaux supérieurs demandent une bibliothèque de connaissance conséquente pour pouvoir identifier facilement des macrostructures globales. Ce constat est confirmé par l'observation d'une certaine hésitation chez certains programmeurs expérimentés, lorsque l'objet du programme ou le problème posé touche à des sujets peu familiers. Dans ces cas, les programmeurs ont tendance à appliquer le modèle mental "Bottom-up" [Von Mayrhauser et Vans, 1998].

Bottom-up : Le modèle mental "Bottom-up" est, comme le modèle mental "Top-down", un modèle mental de soutien au processus de compréhension de programme. Contrairement au modèle mental descendant "Top-down", le modèle mental "Bottom-up" est un modèle mental ascendant. Ce modèle mental va tout simplement inverser la hiérarchie des objectifs et des plans définis dans le modèle mental descendant. Par exemple, les balises aidant à l'identification des microstructures du plan de mise en oeuvre sont des balises de base du langage de programmation utilisé (boucles, conditionnelles, variables ...). Ces microstructures sont ensuite reliées entre elles pour constituer des macrostructures (procédures, structures, objets ...) qui à leur tour pourront être identifiées soit comme microstructures du plan hiérarchique supérieur, soit comme microstructure du même plan (méthodes utilitaires, librairies externes ...).

Bien que Pennington [Pennington, 1987a, 1987b] ne définisse pas le modèle mental "Bottom-up" en termes de hiérarchie d'objectifs et de plans, il utilise le principe de modèle mental ascendant pour identifier et définir les modèles mentaux "Program" et "Situation". Quel que soit le modèle mental utilisé ("Program" ou "Situation"), Pennigton constate néanmoins que le modèle ascendant est le modèle de prédilection des programmeurs n'ayant pas une grande expertise dans le langage ou paradigme de programmation, ou dans le domaine technique ou fonctionnel lié au programme cible. Ce même constat est établi par plusieurs études plus récentes qui identifient la hiérarchie d'objectifs et de plans dans l'application du modèle mental "Bottom-up" [Balijepally et al., 2015 ; Burkhardt et al., 1998 ; Von Mayrhauser et Vans, 1998].

Program : Le modèle mental "Program" est l'un des modèles les plus utilisés lors du processus de compréhension de programmes. Il se base sur la reconnaissance de structures textuelles, comme le modèle mental "Text-based", et plus particulièrement sur les structures proposées par le code du programme : variables, boucles, conditions, fonctions de base, structures simples et complexes, telles que les listes, les dictionnaires ou encore les objets, entre autres.

L'application de ce modèle mental se fait généralement de manière ascendante, comme le confirment les études de Pennington [Pennington, 1987a, 1987b]. Ce dernier décrit le modèle mental "Program" comme la première représentation mentale construite par les programmeurs lorsqu'ils font face à un code inconnu. Ils construisent une abstraction du programme par le biais de flux de contrôle, de bas en haut, grâce à la reconnaissance de balises et de blocs de code connus (ou microstructures connues). Le point de départ de ce modèle mental se situe donc au niveau des connaissances existantes du programmeur en matière de code et structure de code. C'est le plan de mise en oeuvre défini par Soloway, Adelson et Ehrlich [Soloway et al., 1988]. Ensuite, le programmeur fait évoluer les microstructures identifiées en macrostructures, pour améliorer sa compréhension du programme et la compréhension du domaine du programme. Il progresse donc vers le plan tactique de Soloway, Adelson et Ehrlich. Pour Pennington, lorsque la compréhension du domaine du programme arrive à maturité, l'objectif du modèle mental "Program" semble atteint. La suite du processus de compréhension du programme est alors déléguée à l'application d'un autre modèle mental : le modèle mental "Situation".

Bien que le modèle mental "Program" soit généralement appliqué de manière ascendante, certaines études, comme celle de Hmelo-Silver et Pfeffer

[Hmelo-Silver et Pfeffer, 2004] et celle de Wiedenbeck, Fix et Scholtz [Wiedenbeck et al., 1993] ont montré qu'avec une certaine connaissance du domaine du programme, les programmeurs avaient tendance à appliquer ce modèle mental de manière descendante. L'identification de macrostructures, voire de stratégies locales du plan tactique, est plus aisée pour les programmeurs connaissant le domaine du programme. Le point de départ du processus de compréhension de programme se situe donc à un niveau intermédiaire. Les études montrent que les programmeurs ont alors tendance à appliquer le modèle mental "Program" de manière descendante uniquement lorsque l'action à réaliser sur le programme nécessite un niveau de compréhension plus détaillé, mais continuent à appliquer la manière ascendante pour conforter la compréhension globale du programme.

Situation : Le modèle mental "Situation" est, comme le modèle mental "Program", l'un des modèles mentaux les plus utilisés lors du processus de compréhension de programme. Il est généralement appliqué de manière ascendante. Contrairement au modèle mental "Program", ce modèle mental a besoin d'une certaine compréhension du domaine du programme, telle que la structure du programme et les fonctionnalités génériques, pour être correctement appliqué. Pennington [Pennington, 1987a, 1987b] décrit le modèle mental "Situation" comme la suite logique au modèle mental "Program" dont l'objectif est d'acquérir une compréhension du domaine du programme. Le modèle mental "Situation" n'est complet que lorsque l'objectif du programme est atteint, en d'autres termes, lorsque le plan stratégique de Soloway, Adelson et Ehrlich [Soloway et al., 1988] est compris. Quant au point de départ de ce modèle mental, il se situe là où le modèle mental "Program" se termine : au niveau du plan tactique de Soloway, Adelson et Ehrlich. Le modèle mental "Situation" se concentre principalement sur la compréhension des flux de données, via un processus de regroupement et de segmentation, tandis que le modèle mental "Program" se concentre plus sur les flux de contrôle.

2.3.2 Modèles mentaux partagés

Les études citées dans la section précédente et qui ont amené à la l'identification des modèles mentaux "Text-based", "Top-Down", "Bottom-up", "Program" et "Situation" ont toutes un point commun : elles concernent le processus de compréhension d'un programme pour un programmeur, seul face à un problème spécifique à résoudre, dans un programme ou un paradigme de programmation qui lui est inconnu. Mais que se passe-t-il lorsque ce même programmeur se retrouve à devoir résoudre un problème qui nécessite l'interaction avec d'autres programmeurs au sein d'une équipe ? Le niveau d'expertise de chaque programmeur de l'équipe, mais également le niveau de compréhension du programme et du domaine du programme, sont différents. Chaque programmeur appliquera un modèle mental différent, celui qui lui convient le plus pour appréhender le problème posé. Pour réduire ces différences, il est donc essentiel de définir des mécanismes communs qui permettront aux programmeurs d'améliorer le processus de compréhension de programme de manière individuelle et groupée. Les modèles mentaux vont répondre à ce besoin.

Klimoski et Mohammed [Klimoski et Mohammed, 1994] définissent les modèles mentaux partagés comme "la représentation mentale chevauchante des membres de l'équipe des éléments clés de l'environnement de travail de l'équipe" (traduction). Bien que cette définition soit assez vague au regard des

autres modèles mentaux cités dans la section précédente, c'est la définition la plus précise qui peut être donnée. L'hétérogénéité des environnements de travail, ainsi que la diversité des membres composant une équipe, font qu'il existe une multitude de modèles mentaux partagés qui, de plus, vont évoluer suivant la progression des membres de l'équipe ou de l'environnement de travail. Ce constat, largement détaillé par Van den Bossche, Gijselaers, Segers, Woltjer et Kirschner [Van den Bossche et al., 2011], rend l'identification d'un modèle mental partagé en particulier impossible. Cette ambiguïté, ou confusion dans l'identification des modèles mentaux partagés, est l'une des raisons pour lesquelles peu d'études traitent du sujet, comme l'expliquent Mohammed, Klimoski et Rentsch [Mohammed et al., 2000].

Les modèles mentaux partagés sont donc innombrables et difficiles à identifier, mais ils peuvent être catégorisés, comme le montre l'étude de Johnson, Lee, Lee, O'Connor, Khalil et Huang [Johnson et al., 2007]. Cette catégorisation peut être faite sur base de l'objectif attendu lors de l'étude de ces modèles mentaux. Ces objectifs sont : (1) la performance de l'équipe, (2) la maintenabilité du programme et (3) la polyvalence des membres de l'équipe. C'est l'objectif de performance de l'équipe qui est le plus plébiscité par les études [Brannick et al., 1997 ; Johnson et al., 2011 ; Schmidt et al., 2014].

Les modèles mentaux partagés sont peu étudiés et les études menées ne couvrent pas l'étendue des possibilités que ces modèles mentaux peuvent offrir. Ils ont pourtant une grande influence dans le processus de compréhension de programme. De ce fait, ils peuvent être considérés comme un acteur principal de ce processus de compréhension, au même titre que les modèles mentaux "Text-based", "Top-Down", "Bottom-up", "Program" et "Situation".

2.3.3 Essoufflement

Comme cité précédemment, de nombreuses études ont été menées concernant les modèles mentaux dans le domaine informatique. Pour la plupart, elles ont été menées principalement à la fin du 20ème siècle. Ce constat d'essoufflement sur ces dernières années est confirmé par Bidlake, Aubanel et Voyer [Bidlake et al., 2020]. Dans leur revue systématique, datant de 2019, ils prennent en compte toutes les études empiriques effectuées auprès de développeurs dans le domaine de la compréhension et la représentation interne du code écrit dans des langages formels. Un net déclin d'intérêt du monde scientifique concernant ce domaine est constaté. Comme le montre la figure 2.1, Bidlake, Aubanel et Voyer ont mené leur recherche sur 71 études et constatent qu'à partir des années 1970, et plus précisément suite à l'étude menée par Shneiderman [Shneiderman, 1976], l'intérêt sur l'étude des modèles mentaux va grandissant pour atteindre un pic dans les années 1990 (avec 39 études sur la décennie). Ensuite, cet intérêt va drastiquement diminuer : seules 12 études sont menées sur les deux décennies qui suivirent.

Bidlake, Aubanel et Voyer ont inclus dans leur recherche des études ne traitant pas uniquement des modèles mentaux, mais également de représentations mentales. Ils ont également inclus des études effectuées uniquement en milieu académique et non liées au monde de l'entreprise. C'est le cas des études de Shneiderman [Shneiderman, 1976], Adelson [Adelson, 1981 et Adelson, 1984], McKeithen, Reitman, Rueter et Hirtle [McKeithen et al., 1981] ou encore Weiser [Weiser, 1982] qui ont été menées avant la formalisation du concept de modèle mental et son adoption par le milieu informatique, mais étudient les représentations mentales induites par la compréhension du code

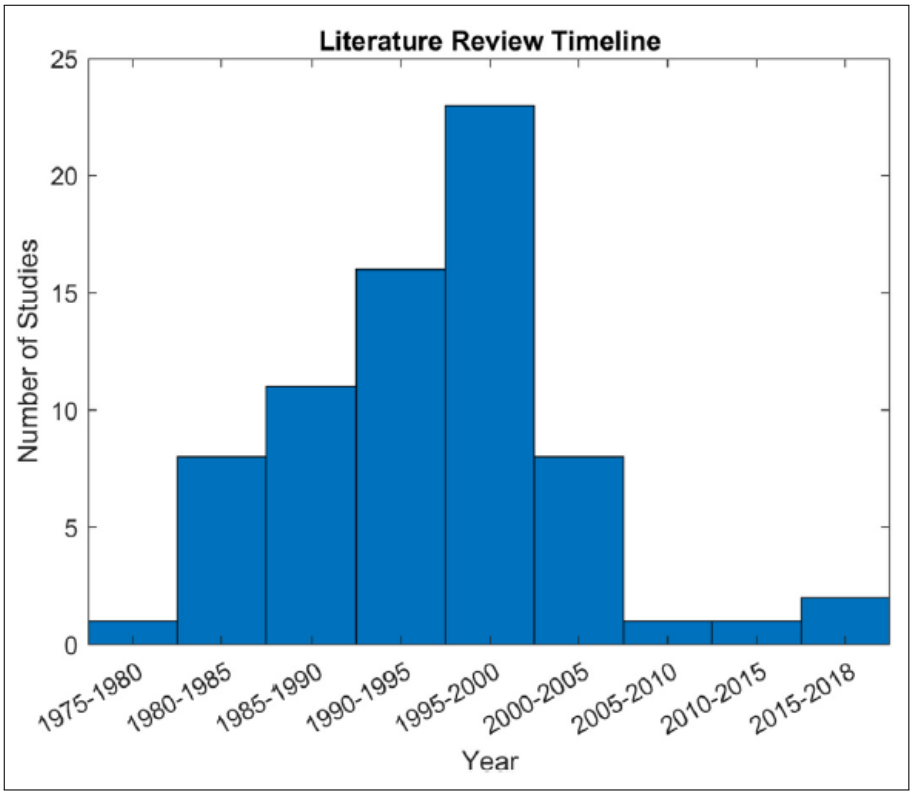


FIGURE 2.1 – Frequency of empirical studies on program comprehension.[BIDLAKE et al., 2020]

basé sur le texte ou sur l'environnement de travail (prémisses des modèles mentaux "Text-based" et "Situation"). Quant à Ramalingan [RAMALINGAM et WIEDENBECK, 1997], Corritore et Wiedenbeck [CORRITORE et WIEDENBECK, 1991] ou encore Wiedenbeck [WIEDENBECK et al., 1999], pour ne citer qu'eux, ils ont mené leurs études auprès d'étudiants en informatique ou auprès de personnes n'ayant aucune connaissance technique dans le domaine étudié.

Une grande partie des études incluses de cet état de l'art (près d'une vingtaine) sont également présentes dans l'article de Bidlake, Aubanel et Voyer. Bien que les résultats de la figure 2.1 Frequency of empirical studies on program comprehension.[BIDLAKE et al., 2020] page 22 ne se basent pas uniquement sur les études menées sur les modèles mentaux liés au processus de compréhension de programme, beaucoup d'autres similitudes dans les résultats ont été observées et un graphique similaire pourrait être tracé sur base des résultats de recherche présentés ici. L'une de ces similitudes est le traitement des modèles mentaux "Text-based", "Situation", "Program", "Top-down" et "Bottum-up". Bien que les deux états de l'art couvrent ces modèles mentaux et quelques autres dans l'article de Bidlake, Aubanel et Voyer, la difficulté de formaliser ces modèles mentaux dans des études à large spectre, ainsi que l'hétérogénéité des techniques proposées par le domaine informatique sont des freins à la popularisation de ce domaine d'étude. Le monde scientifique semble donc se désintéresser de ce domaine d'étude.

2.4 Analyse critique

Dans cet état de l'art, plusieurs problèmes ont été identifiés : (1) l'hétérogénéité du public-cible des différentes études, (2) les domaines de recherche restreints et parfois incompatibles d'une étude à l'autre et (3) la perte d'intérêt du monde scientifique par rapport à un domaine en constante évolution. Ces différents problèmes rendent non seulement difficile l'interprétation des résultats, étant donné qu'il n'y a pas de base d'étude commune, mais également ne permettent pas d'avoir une vision globale de la situation couvrant la majorité des domaines étudiés.

2.4.1 Hétérogénéité du public-cible

Le premier problème est l'hétérogénéité du public-cible des différentes études. De ce problème découle la difficulté de définir clairement des catégories de personnes étudiées. Les termes novices et experts sont utilisés dans la plupart des études et doivent être définis de manière unique suivant le public-cible choisi pour cette étude. D'une étude à l'autre, le panel de novices devient le panel d'experts et inversement.

Ce constat est également fait par Bidlake, Aubanel et Voyer [Bidlake et al., 2020] qui précisent : *"[...] malgré l'utilisation généralisée des catégories expert et novice pour décrire les programmeurs, aucune définition ou mesure commune de l'expertise n'a été élaborée ou adoptée. Même les études qui n'ont pas comparé les programmeurs experts et novices ont souvent utilisé ces catégories pour identifier le groupe de programmeurs qui ont été utilisés comme participants. Il n'y avait pas non plus de cohérence entre l'utilisation d'experts et d'expériences pour décrire les programmeurs et ces termes étaient souvent utilisés de manière interchangeable dans la même étude [*Barfield*, 1986 ;* Fix *et al., 1993 ;* Vessey*, 1987]"* (traduction).

Ce problème est fortement lié à la manière dont le sujet de l'étude a été appréhendé et à son caractère limité, tel que les limites que peut proposer un projet en entreprise dans un secteur d'activité précis ou, simplement, les limites d'un projet basique où l'entièreté du processus doit être maitrisée par les examinateurs.

Le problème est aussi lié à des prémisses assumées, où on décide d'ignorer certaines compétences ou de donner des compétences au public-cible, alors que ces compétences peuvent avoir une influence sur le résultat final. Par exemple, Holland, Griffiths et Woodman [Holland et al., 1997], dans leur étude sur les conceptions erronées en orienté-objet, catégorisent comme novice un développeur expérimenté en développement procédural. Or, il est plus que probable qu'un expert dans un domaine proche du domaine étudié aura des réflexes qu'un développeur n'ayant que des connaissances de base n'aura pas. Et inversement, un développeur au fait des dernières innovations ou simplement fraichement initié à certaines techniques, aura des réflexes qu'un expert en "vieilles" technologies n'aura pas.

Le problème est d'autant plus visible quand le public-cible d'une étude s'élargit aux personnes n'ayant aucune compétence informatique de base. Dans ce cas, ces personnes sont désignées comme novices et toute personne ayant des connaissances en informatique, quelles qu'elles soient, est qualifiée d'experte [Huang et Cakmak, 2015 ; Soloway et Ehrlich, 1984]. Il existe donc

presque autant de définitions de novices et d'experts qu'il y a de types de public-cible différents.

Ce problème rend la possibilité de catégoriser de manière générique les personnes participant aux études très difficile. Cette difficulté est d'autant plus accrue quand on tient compte du fait qu'il peut y avoir plusieurs niveaux dans la définition de novices et d'experts. Il est donc nécessaire de proposer une méthode permettant de catégoriser les personnes participant aux études qui se baserait sur des caractéristiques communes et vérifiables et qui engloberait la majorité des publics-cibles rencontrés jusqu'à présent.

2.4.2 Domaines d'études spécialisés

Le second problème relatif aux domaines d'études spécialisés, dont les différences sont telles qu'il est parfois impossible de les comparer, est tout aussi problématique et difficile à résoudre que le premier.

La plupart des études rencontrées lors de l'état de l'art restreignent leur domaine à l'application d'un concept ou un paradigme de programmation, à la résolution d'un problème ou encore restreignent leur public-cible à une équipe ou à une entreprise limitée à un secteur d'activité. Cela permet bien évidemment d'avoir une photo précise et de tirer des conclusions sur le domaine étudié, mais cela rend la généralisation de certains concepts très difficile.

Comme vu précédemment dans l'état de l'art, les études sur les modèles mentaux en programmation procédurale sont nombreuses et donnent une idée précise de l'utilisation des modèles mentaux utilisés à l'époque (fin des années 80, début 90). Ensuite, avec la popularisation des autres paradigmes de programmation, des études se penchant sur l'orienté-objet ou la programmation logique ont été menées, mais très peu ont comparé les résultats obtenus avec les études précédentes et aucune n'a repris le même domaine qu'une étude précédente. Cette différence s'explique par le fait que le domaine des études relatives à la programmation procédurale est trop spécifique pour être repris dans les études axées sur un autre paradigme de programmation. Souvent, le problème choisi pour illustrer le domaine met en évidence un concept ou un modèle mental lié uniquement au développement procédural, qui ne serait plus d'actualité dans un autre paradigme. Il est donc nécessaire d'identifier les modèles mentaux transversaux, qui pourraient être applicables à la majorité des domaines, et de déterminer ensuite un moyen de les mettre en application dans des études plus spécialisées.

Une partie de solution à la problématique a été fournie par l'état de l'art, qui a dégagé plusieurs modèles mentaux transversaux liés à l'approche globale d'un problème à résoudre et indépendants du domaine étudié. Il est question ici des modèles mentaux "Text-based", "Situation", "Program", "Top-down" et "Bottom-up". De plus, le modèle mental Shared Mental Model est également pris en compte. Ce dernier a un caractère abstrait qui, malgré une conception standard, diverge fortement suivant le contexte dans lequel il est mis en œuvre. Cette sélection de modèles mentaux transversaux n'est malheureusement pas suffisante pour dégager des conclusions génériques, car ces modèles sont liés uniquement à l'approche d'un problème. Il reste donc la problématique des modèles mentaux pour la résolution d'un problème, qui est généralement fortement liée au contexte dans lequel le problème est posé. Cette problématique est très large et ne semble pas solvable par une seule étude.

2.4.3 Essoufflement des études sur le sujet

Le dernier problème, l'essoufflement des études sur le sujet, fait écho au précédent problème : il est difficile d'étudier les modèles mentaux dans un cadre générique, quand le nombre de contextes d'utilisation de ces derniers s'élargit d'année en année. Le nombre de paradigmes et de langages de programmation, ainsi que le nombre grandissant de méthodologies, mais également le nombre grandissant de solutions informatiques dans des domaines d'applications différents, fait que l'étude des modèles mentaux est de plus en plus difficile. L'état de l'art met en évidence une diminution des études réalisées dès le début des années 2000, ce qui correspond à la popularisation des développements orienté-objet en entreprise. Cette popularisation a amené son lot de modèles mentaux et de contextes qui ont rendu l'analyse des modèles mentaux transversaux moins évidents et ont donné des résultats moins exploitables et généralisables. L'effort pour une généralisation des concepts et une identification de modèles mentaux transversaux ne semble pas avoir été mené ces dernières années pour relancer l'intérêt de ces études. Il serait intéressant réaliser cet effort, afin de permettre une plus grande homogénéité dans les études sur les modèles mentaux et relancer l'intérêt de monde scientifique sur le sujet. Cette problématique est, comme la précédente, très large, et ne semble pas solvable par une seule étude.

2.4.4 Conclusion

Au final, il semble difficile de résoudre ces problèmes de manière simple. Le problème d'essoufflement ne semble pas solvable si les études des modèles mentaux restent liées à des domaines trop spécifiques et ne peuvent être reproduites facilement dans des domaines similaires. Le problème de spécialisation des domaines d'étude est compliqué, car chaque spécificité peut avoir une influence sur le résultat final et des études sur un domaine générique pourraient se révéler trop abstraites pour espérer des résultats probants. Par contre, le problème d'hétérogénéité du public-cible et de définition de termes, comme novice et expert, pourrait être résolu en proposant des règles de profilage communes à tout type d'étude. Ces règles se basant sur des critères standards de niveau de connaissances et de pratique. Ce profilage pourrait donner un cadre utilisable aux études sur des domaines plus génériques et donner une réponse partielle au problème de spécialisation des domaines d'étude.

3 Méthodologie de recherche

L'état de l'art met en avant plusieurs pistes à creuser, notamment le besoin de caractériser plus formellement les participants et le besoin d'aborder, de manière plus générique, les différences d'interprétation de certains modèles mentaux. Ces enrichissements pourraient aider à mieux cerner les différences existant entre novices et experts dans l'application de certains modèles mentaux, et pourraient fournir des solutions pour effacer ces différences.

3.1 Problématique et question de recherche

3.1.1 Problématique

Dans le monde l'entreprise, il n'est pas rare, pour les nouveaux employés, généralement des jeunes diplômés, de devoir suivre une formation en interne de plusieurs semaines à leur entrée en fonction. Ce constat est d'autant plus vrai dans le domaine de l'informatique. En effet, en plus de la découverte des mécanismes régissant le monde de l'entreprise et du domaine dans lequel le nouvel arrivant s'engage, il est nécessaire de prendre en compte l'hétérogénéité et la constante évolution du monde informatique dans son ensemble. Dans une moindre mesure, un besoin similaire de formation est constaté lorsqu'un employé expérimenté doit changer de domaine ou simplement d'équipe : un temps d'adaptation est nécessaire pour intégrer le nouveau contexte dans lequel il va évoluer et également pour s'intégrer à l'équipe en place.

Outre les aspects sociaux, contextuels et techniques liés à ce besoin d'adaptation, l'un des buts de ces phases de formation est de garantir un certain niveau d'homogénéité dans les réalisations des produits de l'entreprise. Ces phases de formation ont un coût non négligeable, financier pour l'entreprise, mais également un impact sur le nouvel arrivant ou l'employé en transition, qui doit se remettre en question et faire l'effort de s'adapter à des mécanismes auxquels il n'est pas forcément habitué.

Un des objectifs des entreprises est d'arriver à diminuer ces phases de formation au maximum, afin de réduire les coûts et la charge mentale de ses employés liée à ces transitions. Pour ce faire, il est nécessaire d'identifier les manquements que les employés pourraient avoir concernant les différents mécanismes utilisés au sein de l'entreprise, que ce soit d'un point de vue conceptualisation technique, communication ou méthodologique. À la base de ces thématiques se trouvent généralement un ou plusieurs modèles mentaux qui aident les employés à intégrer plus ou moins facilement les concepts et mécanismes plus ou moins complexes. Dans ces conditions, l'hypothèse selon laquelle ces man-

quements constatés seraient liés aux modèles mentaux pourrait amener les entreprises à concentrer leurs efforts de formation dans l'apprentissage et l'application correcte de ces modèles mentaux, quel que soit le profil de l'employé.

L'identification des modèles mentaux est propre à chaque entreprise, à chaque projet, voire à chaque équipe. Ces modèles sont liés aux paradigmes et langages de programmation, aux méthodologies utilisées, ou encore à la structure de l'entreprise et des équipes. Il est donc difficile d'identifier ces modèles mentaux de manière générique. Par contre, les modèles mentaux liés à la manière qu'ont les nouveaux arrivants ou employés en transition d'appréhender un environnement ou un problème sont applicables dans la majorité des cas et peuvent être étudiés pour déceler d'éventuels manquements chez certaines catégories d'individus.

3.1.2 Question de recherche

La problématique soulevée est complexe, c'est pourquoi cette étude va se concentrer sur un nombre restreint de modèles mentaux qui sont : (1) "Text-based" ; (2) "Situation" ; (3) "Program" ; (4) "Top-down" ; (5) "Bottom-up" et (6) Modèles mentaux partagés. Les cinq premiers modèles mentaux ont la particularité d'être transversaux, à savoir non attachés à un domaine particulier et s'adaptant à la plupart de situations rencontrées lors de développement de solutions. Ils sont également étroitement liés à l'approche individuelle de chaque participant dans différentes situations. Au contraire, le sixième modèle mental concerne l'approche groupée, en équipe, de différentes situations.

Partant de ces modèles mentaux, des éléments de réponse vont être apportés à la question : **Quels manquements peuvent exister dans l'utilisation des modèles mentaux par les développeurs en fonction de leur niveau d'expertise (novice ou expert) ?**

Deux sous-questions vont être posées :

(1) Quelles sont les caractéristiques qui vont aider à définir le niveau d'expertise d'un développeur : novice ou expert ?

(2) Les manquements dans l'utilisation des modèles mentaux sont-ils liés au niveau d'expertise d'un développeur ?

Pour collecter des données, un questionnaire a été conçu et proposé auprès de professionnels du domaine de l'informatique, issus de secteurs d'activité variés (académique, public, privé, bancaire, industriel ...). Dans un premier temps, le niveau d'expertise sera défini à partir des données personnelles des répondants au questionnaire. Ensuite, les répondants ont dû s'exprimer sur quatre aspects particuliers qui sont : (1) L'approche globale d'un nouveau projet ; (2) La communication ; (3) Les concepts techniques et (4) La méthodologie de travail. Des incohérences d'utilisation de modèles mentaux sont alors identifiées dans les réponses obtenues au questionnaire et, à travers ces incohérences, des manquements possibles chez les développeurs. Les incohérences sont finalement confrontées aux niveaux d'expertise précédemment définis.

3.2 Public-cible et contexte

La première étape de ce mémoire a été de réaliser un état de l'art traitant des modèles mentaux utilisés dans le processus de compréhension de programme par les novices et les experts. Il est notamment apparu difficile de définir un panel de participants cohérent pouvant être commun à toutes les études.

Les termes novices et experts pouvant être appliqués à différents groupes de personnes suivant le contexte, le public-cible de cette étude se doit d'être assez large pour représenter un maximum de ces groupes, mais ne doit pas prendre en compte un trop grand nombre de profils différents pour limiter le domaine de recherche. L'éventail de groupes compris dans l'état de l'art commence par des étudiants en début de cycle d'études, ou des étudiants dans un autre domaine que l'informatique, continue avec des groupes de personnes avec des années d'expertise dans un secteur d'activité, et se termine avec des spécialistes de domaines niches et des profils gravitant autour de la programmation, comme des chefs de projet. Il est donc indispensable de réduire le panel par rapport au niveau de l'expertise des participants, mais également par rapport aux profils étudiés.

Cette étude se concentrant principalement sur les modèles mentaux aidant à la réalisation de projets en entreprise, il n'a pas été jugé pertinent de prendre en considération les étudiants en début ou en cours de cycle (ces étudiants n'ayant généralement pas encore appréhendé la plupart des modèles mentaux traités par l'étude) et les personnes ne connaissant pas les arcanes de la programmation. De même, les profils spécialisés, mais ne gravitant pas autour de la programmation, tels que les chefs de projet, ne font pas partie du public-cible. Une exception est toutefois acceptée quand une personne exerçant un de ces profils a une expertise passée dans la programmation, pour peu que cette expertise soit jugée solide. Le public-cible se compose donc de profils techniques, tels que des développeurs et analystes développeurs, mais également des architectes, et aussi des étudiants en fin de cycle informatique.

Le questionnaire de cette étude a donc été proposé à plusieurs entreprises belges actives dans le secteur du développement de solutions informatiques. Directement contactés : (1) Une entreprise évoluant dans le service public ; (2) Une entreprise offrant des services de consultance dans les domaines privés et publics et (3) Une entreprise évoluant dans le secteur de l'énergie. Elles ont accepté de diffuser massivement l'invitation au sondage à leurs employés. Cette invitation étant libre, les employés n'avaient aucune obligation de donner suite à cette enquête. En parallèle, le questionnaire a été diffusé auprès (4) Des étudiants en Master et BAC de l'UNamur en fin ce cycle. Enfin, une invitation à diffuser le sondage auprès des connaissances des sondés accompagnait l'invitation initiale, afin de récolter le plus de réponses possible, auprès de participants à l'expertise la plus hétérogène possible.

Dans les études comparatives basées sur l'expertise des participants, l'un des aspects les plus compliqués est donc la catégorisation des participants selon leur niveau d'expertise. Ce constat est d'autant plus vrai lorsque le panel de participants est très large et/ou que les sujets abordés sont hétérogènes. Les paramètres à prendre en compte sont nombreux et faire une corrélation entre eux n'est pas toujours évident. Les exemples sont nombreux dans la littérature. Parfois, pour un même panel, certains participants sont catégorisés comme novices dans le cadre d'une étude, alors qu'ils présentent une expertise soutenue dans un domaine considéré comme non représentatif par rapport au

sujet-cible de l'étude. Parfois, ces mêmes participants sont considérés comme experts dans le cadre d'une autre étude, car leur expertise entre dans les critères de catégorisation. Ces critères, liés au contexte dans lequel les études sont réalisées, sont donc essentiels. Dans une étude plus généraliste, dans le sens où elle ne se limite pas à un domaine d'expertise ou à un secteur d'activité en particulier, comme celle-ci, le contexte n'est pas défini : il n'est pas possible de se baser sur des indicateurs relatifs au *core business* d'une entreprise ou au secteur d'activité dans lequel les participants évoluent. Pour pallier ce problème, les réponses à la première partie du questionnaire (l'identification du participant) vont être utilisées pour déduire des profils de participants, qui seront ensuite classés dans la catégorie novice ou la catégorie expert. Sur cette base, il sera possible d'analyser les résultats obtenus dans les autres parties du questionnaire et, éventuellement, de détecter des manquements dans les modèles mentaux.

3.3 Collecte des données

Dans le cadre de ce travail, l'application de sondage Drag'n Survey [1] a été imposée par l'UNamur. Cette application permet de réaliser des enquêtes diffusables facilement (au moyen d'une URL) et respectant les règles RGPD en vigueur [2]. L'enquête, un questionnaire divisé en 5 sections, est restée accessible au public-cible pendant une période déterminée de 8 semaines.

Les cinq sections sont : (1) Identification du sondé ; (2) Une approche globale d'un nouveau projet ; (3) Communication inter-équipe, intra-équipe et avec des équipes externes à l'entreprise ; (4) Concepts techniques de réalisation d'une solution et (5) Méthodologie. Chaque section présente un nombre limité de questions, pour s'assurer que les répondants aillent au bout du processus. L'ordre des sections, ainsi que des questions, est prédéfini et fixe. Le questionnaire complet peut être rempli en 30 minutes.

Les réponses aux questions ont été collectées et structurées automatiquement par l'application. Les catégories de questions à choix, comme les questions à choix multiples ou à notation, et les questions à niveaux d'appréciation, font l'objet d'analyses statistiques descriptives permettant une visualisation des résultats en temps réel. L'application permet également, à la fin de la période d'accès d'un questionnaire, de générer des rapports précis sur les résultats, par question ou par sondé.

À la fin de la période d'accessibilité du questionnaire, les réponses aux différentes questions ont été analysées au regard des définitions proposées pour les novices et experts dans le cadre de cette étude. Les résultats permettront de déterminer s'il existe des différences d'utilisation des modèles mentaux entre ces deux catégories et si ces différences mettent en évidence des manquements imputables à leur niveau d'expertise.

La première section se focalise sur l'identité des sondés, en utilisant principalement des questions à choix multiples et des listes déroulantes. Les autres sections sont principalement composées de questions à échelles de Likert [Boone et Boone, 2012 ; Kaptein et al., 2010 ; Sullivan et Artino Jr, 2013] à 5 échelons et de questions à réponses libres permettant aux sondés de dé-

1. https://www.dragnsurvey.com/
2. Article 6.1 (b) du Règlement 2016/679 du Parlement européen et du Conseil du 27 avril 2016

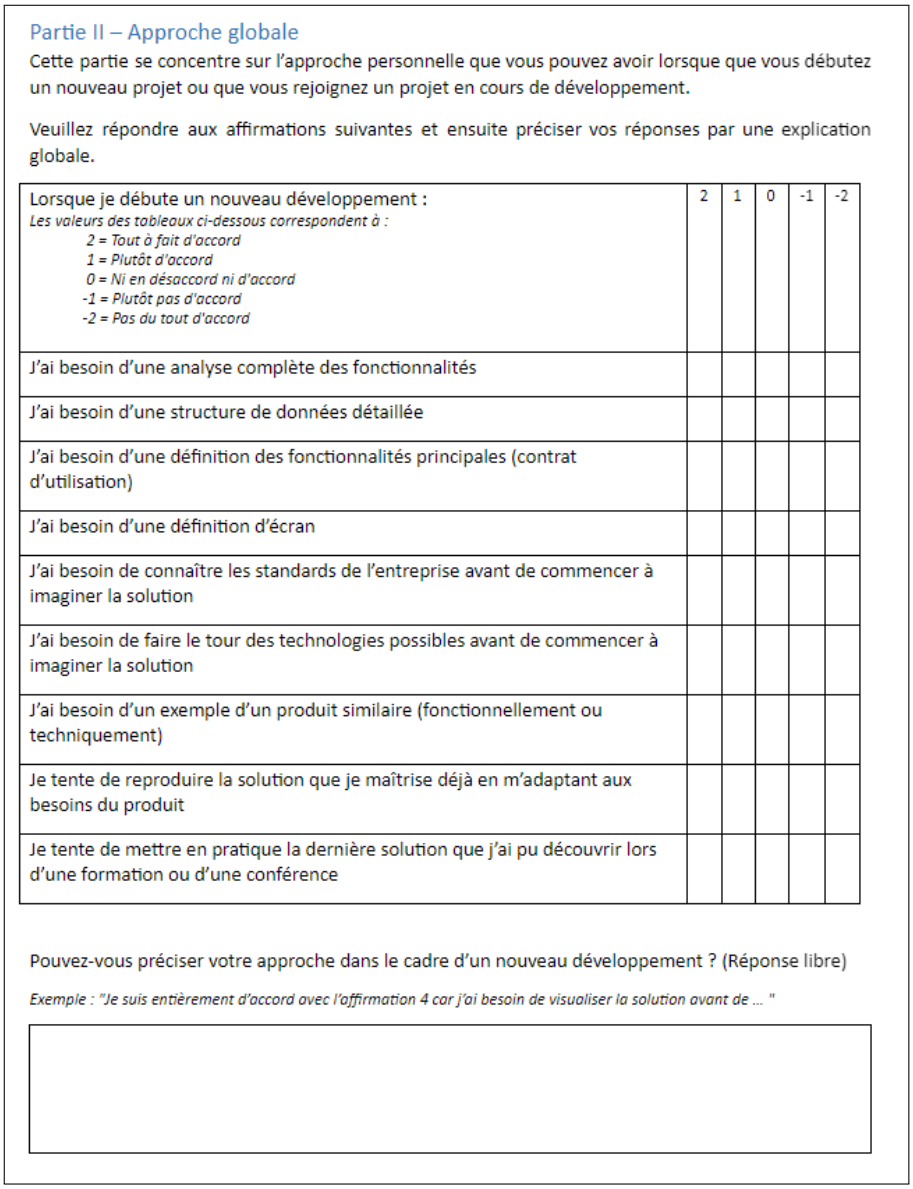
tailler, au besoin, leurs réponses. Le choix des échelles de Likert est motivé par le caractère hétérogène du public-cible et la volonté de généraliser au mieux les sujets abordés. Il s'agit de survoler les sujets en mettant en évidence leurs buts et aspects principaux et de permettre aux sondés de développer leur point de vue dans les questions à réponse libre. L'analyse des réponses libres permet d'identifier d'éventuelles méprises du concept abordé ou, au contraire, une grande connaissance, ou encore une application particulière qui pourrait amener à l'identification de modèles mentaux erronés ou hybrides.

Cette première section permet d'identifier les différents types de sondés sur base de plusieurs critères tels que l'expérience dans leur domaine de prédilection, les années d'expérience tous domaines confondus ou encore leur niveau d'étude. Ces questions vont permettre de catégoriser de manière plus granulaire les sondés et donner la possibilité d'évaluer les réponses aux différents sujets abordés dans le questionnaire sur base de ces catégories. De cette manière, cette étude, basée sur les modèles mentaux plus transversaux, pourra prendre en considération un panel d'experts plus large, car elle pourra tenir compte des sondés expérimentés dans des domaines proches du domaine visé par certaines affirmations du questionnaire. Le traitement des résultats basé sur cette catégorisation, permettra de déterminer si des manquements sont plus présents dans l'une ou l'autre catégorie, si ces manquements sont imputables à une méconnaissance de la part de novice ou à une distorsion du modèle dû à une expérience tronquée.

Les quatre sections suivantes abordent quatre sujets différents, mais suivent un même modèle : une ou plusieurs sous-sections, proposant chacune une situation avec plusieurs affirmations soumises à échelle de Likert, et une question libre permettant de préciser les choix effectués. L'exemple 3.1 illustre la première sous-section consacrée à l'approche globale (section 2).

Approche globale : Ce premier sujet abordé par le questionnaire se divise en deux sous-sections : (1) L'approche prise lorsque le sondé entame un nouveau projet et (2) L'approche prise par le sondé lorsqu'il est incorporé à un projet en cours de réalisation. Ces deux approches, comportant respectivement 9 et 10 affirmations Likert, ont pour objectif d'identifier les modèles mentaux les plus utilisés par ces derniers dans ces différentes situations. Dans les affirmations Likert, il n'est pas directement fait mention des modèles mentaux abordés, mais uniquement des principes qui amènent le sondé à mettre en oeuvre ces modèles mentaux. Par exemple, la première affirmation Likert "Lorsque je débute un nouveau développement, j'ai besoin d'une analyse complète des fonctionnalités" fait référence aux modèles mentaux "Text-based" et "Top-down" définis dans la section 2.3.1 Modèles mentaux de compréhension de programme page 16 et repris par de nombreuses études, comme évoqué dans l'état de l'art précédemment. Avec ce premier sujet, les modèles mentaux "Top-down", "Bottom-up", "Text-based", "Situation" et "Program" sont au centre des évaluations. Les tableaux 3.1 et 3.2 reprennent la liste des affirmations, ainsi que les modèles mentaux qui leur sont associés.

Communication : Ce second sujet se divise en trois sous-sections reprenant les différentes communications que peut avoir un sondé avec le monde qui l'entoure : (1) La communication au sein de son équipe de développement ; (2) La communication avec les autres équipes de son entreprise et (3) La communication avec le monde extérieur, tel que les représentants des clients hors entreprise ou les équipes d'applications externes à l'entreprise. Ces trois



Partie II – Approche globale

Cette partie se concentre sur l'approche personnelle que vous pouvez avoir lorsque que vous débutez un nouveau projet ou que vous rejoignez un projet en cours de développement.

Veuillez répondre aux affirmations suivantes et ensuite préciser vos réponses par une explication globale.

Lorsque je débute un nouveau développement : *Les valeurs des tableaux ci-dessous correspondent à :* *2 = Tout à fait d'accord* *1 = Plutôt d'accord* *0 = Ni en désaccord ni d'accord* *-1 = Plutôt pas d'accord* *-2 = Pas du tout d'accord*	2	1	0	-1	-2
J'ai besoin d'une analyse complète des fonctionnalités					
J'ai besoin d'une structure de données détaillée					
J'ai besoin d'une définition des fonctionnalités principales (contrat d'utilisation)					
J'ai besoin d'une définition d'écran					
J'ai besoin de connaître les standards de l'entreprise avant de commencer à imaginer la solution					
J'ai besoin de faire le tour des technologies possibles avant de commencer à imaginer la solution					
J'ai besoin d'un exemple d'un produit similaire (fonctionnellement ou techniquement)					
Je tente de reproduire la solution que je maîtrise déjà en m'adaptant aux besoins du produit					
Je tente de mettre en pratique la dernière solution que j'ai pu découvrir lors d'une formation ou d'une conférence					

Pouvez-vous préciser votre approche dans le cadre d'un nouveau développement ? (Réponse libre)

Exemple : "Je suis entièrement d'accord avec l'affirmation 4 car j'ai besoin de visualiser la solution avant de ... "

FIGURE 3.1 – Exemple : Première question "Approche globale"

types de communication ont respectivement 7, 6 et 6 affirmations évaluées par échelle de Likert, construites de la même manière qu'à la section précédente. La communication est un sujet important dans la définition des modèles mentaux considérés dans l'approche globale, mais également dans les autres modèles mentaux évoqués dans l'état de l'art : les modèles mentaux partagés ("Shared Mental Model"). Ces derniers sont nombreux et dépendent fortement de l'environnement dans lequel ils se développent. Ils seront aussi considérés dans l'analyse des sujets suivants, mais ils prennent leurs bases principalement dans la communication et la manière qu'ont les sondés de se représenter cette communication à travers les différentes sous-sections. La communication permet de répondre à la question "Comment sont construits les modèles mentaux partagés" ? Quant aux sujets suivants, "Technique" et "Méthodologie", ils permettent de répondre respectivement aux questions : *"De quoi sont composés*

les modèles mentaux partagés ?" et *"Pourquoi ces modèles mentaux partagés sont construits de cette manière ?"*. Les affirmations Likert se penchent sur l'approche des participants par rapport à la communication verbale et écrite, ainsi que sur la résolution de problème individuel et en groupe. Comme pour les modèles mentaux, il n'est pas toujours fait directement mention du sujet abordé par l'affirmation, mais une allusion indirecte est utilisée. Par exemple, le terme documentation est utilisé pour faire référence à la documentation écrite.

	Text-based	**Situation**	**Program**	**Top-down**	**Bottom-up**
1. J'ai besoin d'un analyse complète des fonctionnalités	X			X	
2. J'ai besoin d'une structure de données détaillée	X				X
3. J'ai besoin d'une définition des fonctionnalités principales (contrat d'utilisation)	X	X			
4. J'ai besoin d'une définition d'écrans		X		X	
5. J'ai besoin de connaître les standards de l'entreprise avant de commencer à imaginer la solution		X		X	
6. J'ai besoin de faire le tour des technologies possibles avant de commencer à imaginer la solution		X			X
7. J'ai besoin d'un exemple d'un produit similaire (fonctionnellement ou techniquement)			X		
8. Je tente de reproduire une solution que je maîtrise déjà en m'adaptant aux besoins du produit		X	X		
9. Je tente de mettre en pratique la dernière solution que j'ai pu découvrir lors d'une formation ou d'une conférence			X		X

TABLE 3.1 – Approche Gloable : Modèles mentaux par affirmations (Nouveaux projets)

Technique : Ce troisième sujet se démarque des sujets précédents, dans le sens où il ne se base pas réellement sur les modèles mentaux précédemment cités. Les 13 affirmations présentées aux sondés sont des affirmations qui mettent en avant des préjugés ou des conduites qui sont considérés par certains comme de bonnes pratiques, mais qui doivent être nuancés suivant le contexte dans lequel ils sont appliqués. Ces préjugés et nuances dans l'application des bonnes pratiques renforcent l'établissement de modèles mentaux partagés et pourraient mettre en évidence la difficulté d'identifier des éléments lacunaires dans certains contextes quand, dans d'autres contextes, ces éléments sont jugés non pertinents. L'affirmation du questionnaire suivant : *Un outil "Black Box" est à proscrire, rien ne doit limiter la visibilité d'une solution*, est un exemple qui illustre bien le propos. L'utilisation de librairies ou de frameworks, externes ou internes à l'entreprise, aide à la réalisation de produits, facilite les développements et augmente la productivité, mais leur simple utilisation entraine des comportements et des réflexes qui peuvent sembler erronés dans d'autres circonstances. Ce genre de pratique cache, sous plusieurs couches d'abstraction, certains concepts qui ne sont pas forcément compris et maitrisés par les utilisateurs de ces librairies et frameworks. Quand une ligne de code permet de configurer tout un pan d'une application et définir un enchainement d'étapes,

	Text-based	Situation	Program	Top-down	Bottom-up
1. J'ai besoin d'un analyse complète des fonctionnalités	X			X	
2. J'ai besoin d'une structure de données détaillée	X				X
3. J'ai besoin d'une définition des fonctionnalités principales (contrat d'utilisation)	X	X			
4. J'ai besoin d'une définition d'écrans		X		X	
5. J'ai besoin de connaître les standards de l'entreprise avant de commencer à imaginer la solution		X		X	
6. J'ai besoin d'analyser en détail les développements déjà en place pour me familiariser avec l'existant		X			X
7. J'ai besoin de démarrer sur des développements de type correction pour me familiariser avec l'existant			X		X
8. J'ai besoin de tester l'application existante (tests unitaires ou fonctionnels)		X		X	X
9. Je tente de reproduire une solution que je maîtrise déjà en m'adaptant aux besoins du produit		X		X	
10. Je tente de mettre en pratique la dernière solution que j'ai pu découvrir lors d'une formation ou d'une conférence			X		X

TABLE 3.2 – Approche Gloable : Modèles mentaux par affirmations (Intégration projet existant)

ce qui, si ces librairies ou frameworks n'étaient pas utilisés, demanderait au programmeur une plus grande compréhension des mécanismes sous-jacents, il est probable que les utilisateurs de ces librairies présenteraient des lacunes et des incohérences dans la compréhension de certains modèles mentaux. Les affirmations relatives à ce sujet se penchent sur trois capacités qui se retrouvent chez les programmeurs et qui, suivant le niveau d'adhésion de ces derniers, déterminent des comportements de programmation différents qui influencent la perception de certains modèles mentaux. Quand certaines affirmations font référence à la capacité de certains développeurs à innover dans les solutions qu'ils proposent, d'autres affirmations se penchent sur la capacité de certains à la réutilisation de solutions. La troisième série d'affirmations s'oriente sur l'importance que le programmeur donne à la qualité du code fourni. Ces trois capacités peuvent aider à la catégorisation des certains types de programmeurs et à déterminer leur manière d'aborder certains modèles mentaux, ainsi que le niveau d'acceptation et de compréhension de ces derniers.

Méthodologie : Ce dernier sujet présente 8 affirmations. Les méthodologies de conception et de gestion sont nombreuses et peuvent parfois être complémentaires, comme totalement incompatibles. La pratique de l'une ou l'autre de ces méthodologies peut grandement influencer la manière dont les modèles mentaux sont construits et utilisés. Certaines méthodologies favorisent les modèles mentaux du type "Top-Down" comme "Bottom-up" ou "Situation" et "Program". Bien que le Domain Driven Design (DDD) [3] ne soit pas consi-

3. https://domaindrivendesign.org/

déré comme une méthodologie en soi, cette approche est considérée dans cette étude, car elle favorise l'utilisation du modèle mental "Situation", tandis que le Test Driven Development (TDD) [4], également considéré dans les affirmations de cette section, favorise le modèle mental "Program". Les principes Agile, tels que l'Extreme programming [5] ou Scrum [6], sont aussi suggérés dans les différentes affirmations, ces deux dernières pratiques étant respectivement orientées "Buttom-up" et "Top-down". Suivant les contextes, l'utilisation d'une ou plusieurs méthodologies va également influencer le modèle mental partagé ("Shared Mental Model") au sein de l'équipe et va potentiellement créer des différences de perception entre les sondés. Ces différences ne seront pas forcément considérées comme des lacunes suivant certains points de vue, mais peuvent mettre en évidence certaines pratiques "à risque".

L'étude approfondie des réponses libres ne sera pas abordée dans ce document. Une première lecture de ces réponses montre que les participants n'ont, pour la plupart, pas argumenté leurs réponses. Le nombre restreint de réponses pertinentes, ainsi que la régularité relative des répondants ayant pris la peine de remplir les réponses libres, rendent l'étude de ces éléments compliquée à interpréter et risque de fausser les résultats en prenant en compte l'avis de seulement une poignée de participants. En prenant comme exemple le dernier sujet du questionnaire "Méthodologie", sur les 32 sondés ayant fourni une réponse libre, seuls 12 sondés précisent leurs choix de manière exploitable. Seulement, sur ces 12 sondés, seuls 5 d'entre eux ont pris le pli de justifier concrètement leurs choix dans les réponses libres des autres sujets. Il est donc hasardeux de tenter de trouver des réponses probantes sur base d'un sous-échantillon ne représentant que 15,63 % des participants finaux (n=32). Toutefois, les réponses libres seront utilisées de manière sporadique pour illustrer certains résultats ou appuyer une piste de réflexion.

4. https://www.agilest.org/devops/test-driven-development/
5. http://www.extremeprogramming.org/
6. https://www.scrum.org/

4 Résultats

4.1 Description des participants

4.1.1 Recensement

Afin de permettre une catégorisation des différents profils des participants à l'étude, plusieurs critères ont été définis et présentés dans la première section du questionnaire. Ces critères sont : (1) Le sexe ; (2) La tranche d'âge ; (3) Le niveau d'étude ; (4) La fonction actuelle principale ; (5) Le nombre d'années d'expérience dans cette fonction principale ; (6) Le nombre d'années d'expérience dans le domaine informatique ; (7) Les paradigmes de programmation connus et (8) Les langages de programmation privilégiés par le participant.

51 réponses ont été obtenues via le questionnaire : six femmes (11,76 %) et 45 hommes (88,24 %). Plus d'un tiers des sondés se trouve dans la tranche d'âge 25-35 ans (18 sondés, soit 35,29 %) et plus d'un quart dans la tranche 35-45 ans (13 sondés, soit 25,49 %). La tranche des 18-25 ans (neuf sondés, soit 17,65 %) est la troisième la plus représentée. Enfin, les plus de 45 ans représentent un cinquième des sondés, avec 13,73 % des 45-55 ans (sept sondés) et 7,84 % des plus de 55 ans (quatre sondés).

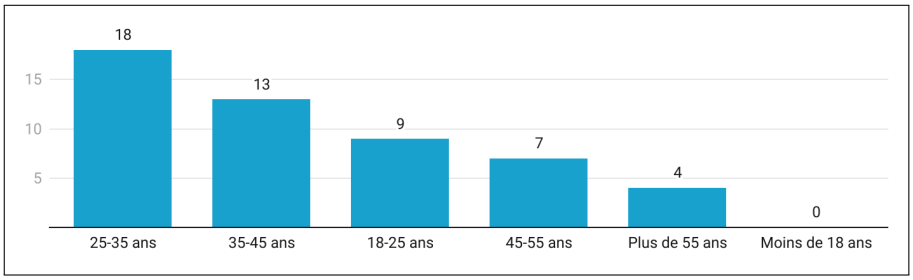


FIGURE 4.1 – Catégories d'âge des sondés au moment de l'étude

La majorité des sondés (soit 46, n = 51) ont un **niveau d'étude** au moins équivalent au type court ou baccalauréat. 30 sondés (soit 58,83 %) sont titulaires d'un diplôme d'études supérieures de type court, tandis que 16 sondés (soit 31,37 %) ont au minimum un master. Quatre sondés (soit 7,84 %) ont un diplôme de secondaire supérieur et un sondé ne possède pas de diplôme.

Outre le niveau d'étude, la **fonction principale** des sondés peut être un indicateur pertinent et entre, de ce fait, dans le processus de catégorisation des

participants. La majorité des sondés (soit 32, 62,75 %) ont une fonction de développeur : un tiers (17 sondés, 33,33 %) se considèrent comme développeurs, tandis que 15 sondés (soit 29,41 %) se considèrent comme analystes développeurs. Ensuite viennent les architectes, au nombre de neuf (soit 17,65 %), et les étudiants/stagiaires, au nombre de six (soit 11,76 %). Les quatre derniers participants (soit 7,84 %) se répartissent dans des rôles tels que le testing, l'intégration ou la gestion opérationnelle.

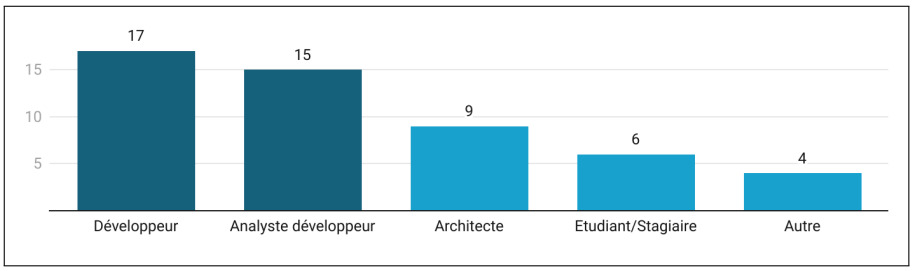


FIGURE 4.2 – Fonctions principales occupées par les sondés au moment de l'étude

Le **niveau d'expérience** du sondé **dans la fonction en cours** est également considéré comme un indicateur intéressant. Le panel de sondés (n = 51) semble ici assez homogène. Six sondés (soit 11,76 %) n'ont aucune expérience : ce sont des étudiants en fin de cycle ou en stage d'apprentissage. Pour les autres, neuf sondés (soit 17,65 %) ont entre 1 et 3 ans d'expérience, huit (soit 15,69 %) entre 3 et 5 ans, dix (soit 19,61 %) entre 5 et 10 ans, 14 (soit 27,45 %) entre 10 et 20 ans et quatre (soit 7,84 %) plus de 20 ans. Dans le milieu professionnel, les personnes faisant partie des deux premières catégories (aucune expérience et moins de 3 ans d'expérience) sont généralement considérées comme des novices dans le domaine. Les personnes faisant partie des deux catégories suivantes sont considérées comme des seniors (les 3-5 ans sont parfois qualifiés de medior). Les personnes faisant partie des deux dernières catégories sont considérées comme des experts du domaine. Il y a donc un écart de 7 à 10 ans d'expérience entre le niveau novice et le niveau expert, si l'on suit les standards du milieu professionnel. Il est donc difficile de catégoriser les personnes dites seniors. De plus, le niveau d'expérience dans la fonction occupée au moment de l'étude n'est pas forcément représentatif du niveau réel des sondés.

L'indicateur suivant, le **nombre d'années d'expérience dans le domaine informatique**, permet (1) de se rendre compte de l'expertise réelle du sondé, tous domaines d'activité confondus, et (2) par une recherche croisée avec l'indicateur précédant, de se rendre compte que la définition de novice ne se limite pas à ce dernier indicateur. Une certaine proportion des sondés suit, ou a suivi, une évolution dans leurs activités qui peut influencer la catégorisation de leur niveau d'expertise. Sur le panel de novices considérés selon le précédent indicateur (soit 15, 29,41 %), quatre (soit 26,67 % d'entre eux) ont un niveau d'expérience dans le domaine informatique supérieur (évolution principalement du développement à l'architecture). Une différence similaire est observée sur le panel des experts (soit 18 sondés, 35,29 %), où quatre (soit 22,22 % d'entre eux) ont une carrière dans l'informatique qui dépasse leur niveau expertise dans leur domaine actuel. Généralement, il est question ici d'un changement de paradigme de programmation ou, comme les novices, d'une évolution vers l'architecture. Quant au panel qualifié de senior (soit

18 sondés, 35,30 %), neuf (soit 50 % d'entre eux) sont en cours d'évolution vers un domaine d'expertise différent : soit un changement complet d'orientation, soit un changement de langage de programmation, voire de paradigme de programmation.

Les figures 4.3 et 4.4 montrent la différence d'interprétation qu'il peut y avoir suivant l'importance donnée aux années d'expérience d'un individu ou l'importance donnée à l'activée principale. La figure 4.4 montre que 12 participants à cette étude peuvent aisément être considérés comme novices, avec moins de 3 ans d'expériences dans le domaine informatique. Par contre, en ne s'intéressant qu'à l'activité principale des participants, la figure 4.3 montre que 15 sondés peuvent être considérés comme novices suivant le même critère, et ce même si l'un d'entre eux a plus d'une dizaine d'années d'expérience.

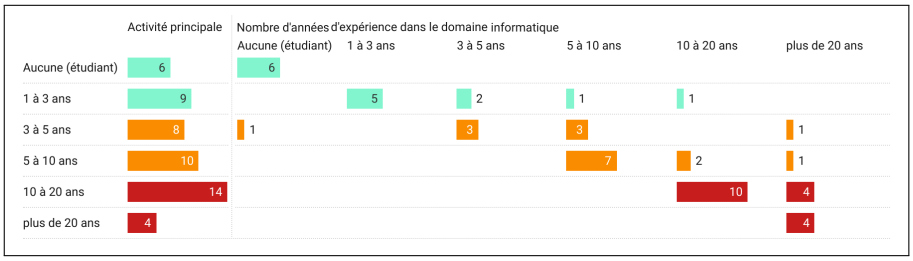


FIGURE 4.3 – Tableau croisé - Activité principale par années d'expérience (n=51)

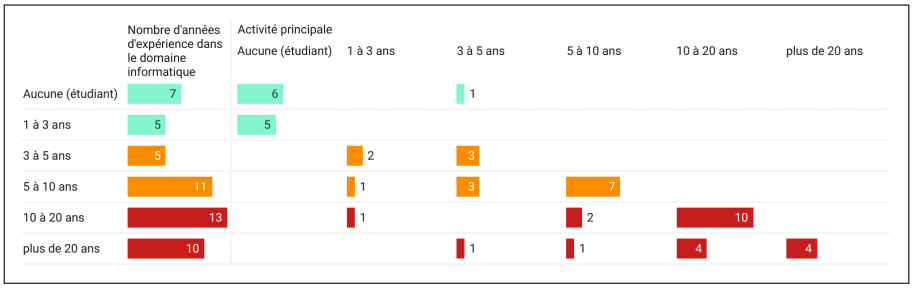


FIGURE 4.4 – Tableau croisé - Années d'expérience par années dans l'activité principale (n=51)

Les derniers indicateurs qui entrent en ligne de compte dans la tentative de définition des concepts de novice et d'expert sont **les paradigmes de programmation connus** et **les langages de programmation utilisés**. Dans cette étude, la quasi-totalité des sondés (49 sondés, soit 96,08 %) se dit familière avec le paradigme orienté-objet, avec comme langage de programmation de prédilection le Java (29 sondés, soit 59,18 %), le C++ ou C# (18 sondés, soit 36,73 %), le Python (15 sondés, soit 30,61 %) et le Typescript ou Javascipt (21 sondés, soit 42,86 %). Le second paradigme de prédilection des sondés est la programmation impérative avec 29 sondés (soit 56,86 %), avec comme langage du C, COBOL, Pascal ou encore PL/SQL. Ensuite viennent les paradigmes de programmation événementiels (19 sondés, soit 37,26 %) et déclaratifs (18 sondés, soit 35,29 %). Le paradigme de programmation concurrentiel est le moins représenté avec neuf occurrences (soit 17,65 %). Il est à noter que 23 sondés (soit 45,10 %) et dix sondés (soit 19,61 %) ont sélectionné

respectivement les paradigmes de programmation "basé web" et "visuel", ces derniers n'étant pas réellement des paradigmes de programmation.

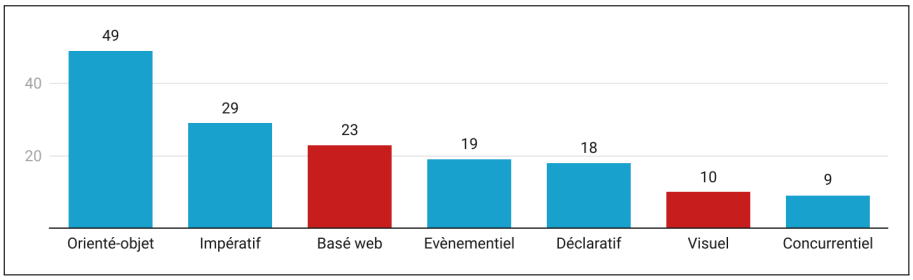


FIGURE 4.5 – Paradigmes maîtrisés par le panel (n=51)

4.1.2 Profilage

Sur base des indicateurs de recensement décrits dans la section précédente, il est possible d'identifier les critères nécessaires pour profiler les participants et les catégoriser en novices ou experts.

Au regard des résultats obtenus, un critère semble fédérer la majorité des participants : **le paradigme de programmation**. Sur les 51 participants pris en compte dans cette partie de l'étude, 49 (soit 96,08 %) déclarent connaitre le paradigme de programmation orienté-objet. Il serait simple de prendre ce critère comme point de comparaison et de se contenter de comparer les années d'expérience des participants pour en déduire leur niveau d'expertise. Malheureusement, cette méthode connait deux limites. D'abord, les participants ont, pour la plupart (42), sélectionné d'autres paradigmes de programmation que l'orienté-objet. Le questionnaire ne permet pas de déterminer dans quel paradigme le participant a le plus d'expérience. Enfin, des participants, qui ont sélectionné l'orienté-objet, ont également sélectionné les paradigmes "basé web" et "visuel" : 25 participants (sur 49, 51,02 %) ont sélectionné l'un des deux ou les deux paradigmes. Ces paradigmes n'en sont pas, comme évoqué plus haut. Il existe donc une compréhension erronée chez certains participants concernant le concept de paradigme de programmation. Ce simple constat rend caduque l'éligibilité du critère des paradigmes de programmation comme critère de profilage des participants.

Le second critère sur lequel pourrait reposer le profilage est l'activité principale exercée par le participant au moment de l'étude. 32 répondants (sur 51, soit 62,75 %) déclarent être développeurs ou analystes-développeurs. Ce critère, couplé à l'information relative au nombre d'années d'expérience dans cette fonction, pourrait constituer une base solide à la définition du profil des participants. Considérer qu'un participant ayant plus de 10 ans d'expérience dans son activité principale présente un profil d'expert est un bon point de départ. À l'opposé, un participant ayant moins de 3 ans d'expérience dans son activité principale pourrait représenter un profil de novice. Malheureusement, cette considération présente déjà une lacune : le changement d'activité en cours de carrière du participant. Il n'est pas logique de considérer comme novice une personne déclarant avoir moins de 3 ans d'expérience en tant qu'analyste-développeur, alors que ce dernier a déjà une longue expérience en tant que développeur. De même, un participant considérant son activité principale comme

développeur orienté-objet et déclarant une expérience de moins de 3 ans, mais ayant une longue expérience dans la pratique de la programmation procédurale, ne peut sans doute pas être considéré comme ayant un profil novice. Enfin, il reste à identifier les profils des participants déclarant une expérience dans leur activité principale comprise entre 3 et 10 ans. Il est donc nécessaire d'introduire une seconde information pertinente pour distinguer les différents profils : le nombre d'années d'expérience dans le domaine informatique. Cette information, bien que très générique, peut aider à la catégorisation si l'hypothèse de travail suivante est émise : le nombre d'années d'expérience dans le domaine informatique du participant représente un nombre d'années effectives (activité professionnelle) et ne tient pas compte des années d'études, de formation ou de pratique de l'informatique dans le cadre d'un hobby. Il est donc possible, sur base de l'activité principale des participants, de définir ces différents profils :

— **Profil novice** : participants ayant très peu d'expertise
— **Profil intermédiaire faible** : participants ayant une petite expertise dans le domaine informatique, que ce soit moins de 5 ans dans une seule activité ou moins de 10 ans, mais avec un ou plusieurs changements d'activité au cours de cette période.
— **Profil intermédiaire fort** : participants ayant une expertise solide dans le domaine informatique, mais en cours de transition dans une nouvelle activité. Ce profil concerne généralement les participants ayant une bonne expertise dans un paradigme de programmation et en cours de transition vers un nouveau paradigme de programmation. Sont donc repris dans ce profil les participants ayant plus de 10 ans d'expertise dans le domaine informatique et moins de 3 ans dans leur activité principale, ainsi que les participants ayant entre 5 et 10 ans d'expertise dans le domaine et plus de 3 ans dans leur activité principale.
— **Profil expert** : participants ayant une expertise confirmée dans le domaine informatique, c'est-à-dire plus de 10 ans et au moins 3 ans d'expérience dans leur activité principale.

Il est à noter que cette définition de profil ne tient pas en compte la volatilité des participants en ce qui concerne leur activité principale. Un participant présentant plus de 10 ans d'expertise dans le domaine informatique peut passer ces années à changer d'activité et donc à ne pas gagner assez d'expérience dans une activité pour être réellement considéré comme expert. Ce type de participant devrait alors être considéré comme ayant un profil novice ou intermédiaire faible. Cependant, ce critère n'étant pas pris en compte dans cette étude, ces participants risquent d'être considérés comme experts.

La figure 4.6 montre la répartition des profils pour les activités de développeur et analyste-développeur, qui représentent 62,75 % des participants à cette étude. La catégorie novice regroupe les profils novices et intermédiaires faibles, tandis que les profils experts et intermédiaires forts constituent la catégorie expert.

En ce qui concerne les 37,25 % de participants ayant une activité autre, une adaptation de l'attribution des profils va permettre de catégoriser simplement ces participants. L'activité suivante la plus représentée est l'activité architecte : neuf participants (soit 17,65 %, n=51). L'hypothèse émise dans ce cas est que l'activité d'architecture ne s'improvise pas : il faut une grande expertise dans le domaine informatique, une base solide ou une assez longue expérience dans cette activité, ce qui permet à ce profil d'être considéré comme

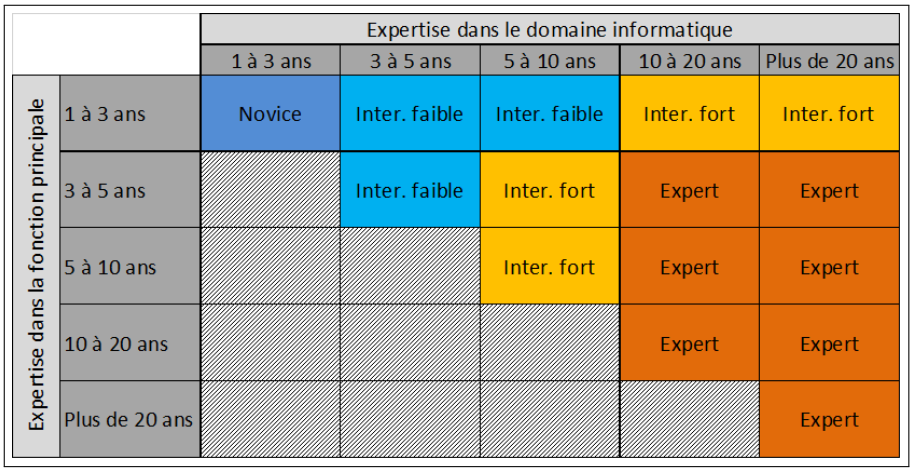


FIGURE 4.6 – Profils pour les développeurs et analystes développeurs

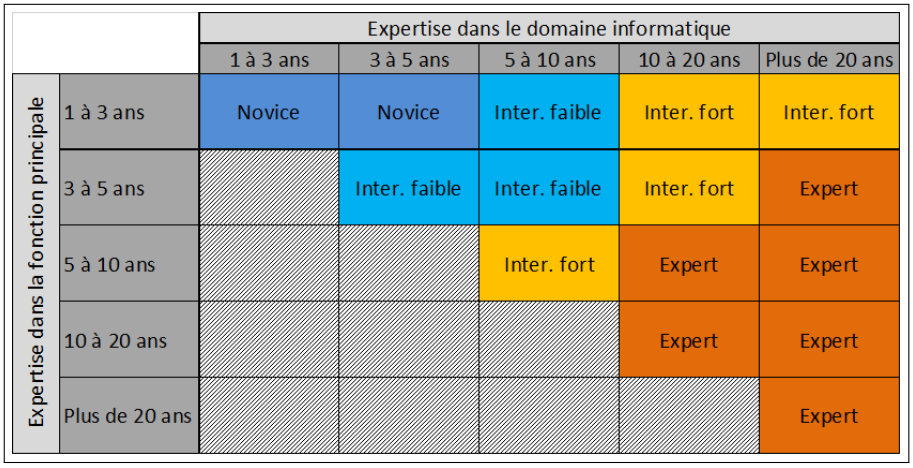


FIGURE 4.7 – Profils pour les architectes

un expert. Sur base de cette hypothèse, la figure 4.7 montre comment les profils sont attribués aux participants avec une activité principale d'architecte. La troisième activité principale la plus représentée dans cette étude est l'activité d'étudiant, six participants (soit 11,76 %). Cette activité est directement assignée à la catégorie novice, en partant de l'hypothèse que les participants suivant des formations ou des études à horaire décalé, mais présentant une expertise confirmée dans le domaine informatique, n'ont pas désigné leur activité principale comme étudiant. Ce cas pouvant néanmoins arriver, il est considéré comme négligeable et n'est pas pris en compte dans cette étude. Les autres activités représentées dans cette étude, à savoir le testing et l'intégration système, représentent quatre participants de cette étude (soit 7,84 %) et sont difficilement classables sur base de leur activité principale, qui gravite autour des sujets étudiés sans être complètement concernée par ces derniers. Le critère de définition pour ce groupe de participants sera donc le nombre d'années d'expertise dans le domaine informatique, hors activité principale, en émettant l'hypothèse que ces années d'expertise ont été réalisées dans l'une des activités précédemment citées. De plus, la seconde hypothèse émise pour

ce genre de participants est la forte probabilité d'une diminution du niveau d'expertise suite à une longue période d'activité annexe. La figure 4.8 montre les profils associés à ce genre de participants.

Expertise dans la fonction principale	Expertise dans le domaine informatique				
	1 à 3 ans	3 à 5 ans	5 à 10 ans	10 à 20 ans	Plus de 20 ans
1 à 3 ans	Novice	Novice	Novice	Inter. fort	Expert
3 à 5 ans		Novice	Inter. faible	Inter. fort	Expert
5 à 10 ans			Inter. faible	Inter. faible	Inter. fort
10 à 20 ans				Inter. faible	Inter. faible
Plus de 20 ans					Inter. faible

FIGURE 4.8 – Profils pour les autres activités

La figure 4.9 montre le profilage global des participants à l'étude. Ce profilage va permettre d'attribuer les manquements identifiés à une catégorie de participants.

Catégorie	Profil	Développeur	Architecte	Étudiant	Autre	Total profil	Total cat.
Novice	Novice	5	2	6	1	14	21
	Inter. faible	4	1	0	2	7	
Expert	Inter. Fort	9	1	0	0	10	30
	Expert	14	5	0	1	20	

FIGURE 4.9 – Classification finale

4.2 Sujets

Des réponses ont été collectées sur quatre sujets, à savoir (1) Approche globale ; (2) Communication ; (3) Technique et (4) Méthodologie. Le nombre de réponses obtenues n'est pas constant pour les quatre sujets, ni au sein d'un même sujet. Dès lors, les chiffres présentés ci-après ont été adaptés pour ne prendre en considération que les réponses provenant de sondés ayant répondu à l'entièreté des questions d'un sujet (réponses libres non comprises).

Si le niveau d'adhésion aux affirmations permet d'identifier les modèles mentaux les plus appréciés par les sondés, à savoir les plus souvent utilisés, il n'aide pas à l'identification des manquements que certains sondés possèdent dans la représentation et l'application de ces modèles. Ces possibles lacunes sont identifiées via l'interprétation des réponses libres.

4.2.1 Approche globale

Dès le premier sujet, le panel de sondés repris pour cette partie de l'étude tombe à 39.

Les figures 4.10 et 4.11 montrent de manière générale une tendance assez positive concernant les affirmations proposées dans ces sujets. Les modèles mentaux globaux généralement appliqués lors de la réalisation de projets semblent l'être par la plupart des sondés. En reportant les tendances par affirmations fournies par les sondés (voir figure 4.10 et 4.11), il est possible de déterminer l'attrait des sondés vis-à-vis des modèles mentaux liés au processus de compréhension d'un programme (définis dans le point 2.3.1 Modèles mentaux de compréhension de programme page 16). La figure 4.12 montre l'attrait moyen pour ces modèles mentaux par rapport aux différentes affirmations. Il est à noter que, malgré leur nature différente, les modèles mentaux "Text-based" et "Situation" présentent une adoption similaire par les sondés, avec respectivement une moyenne de 29,17 (74,79 %) et 26,82 (68,77 %) sondés dont l'attrait pour l'application de ces modèles mentaux semble plus marqué [1]. La même constatation peut être faite avec les modèles mentaux d'approche opposée "Top-down" et "Bottom-up", avec respectivement une moyenne de 23,88 (61,23 %) et 23,78 (60,97 %) sondés. Enfin, le modèle mental "Program" semble le moins appliqué ou maitrisé par les sondés, avec une égalité à 10,2 (26,15 %) des sondés entre un attrait neutre [2] pour l'application de ce modèle mental et une opposition [3] à son application.

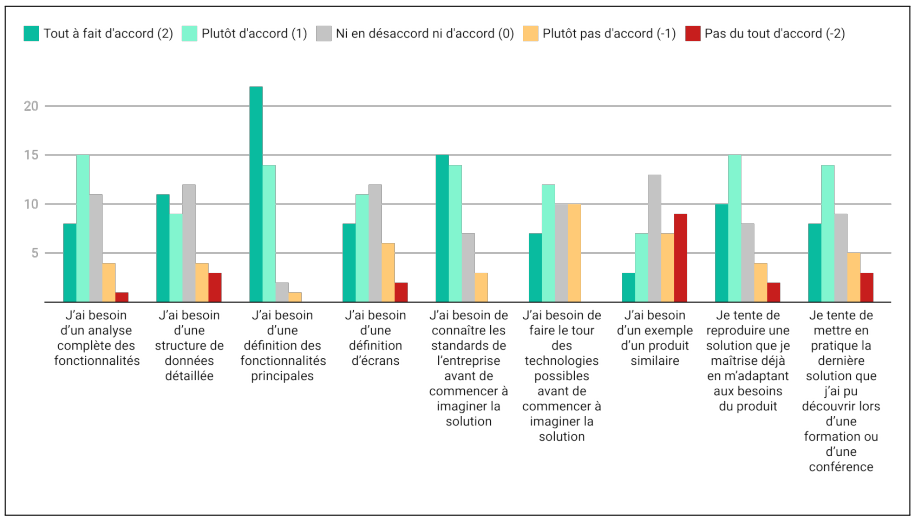


FIGURE 4.10 – Approche Globale : Tendances à l'entame d'un nouveau projet (n-39)

L'analyse des réponses libres fournies vient confirmer les constatations de la figure 4.12, tout en apportant quelques nuances. Les réponses fournies montrent une nette préférence des sondés dans l'application du modèle mental "Top-down" à l'entame d'un nouveau projet, avec des commentaires donnant

1. Un attrait est marqué quand les graduations de l'échelle de Likert "Tout à fait d'accord" et "Plutôt d'accord" ont été sélectionnées par le sondé
2. Un attrait est dit neutre quand la graduation de l'échelle de Likert "Ni en désaccord ni d'accord" a été sélectionnée par le sondé
3. Une opposition correspond aux graduations de l'échelle de Likert "Plutôt pas d'accord" et "Pas du tout d'accord"

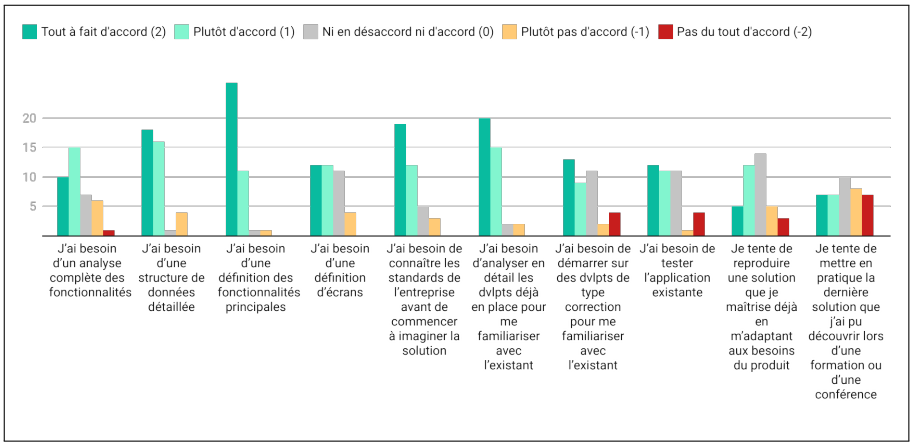


FIGURE 4.11 – Approche Globale : Tendances à l'incorporation dans un projet existant (n-39)

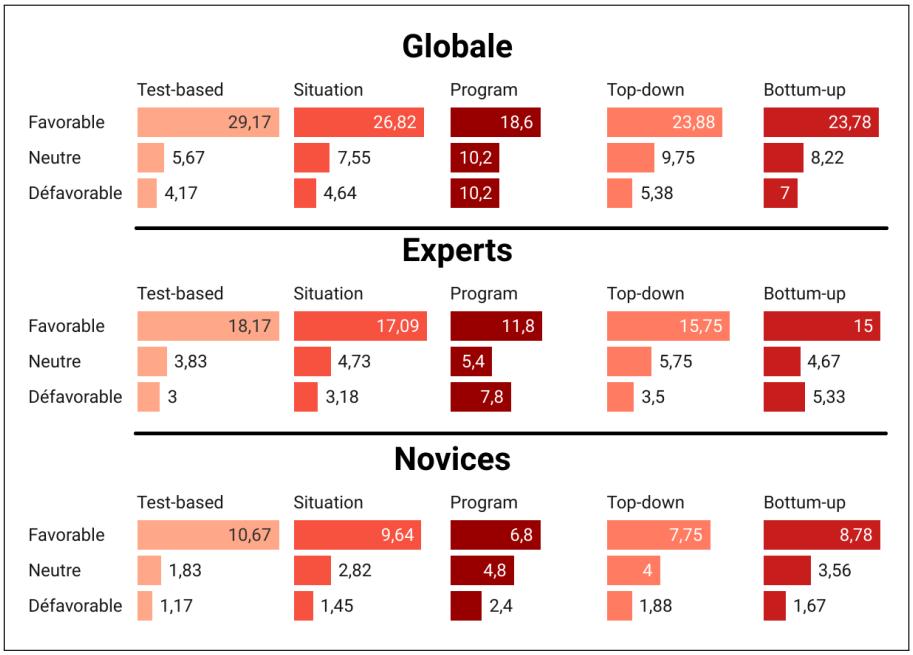


FIGURE 4.12 – Approche Globale : Moyenne d'adoption des modèles mentaux

une grande importance à l'analyse détaillée du besoin avant de démarrer la réalisation du projet. Un sondé précise : "1. Analyse du cahier des charges 2. Discussion avec les clients des besoins (fonctionnels , techniques ...) 3. Mise en place des besoins hardware et réseaux 4. Analyse des données 5. Analyse architecture logicielle 6. Modélisation des bases de données => Développement"). Par contre, lors de l'incorporation du sondé à un projet existant, la tendance des réponses libres se dirige plus vers une approche "Bottom-up" couplée au modèle "Situation". Un sondé explique : "[...] quand on bosse sur une appli existante, il est important d'avoir une visualisation complète du modèle de données afin voir ce qui existe, ce qui peut être réutilisée et comment a été pensée l'application ainsi que le domain model." Ces observations semblent

confirmer les résultats présentés dans l'état de l'art.

4.2.2 Communication

35 sondés ont répondu à l'ensemble des questions de ce sujet. Les trois sous-sections qui concernent ce sujet (la communication intraéquipe, la communication interéquipe et la communication avec le monde extérieur) comptent respectivement sept, six et six affirmations soumises à l'échelle de Likert. Chaque sous-section met en concurrence les types de communication verbale et communication écrite, ainsi que l'approche suivie par les sondés dans des situations communes, comme l'adoption de nouveaux concepts, de nouveaux outils ou encore le transfert de connaissances. La figure 4.13 montre les résultats obtenus pour la sous-section sur la communication intraéquipe, qui, d'un point de vue général, met en avant l'importance de la communication verbale, avec 82,86 % des sondés répondant favorablement à la première affirmation. La communication écrite semble également assez importante, avec 77,14 % des sondés répondant favorablement à la seconde affirmation et 65,71 % des sondés répondant défavorablement à la troisième affirmation. En ce qui concerne les 4 dernières affirmations de la figure 4.13, les avis semblent moins tranchés, mais penchent vers une élaboration de solution en groupe ou individuelle soutenue pas une communication écrite solide. Il est à noter que l'avant-dernière affirmation de la figure 4.13, qui met en avant l'individualisme du sondé en cas de résolution de problème, présente des avis moins tranchés. Cela tend à mettre en avant l'importance de l'application d'un modèle mental partagé, qui permettrait d'aider les membres d'une équipe à résoudre certains problèmes quand l'un des membres de l'équipe maitrise moins certaines parties du projet.

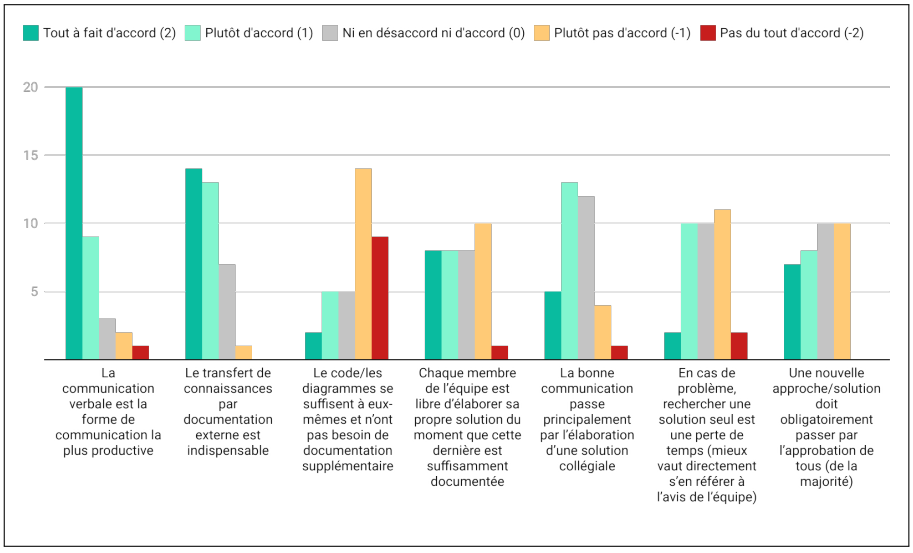


FIGURE 4.13 – Communication : Communication intraéquipe

Sur la figure 4.14, qui présente les résultats des affirmations de la sous-section sur la communication inter-équipe, la tendance observée dans la sous-section précédente, qui mettait en avant la communication verbale, semble moins prédominante. La troisième affirmation de la figure 4.14, mettant l'accent sur la communication verbale, montre toujours un attrait des sondés, avec 60 % d'entre eux répondant favorablement, mais la première affirmation

montre que seulement 8,57 % sont défavorables à une communication exclusivement écrite lorsqu'il s'agit de communiquer avec une autre équipe. Les trois dernières affirmations de la figure 4.14 se concentrent sur l'importance d'une structure inter-équipe soutenant la communication. Cette structure est, de manière générale, favorablement perçue par les sondés et met en avant le besoin d'acquisition d'un modèle mental partagé qui s'étendrait au-delà de la sphère de l'équipe.

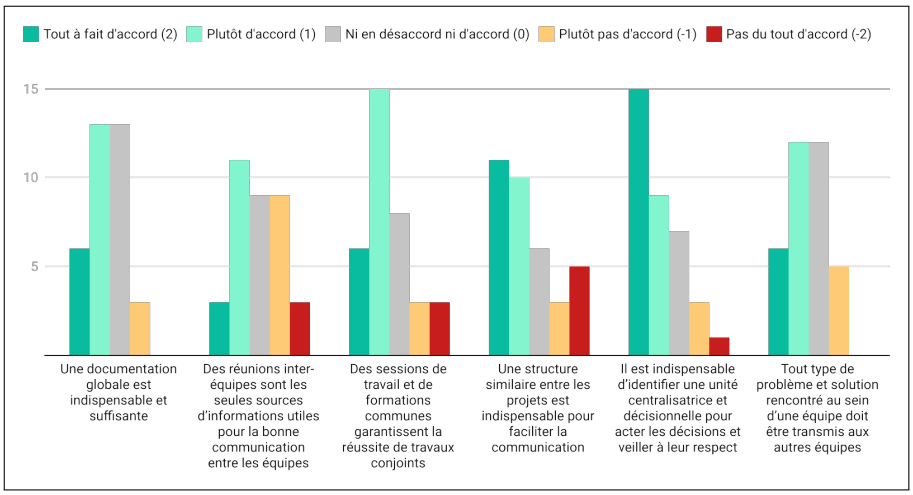


FIGURE 4.14 – Communication : Communication interéquipe

Enfin, la figure 4.15 montre les résultats de la sous-section sur la communication avec le monde extérieur, hors équipes de développement de l'entreprise. Ici, l'importance de cette communication semble être en perte de vitesse au regard des seconde et quatrième affirmations de la figure 4.15 ; les avis favorables y sont plus fortement contrebalancés par les avis défavorables. La quatrième affirmation est particulièrement révélatrice, avec un nombre d'avis pratiquement identique entre les deux camps. Cette observation est confortée par les résultats de la première affirmation de la figure 4.15 qui, avec ses 74,29 % des sondés à l'avis favorable, met en avant la communication écrite. Quant aux résultats des troisième, cinquième et sixième affirmations et, bien qu'ils semblent tendres légèrement vers une évaluation favorable, il est difficile d'émettre des observations pertinentes, le taux de neutralité pour ces affirmations étant proche du tiers des sondés. Les sondés semblent donc partagés entre une communication générale, documentation de référence ou formation générique, et une communication à la demande, par questions/réponses.

4.2.3 Technique

Avec 33 sondés ayant répondu aux affirmations de cette section, la figure 4.16 montre les capacités d'innovation et de réutilisation de code et des outils. La figure 4.17 montre l'attrait des sondés relatif à la qualité de code fourni.

L'interprétation des affirmations présentées dans figure 4.16 montre un certain attrait du panel pour la réutilisation de code, mais cet attrait semble toutefois limité. Le panel est favorable (19 sondés sur 33 pour la seconde affirmation) à une réutilisation de code comme base de la nouvelle solution, mais est défavorable (20 sondés sur 33 pour la première affirmation) à une réutilisa-

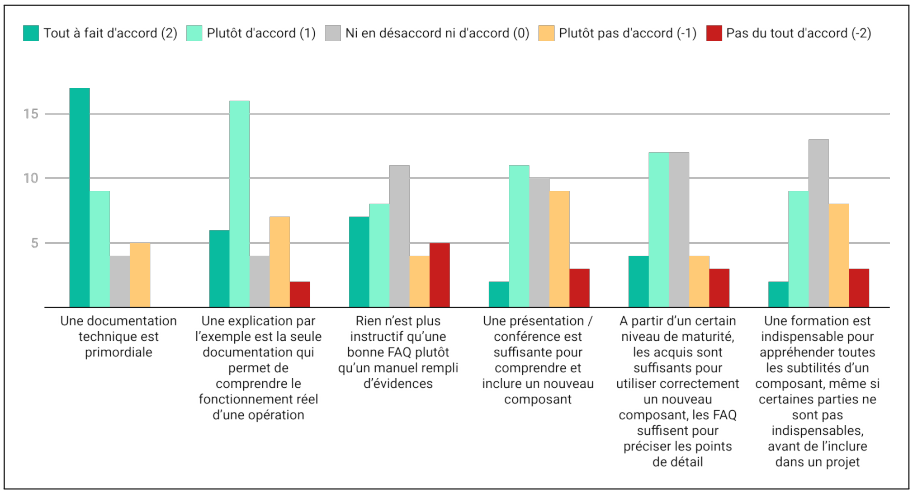


FIGURE 4.15 – Communication : Communication avec le monde extérieur

tion systématique. Il est à noter que la réutilisation avec adaptations mineures (sixième affirmation) partage plus le panel, qui a tendance à préciser dans leur réponse libre que cela va fortement dépendre de l'environnement dans lequel le projet doit être réalisé. A contrario, même si le panel semble plus pencher vers une réutilisation de code, la capacité d'innovation est fortement plébiscitée, avec un attrait certain pour l'innovation (respectivement 20, 18 et 23 avis favorables sur 33 aux troisième, cinquième et septième affirmations), tout en gardant une petite tendance plutôt défavorable (13 avis défavorables sur 33 ainsi que 13 avis neutres pour la quatrième affirmation) à une solution 100% innovante.

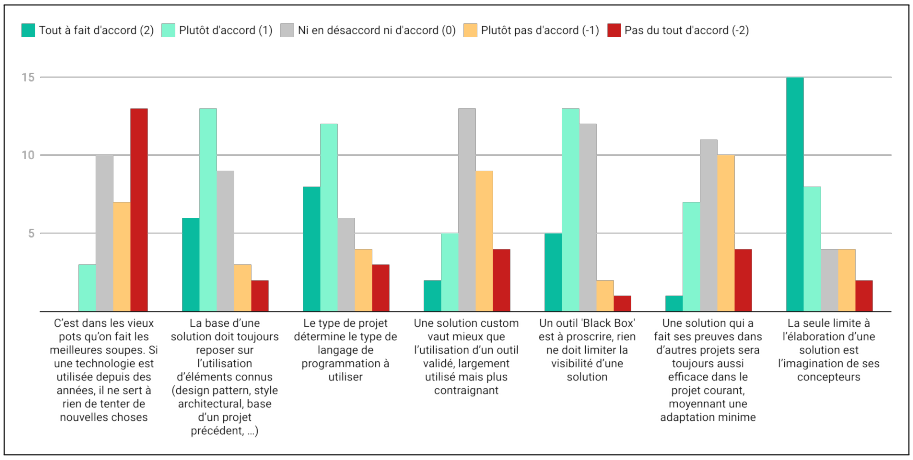


FIGURE 4.16 – Technique - Capacité d'innovation et de réutilisation de code

La figure 4.17 montre clairement que les sondés portent une attention particulière à la qualité de code et sont sensibles aux modèles de conception qui en découlent, mais une étude plus approfondie des réponses libres démontre que certains concepts ne sont pas maitrisés. Par exemple, cinq sondés remettent en cause l'intérêt d'effectuer des tests unitaires sur du code de mauvaise qualité, préférant sans doute jeter ce code et en réécrire un nouveau de meilleure

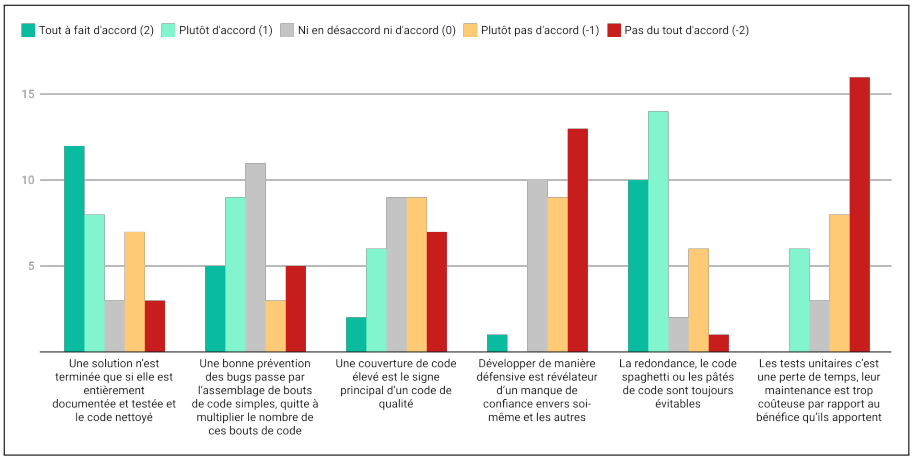


FIGURE 4.17 – Technique - Capacité de qualité de code

qualité. Pourtant, réaliser des tests unitaires sur une fonctionnalité mal codée permet de mettre en évidence les potentiels problèmes de réalisation et donc aider à la réécriture correcte de la fonctionnalité, et de ce fait améliorer la qualité générale du code. Il semble donc que des lacunes dans l'application des modèles mentaux "Top-down" ou "Bottom-up" soient présentes chez certains sondés et plus précisément lors de la transition entre le plan tactique et le plan de mise en oeuvre (voir section 2.3.1 Modèles mentaux de compréhension de programme page 16).

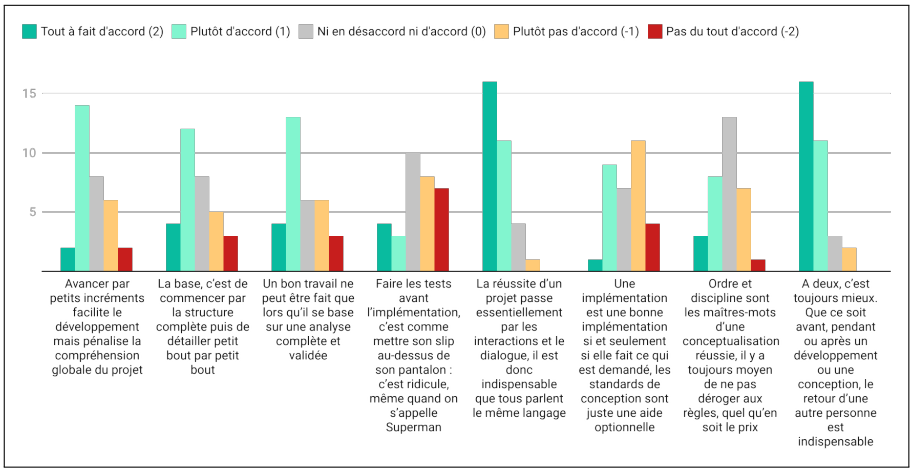


FIGURE 4.18 – Méthodologies

4.2.4 Méthodologie

Dans cette section, 32 sondés ont répondu aux différentes affirmations et, de manière générale, montrent un certain attrait pour les différentes méthodologies suggérées par ces affirmations. Seule la sixième affirmation de la figure 4.18, relative à l'utilisation optionnelle des standards de programmation au profit d'une implémentation sur mesure, présente une évaluation défavorable

(15 avis sur 32). Cette constatation ne fait pas référence directement à une méthodologie, mais confirme les observations effectuées dans la section 4.2.3 Technique, où a été mis en évidence l'attrait des sondés pour l'utilisation des standards de conception et la réutilisation des éléments connus en guise de socle de départ aux développements. Il est à noter que dans les réponses libres des sondés, et comme le montrent les quatrième et dernière affirmations de la figure 4.18, les avis sont massivement favorables (27 avis favorables sur 32 aux deux affirmations) pour le développement en binôme ou du moins la collaboration à plusieurs sur un même thème. Cela renforce également le besoin de définir un modèle mental partagé au sein d'une équipe, dans l'optique d'une production accrue et d'une collaboration facilitée. De plus, les réponses libres montrent une forte adhésion au modèle de développement incrémental (méthodes Agile), comme le soulignent les avis donnés aux deux premières affirmations de la figure 4.18. Mais la troisième affirmation montre le besoin d'une certaine structure, avec l'utilisation d'analyse comme principal guide de conceptualisation, sans pour autant être trop directive. Cette constatation est également renforcée par les avis plutôt partagés des sondés à la septième affirmation faisant l'éloge de la rigueur dans la conceptualisation.

D'un point de vue méthodologique, il semble donc que ce soit le modèle mental "Top-down" qui soit le plus plébiscité, les avis des sondés étant généralement plus favorables à une approche descendante, quelle que soit la technique utilisée. La seconde affirmation montre que les sondés sont assez favorables à une approche Agile, qui préconise une approche descendante pour l'ajout d'une fonctionnalité par petits incréments étudiés et analysés en amont. La troisième affirmation montre un attrait similaire à la précédente affirmation, alors que cette affirmation met en avant l'approche fortement descendante que propose la méthodologie Waterfall. Enfin, la cinquième affirmation, concernant la méthodologie TDD, montre l'attrait des sondés pour l'approche descendante de cette méthodologie, qui requiert une connaissance du domaine lié au programme pour permettre la mise en place de tests pertinents avant l'implémentation de la fonctionnalité.

5 Discussion

La lecture des résultats obtenus lors de l'étude semble s'inscrire dans la lignée des résultats obtenus dans les études précédentes. Ces résultats semblent tout de même apporter des constatations bien plus générales que certaines autres études aux sujets plus précis et effectuées sur un panel de sondés plus restreint. Cependant, les différents constats observés ne semblent pas faciliter la découverte de pistes de réponses aux questions relatives à cette étude. Ce bilan n'est pas étonnant ; les résultats présentés précédemment ne prennent pas réellement en compte deux caractéristiques majeures : (1) La définition concrète des participants, ainsi que leur catégorisation en termes de novices et d'experts, et de ce fait, l'interprétation des résultats sur base de cet axe, et (2) L'étude approfondie des réponses libres de chaque participant, censées fournir des précisions précieuses sur les motivations de leur avis aux affirmations soumises à l'échelle de Likert.

Comme cité dans la section 3.3 relative à la collecte des données (page 29), les réponses libres n'étant pas exploitables au regard de l'objectif de cette étude, cela pousse à considérer la voie du questionnaire à réponses libres non adaptée pour obtenir des résultats exploitables. Confronter les participants aux avis qu'ils ont donnés aux affirmations soumises à l'échelle de Likert au travers d'un entretien aurait peut-être apporté plus de nuances à leurs réponses et aurait probablement mieux répondu aux attentes de cette étude.

Malgré ce constat négatif relatif aux réponses libres fournies par les sondés, l'étude des affirmations soumises à l'échelle de Likert sur base de la définition et la catégorisation des participants, en termes de novices et d'experts, va permettre de donner une tendance réaliste des différentes approches, techniques et méthodologies abordées par chaque partie, mais aussi de déterminer les différences et lacunes que les groupes de participants peuvent présenter en ce qui concerne les modèles mentaux abordés dans le questionnaire.

5.1 Relecture des résultats

Avec des définitions des termes novices et experts plus claires, il est possible de relire les résultats obtenus précédemment de manière différente. Cette relecture doit permettre de mettre en évidence les possibles différences entre les catégories de participants et, éventuellement, détecter des manquements au niveau des modèles mentaux, que ce soit chez les novices ou chez les experts.

Identification : La première faiblesse détectée au niveau de la perception des participants par rapport aux modèles mentaux est celle concernant les pa-

radigmes de programmation. L'introduction dans le questionnaire de 2 méthodologies, basé web (étapes de conception d'un site web) et visuel (conception de solutions facilitée par des outils visuels regroupant, par exemple, le Scrum ou UML), dans les propositions de paradigmes, ont montré qu'un certain nombre de participants, 27 soit 52,94 % (n=51) ne maitrisaient pas le modèle mental de paradigme de programmation. L'analyse de ces résultats, conjuguée avec la catégorisation, montre que 13 participants novices et 14 participants experts ont sélectionné l'un des 2 concepts ou les 2 concepts. Ces résultats montrent qu'en matière de paradigme de programmation, il n'y a pas de différence entre les novices et les experts : le modèle mental semble mal compris par une bonne partie des participants, quel que soit son niveau d'expertise.

Approche globale : Les résultats de l'approche globale permettent de mettre en évidence les tendances des participants à appliquer des modèles mentaux globaux lors de la réalisation de projets. Les résultats en première lecture montrent que les modèles mentaux "Text-based" et "Situation" semblent bien acceptés, avec plus des deux tiers des participants à tendance favorable. La même observation a été faite sur modèles mentaux "Top-down" et "Bottom-up", avec un peu moins des deux tiers des participants. Enfin, le modèle mental "Program" semble présenter une tendance assez neutre. Bien qu'il y ait une certaine tendance favorable visible sur la figure 5.1, le modèle mental "Program" n'est pas bien assimilé par les participants. Une étude croisée des avis donnés par rapport aux affirmations soumises à l'échelle de Likert liées aux autres approches, ainsi que l'étude des réponses libres des participants, montrent que ces derniers, quelle que soit leur catégorie, sont favorables à l'utilisation du modèle mental "Program" comme support - ou valideur - des autres approches, plutôt que d'utiliser ce modèle mental comme approche principale et formateur de solution.

Ces résultats peuvent être affinés avec l'application des catégories novices et experts. Sur les 39 participants ayant répondu à l'entièreté de cette partie du questionnaire, 14 font partie de la catégorie novice et 25 de la catégorie expert. Ce nouvel angle d'interprétation des résultats ne permet pas de dégager des différences notables entre novices et experts. La figure 5.1 montre que les tendances par catégories restent semblables aux résultats obtenus précédemment. Les proportions restent identiques et l'analyse des réponses libres ne donne pas de justifications qui pourraient aider à identifier des différences notables sur les approches appliquées par chacun lors de l'intégration à un projet.

Communication : Sur les 35 participants ayant répondu à cette partie, 21 sont catégorisés comme experts et 14 comme novices. La communication étant la base principale des modèles mentaux partagés, il est intéressant de voir comment les novices et experts voient la communication et comment ils appréhendent cette dernière dans différentes situations.

La première sous-section, concernant la communication intraéquipe, montre une différence entre les novices et les experts minime, dans le sens où les avis des novices semblent moins tranchés. L'approche à la communication verbale ou écrite est similaire pour les deux catégories ; la communication verbale est plébiscitée, tout comme la communication écrite, mais est considérée comme insuffisante pour une bonne communication générale. Par contre, des différences apparaissent en ce qui concerne les points de vue sur la résolution de problème en équipe ou individuelle. Que ce soit les novices ou les experts, les tendances penchent toujours vers une élaboration de solution en groupe ou

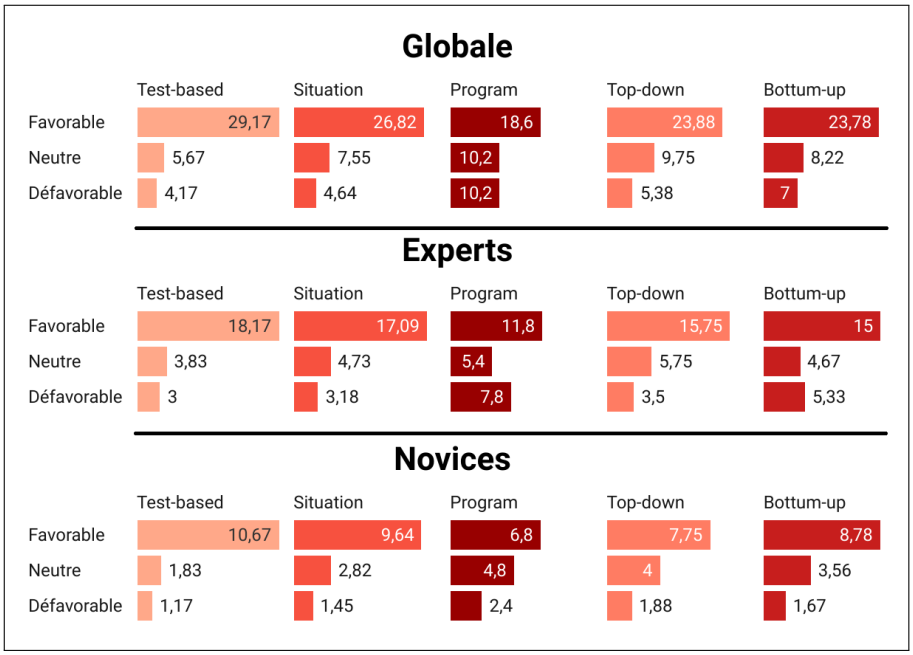


FIGURE 5.1 – Relecture Approche Globale - Moyenne d'adoption des modèles mentaux

de solutions individuelles soutenues par une communication écrite solide, mais cette tendance est moins tranchée du côté des experts. Ces derniers semblent pencher vers une élaboration de solution individuelle, comme le montre la figure 5.2, qui reprend les réponses des novices et experts aux quatre affirmations concernant la résolution de problème au sein d'une équipe. Les résultats relatifs à la dernière affirmation montrent clairement un avis défavorable des experts (38,10 % des experts semblent défavorables à une validation collégiale systématique, contre 14,29 % des novices). L'analyse des réponses libres des experts ayant répondu défavorablement à la dernière affirmation montre qu'ils justifient leur choix par le caractère complexe de certaines solutions, dont la compréhension ne serait pas à la portée de certains membres de l'équipe.

La figure 5.2 montre également les avis moins tranchés des novices concernant la communication intraéquipe, cette tendance se confirmant dans l'analyse des trois premières affirmations de la figure 5.2 et des réponses libres de cette sous-section. Ce constat peut se justifier par le manque d'expérience de travail en équipe, qui est confirmé par les réponses libres des novices (exemple de réponse libre : *"[...] Je considère que la recherche de solution est une responsabilité individuelle et qu'elle ne doit pas être déléguée [...] La prise de décision collégiale est un processus long qui devrait n'être réservé qu'aux décisions importantes. [...]"*).

La communication intraéquipe montrant finalement peu de différences entre les novices et les experts, elle permet de faciliter l'élaboration et la transmission de modèles mentaux partagés.

En ce qui concerne la seconde sous-section, la communication inter-équipe, les différences rencontrées sont similaires à la première sous-section. Les avis des novices sont moins tranchés. Par contre, certains avis semblent montrer un début de divergence d'opinions concernant l'importance de la communica-

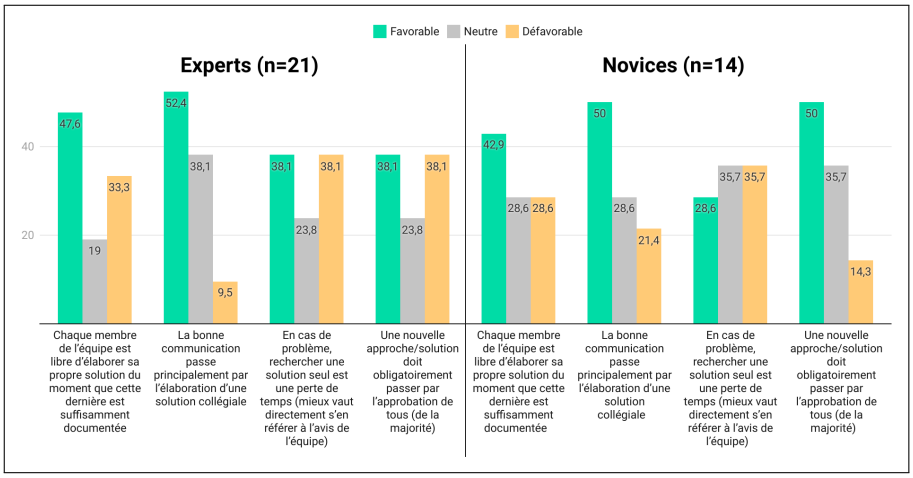


FIGURE 5.2 – Relecture Communication : Communication intraéquipe

tion verbale inter-équipes, ainsi que la communication des solutions trouvées dès leur application. L'interprétation des réponses libres de cette sous-section montre que cette différence d'opinions se concentre essentiellement sur l'hétérogénéité des équipes et sur les objectifs de ces dernières. Les experts mettent en avant les difficultés que les solutions communes peuvent engendrer. Les buts de chaque équipe divergent et les technologies peuvent être incompatibles. Les formations et réunions communes semblent donc moins privilégiées, au profit d'une communication écrite plus souple, moins contraignante. Les novices donnent un avis plus favorable à une communication systématique et collégiale, sans faire de distinction entre la communication verbale ou écrite. Ce constat est confirmé par les résultats de la troisième affirmation de la figure 5.3, qui montre une nette tendance favorable des experts à la communication des problèmes (61,90 % des experts contre seulement 35,71 % des novices) et solutions rencontrés, mais de manière écrite et essentiellement centrée sur l'architecture commune (précisions apportées dans les réponses libres) et montre une certaine retenue des novices sur le sujet, justifiant leur choix par la crainte de rendre interdépendantes les différentes équipes.

Cette sous-section faisant référence aux modèles mentaux partagés plus généraux, liés aux styles architecturaux et aux infrastructures d'entreprises, il semble logique que les novices aient une approche moins adaptée et aient plus de difficultés à appréhender ces différents modèles. Une vision plus globale est nécessaire, une communication ciblée est plus productive qu'une communication systématique, telle que vue par les novices.

La troisième sous-section, la communication avec le monde extérieur, montre des différences plus marquées entre les opinions des novices et des experts. Elles se situent au niveau de la communication écrite, qui semble être privilégiée par les experts, quand les novices, bien que conscients qu'organiser une communication verbale entre acteurs venant de secteurs différents s'avère difficile, continuent à privilégier la communication verbale à la communication écrite. Les résultats repris pour la première affirmation montrent que 80,95 % des experts mettent l'accent sur la communication écrite, quand seulement 64,29 % des novices considèrent la documentation technique comme primordiale, en précisant leur choix, dans les réponses libres, par la nécessité de formations et réunions explicatives (exemple de réponse libre : *"Pour expliquer un projet à*

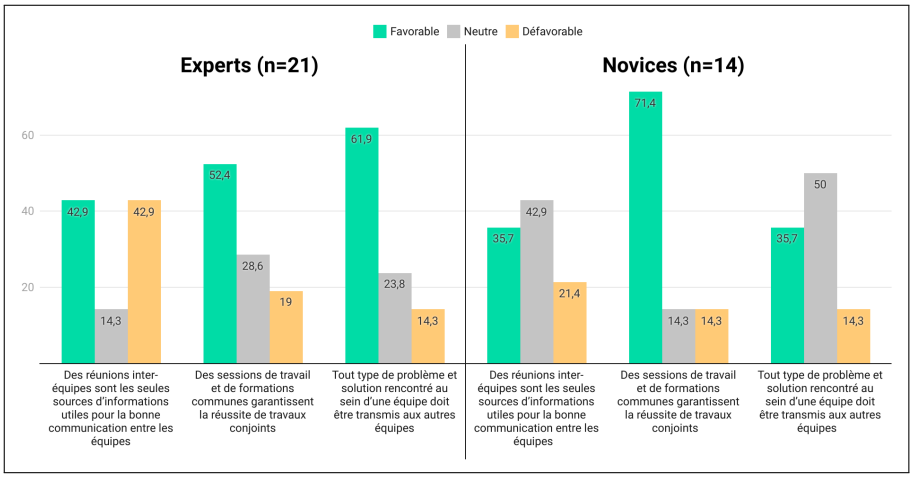


FIGURE 5.3 – Relecture Communication : Communication inter-équipe

l'extérieur, [...]. Une formation peut être nécessaire selon la complexité. [...]").

La figure 5.4 confirme l'observation faite précédemment. Les résultats de la première affirmation montrent qu'une certaine partie des novices est plus hésitante sur l'importance de la communication écrite, tandis que les résultats de la seconde affirmation montrent qu'une FAQ, proche d'une retranscription d'une communication verbale, est plus appréciée par les novices. Par contre, les résultats de la dernière affirmation montrent que les experts, confiants en leurs capacités, peuvent abandonner la communication écrite au profit de FAQ.

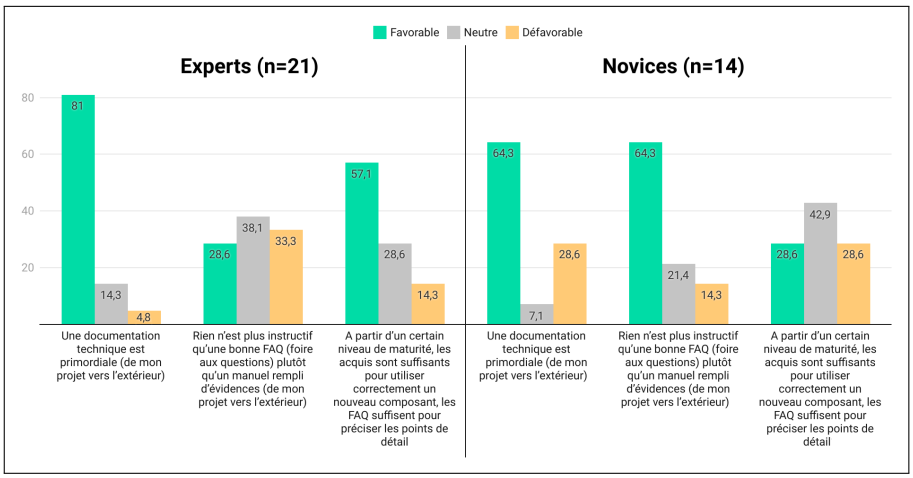


FIGURE 5.4 – Relecture Communication : Communication avec le monde extérieur

Technique : Sur les 33 participants ayant répondu à cette partie, 20 sont catégorisés comme experts et 13 comme novices. Les sujets abordés par cette section montrent deux tendances différentes. Le sujet, concernant les capacités d'innovation et de réutilisation de code et des outils, divise toujours les participants, qu'ils soient novices ou experts. La réutilisation de code est toujours

jugée comme indispensable par les deux catégories, mais doit être limitée pour laisser place à l'innovation. L'analyse des réponses libres des experts montre qu'ils sont plus enclins à l'innovation, afin de garantir la pérennité et se prémunir de la volatilité de certaines technologies. Les novices, quant à eux, sont plus partagés, plébiscitant clairement l'innovation (plus de 80 % d'avis favorables à la troisième affirmation de la firgure 5.5), ils sont également partisans de la réutilisation de code existant. Ces novices pro-innovation et pro-réutilisation ne justifient pas leur point de vue dans les réponses libres. Il serait intéressant de conforter ce panel à une étude plus poussée sur les modèles mentaux "Program", plus propice à la réutilisation de code, et "Situation", plus propice à l'innovation. Enfin, les experts semblent plus partagés que les novices, voire présentent des avis opposés, en ce qui concerne les solutions personnalisées. Cette tendance, bien que confirmée par les résultats de cette étude, n'est pas assez justifiée par les participants pour être approfondie, mais il serait intéressant d'étudier cette tendance, car elle touche aux principes des modèles mentaux partagés. Une solution personnalisée devant être comprise par tous les membres d'une équipe, il est nécessaire que cette équipe partage les mêmes modèles de base, pour éviter une courbe d'apprentissage trop longue. Cette observation semble contredire celle faite au niveau de la communication inter-équipe, où les experts semblaient plus favorables à l'élaboration de solution individuelle, comparée à l'élaboration d'une solution en équipe, tandis que les novices penchent plus pour une solution collective et la validation de leurs pairs.

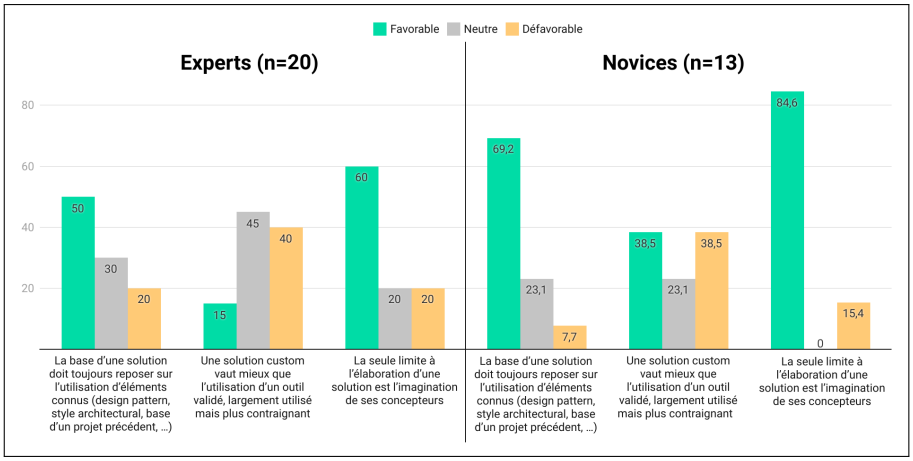


FIGURE 5.5 – Technique - Capacité d'innovation et de réutilisation de code

Le sujet concernant l'importance donnée à la qualité du code fourni montre des avis bien plus tranchés chez les experts que chez les novices, comme le montre la figure 5.6. Les principes de "Clean Code" et "Clean Architecture", qui aide à la formation de modèles mentaux, tel que la découpe du code en méthodes spécialisées ou l'abstraction pour cacher des comportements complexes, sont des principes qui ne sont pas toujours facilement assimilés par les novices. L'importance d'avoir un code testé et propre ne semble pas être une priorité pour les novices, qui justifient leurs choix dans les réponses libres par un manque de temps et de valeur ajoutée pour la fonctionnalité demandée. Cette approche des novices pourrait expliquer en partie les résultats obtenus dans le premier sujet de cette section, où les novices sont plus enclins à l'innovation qu'à la réutilisation du code. Si un code n'est pas propre, il est difficile de le réutiliser sans un effort préalable d'adaptation. Les experts, plus enclins à

la réutilisation, font donc plus attention à la découpe et à la propreté du code pour faciliter, le cas échéant, la réutilisation de code. Cette différence marquée est probablement l'un des plus gros freins à la mise en place de modèles mentaux partagés. Lorsque la différence d'expérience entre les membres d'une équipe est grande, la manière d'aborder certains sujets, tels que la propreté du code ou le test, et il est difficile de trouver un juste milieu qui satisfasse tout le monde et qui pourrait être reproduit lors de l'incorporation d'un nouvel élément à l'équipe. Ici encore, les résultats obtenus ne permettent pas d'avancer plus d'observations pertinentes, les experts n'ayant pas justifié leurs choix et les novices ne justifiant leurs choix qu'avec des raisons extérieures, telles que les contraintes budgétaires et temporelles accordées à la réalisation d'un projet. Néanmoins, ces résultats confortent les observations faites lors de l'état de l'art concernant les modèles mentaux partagés, où il est fait mention de la fragilité de ces derniers, fortement dépendants de la composition des équipes, et de la difficulté rencontrée par les nouveaux arrivants et leur temps d'adaptation plus ou moins long.

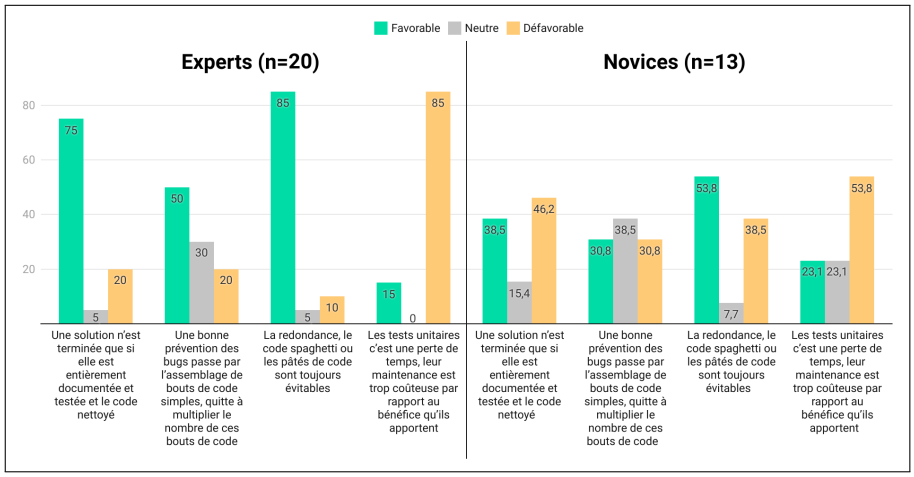


FIGURE 5.6 – Technique - Capacité de qualité de code

Méthodologie : Sur les 32 participants ayant répondu à cette partie, 20 sont catégorisés comme experts et 12 comme novices. Dans cette section, les méthodologies abordées sont autant de modèles mentaux qu'il faut appréhender avec minutie pour en retirer tous les concepts et principes, afin de les mettre en pratique par la suite. Il faut donc une certaine expertise pour avoir eu l'occasion de les appréhender tous, ou du moins de les avoir étudiés sans les pratiquer, et en comprendre les tenants et aboutissants. Il est donc logique que les différences soient plus marquées entre les novices et les experts pour toutes les méthodologies abordées. La figure 5.7 montre les avis sur les affirmations soumises à l'échelle de Likert qui ont une différence plus marquée que les autres affirmations du questionnaire. Les résultats de la première affirmation montrent que les novices ont besoin de plus d'encadrement et de structure que les experts. Les justifications trouvées dans les réponses libres montrent que les novices ont une tendance à apprécier la méthodologie Waterfall plus que les méthodologies Agiles, car ils sont en attente de plus de sécurité qu'une analyse complète et validée peut leur apporter. La seconde affirmation mettant en avant la méthodologie TDD (Test Driven Development) montre une certaine adhésion des experts, les indécis et réfractaires n'ayant

pour la plupart pas d'expérience avec cette méthodologie, quand les novices montrent une grande indécision. Cette observation se justifie par le fait que le TDD est toujours vu comme une méthodologie orientée uniquement sur le test et non comme une méthodologie structurante, influençant la manière de coder et entrant en résonance avec les principes de "Clean Code". Il est donc normal de retrouver des différences similaires entre novices et experts dans la section précédente, relative à la technique et à l'importance donnée à la qualité du code fourni. Les résultats de la troisième affirmation, abordant les styles architecturaux qui font le lien entre les principes exprimés par les méthodologies et la manière d'implémenter ces principes, montrent une certaine division des novices dans l'importance du rôle des styles architecturaux dans les développements, contrairement aux experts, qui reconnaissent l'importance des styles architecturaux, même si certains pensent qu'ils ne participent pas toujours à la réussite d'une implémentation. Que ce soit au niveau des novices ou des experts, cette réticence pourrait se justifier par la méconnaissance des styles architecturaux ou à leur utilisation détournée, mal adaptée, pour résoudre un problème particulier, les réponses libres n'étant pas assez complètes pour corroborer cette hypothèse. La dernière affirmation traite plus généralement du respect strict des méthodologies abordées et les avis sur cette dernière montrent une certaine réticence des novices à déroger aux règles et adapter leur manière de faire, ce qui rentre en contradiction avec les résultats obtenus sur l'affirmation précédente, où les novices étaient plus partagés. Il est difficile de déduire une observation pertinente de ces derniers résultats, vu le peu de justifications dans les réponses libres, mais le manque d'adaptabilité des novices pourrait être lié simplement au manque de maitrise des éléments qu'ils manipulent. Quant aux experts, les résultats montrent une indécision dans le respect strict des principes méthodologiques, généralement justifiée par le manque de liberté des experts dans le choix de ces méthodologies, généralement imposées par l'entreprise ; ils doivent donc s'adapter, faute de pouvoir les faire évoluer ou en changer.

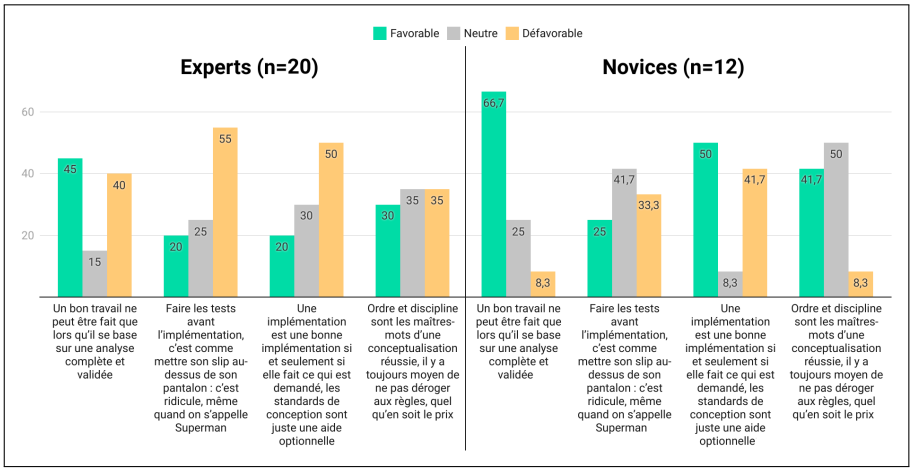


FIGURE 5.7 – Méthodologies

5.2 Limites et biais

Bien que la collecte des résultats, ainsi que la relecture de ces derniers, sur base du profilage des participants, apportent certains éléments de réponses à la question de recherche, il est à noter que le questionnaire proposé aux participants a été construit sur base de quelques suppositions, qui ont une influence non négligeable sur la manière d'interpréter les résultats. Le questionnaire étant une élicitation [1], la plupart des sujets, concepts ou encore modèles mentaux abordés par l'étude ne sont ni définis, ni expliqués. Il est supposé que les participants à l'étude connaissent et comprennent ces concepts. Cette manière d'appréhender l'étude pourrait amener plusieurs interprétations différentes des résultats. C'est le cas, par exemple, du concept de paradigme de programmation proposé dans la section d'identification du public-cible. Une définition formelle n'étant pas fournie aux participants, choisir une proposition qui n'est pas un paradigme de programmation n'est pas forcément un indicateur d'une mauvaise compréhension du concept, mais peut être compris comme une mauvaise interprétation du questionnaire par le participant. Ce genre de considération est une limite du questionnaire proposé et pourrait amener des interprétations différentes des résultats si cette étude devait être renouvelée.

De plus, le fait que le nombre de participants soit assez limité par rapport au but très généraliste visé par cette étude, est une menace évidente à la validité des résultats. Le panel de sondés ne couvrant pas la totalité des profils de programmeurs possibles, ainsi que le nombre limité de répondants par profils identifiés, augmentent le risque d'obtenir des résultats tronqués. Ici aussi, renouveler l'étude et proposer le questionnaire à un panel différent, plus ou moins hétérogène, pourrait amener des résultats différents, voire en contradiction avec les résultats de cette étude.

1. L'élicitation est une manière de soutirer de l'information à une personne ou un groupe, sans que le but réel de la conversation ou de l'interview ne soit divulgué.

6 Conclusion et perspectives

Dans le cadre de ce mémoire, un état de l'art, sous la forme d'une revue systématique de la littérature, a permis de définir le concept de modèles mentaux, ainsi que d'identifier les éléments clés qui ont été utilisés par la suite comme base de l'étude comparative visant à identifier des manquements dans l'application des modèles mentaux lors du processus de compréhension d'un programme.

L'étude préliminaire a mis en évidence l'importance des modèles mentaux lorsqu'il s'agit d'expliquer comment l'esprit d'un individu traite une grande quantité d'informations de manière efficace. Ce concept pouvant être adapté à tout type de domaine, les sciences de l'informatique ne font pas exception. La revue de la littérature a permis d'identifier certains types de modèles mentaux, appliqués au domaine informatique et plus particulièrement à la programmation, qui se basent sur la compréhension du discours et sur la représentation textuelle correspondant à la forme littérale du texte. De là, le nombre de modèles mentaux identifiés a été réduit, pour ne garder que les modèles mentaux qui entrent dans le processus de compréhension de programme. Cette sélection, utile pour la suite de ce mémoire, ne fut pas réalisée sans mal, étant donné l'évidente baisse d'intérêt du monde scientifique concernant l'impact des modèles mentaux lors de la compréhension et de la représentation interne du code écrit dans des langages formels. De plus, le manque d'homogénéité dans la définition du public-cible, ainsi que la difficulté de mener des études à large spectre, n'ont pas aidé à l'identification des modèles mentaux.

Après la rédaction de cet état de l'art et la sélection des modèles mentaux, l'étude comparative a pu être menée. Cette étude a débuté par l'élaboration d'un questionnaire mettant en évidence l'application des modèles mentaux sélectionnés dans le processus de compréhension d'un programme. Ce questionnaire fut proposé à un large public, acteur dans le domaine de la programmation, et les résultats obtenus ont été retranscrits tels quels, pour ensuite être relus à la lumière d'un profilage établi sur base du recensement des participants, ce profilage définissant de manière précise les concepts de novices et experts. Malheureusement, l'analyse de ces résultats n'a pas permis d'apporter toutes les réponses aux questions soulevées par ce mémoire.

L'objectif de ce mémoire était de répondre à la question : **Quels manquements peuvent exister dans l'utilisation des modèles mentaux par les développeurs en fonction de leur niveau d'expertise (novice ou expert) ?** De cette question découlait deux sous-questions : (1) **Quelles sont les caractéristiques qui vont aider à définir le niveau d'expertise d'un développeur : novice ou expert ?** et (2) **Les manquements dans l'utilisation des modèles mentaux sont-ils liés au niveau d'expertise d'un développeur ?** Une réponse à la première sous-question a pu

être donnée grâce au profilage réalisé lors de la première lecture des résultats du questionnaire. Se basant sur l'expérience déclarée des participants, ainsi que sur leur évolution dans leur fonction, des grilles de comparaison ont été proposées et une catégorisation des participants a pu être effectuée. Les niveaux d'expertise novice et expert ont donc reçu une définition stricte, ne tenant pas compte de l'environnement dans lequel le participant évolue, mais bien du niveau d'expertise général de ce dernier. Malheureusement, apporter une réponse à la seconde sous-question, sur base de la relecture des résultats à la lumière du profilage effectué, s'est avéré beaucoup moins aisé. L'objectif de la relecture des résultats était d'identifier des divergences d'opinions entre les novices et les experts. Bien que certains résultats montraient un écart non négligeable entre les profils des répondants, d'autres résultats traitant du même sujet montraient une quasi égalité. Les questions étant trop génériques et les réponses libres des participants n'apportant pas assez de précision pour permettre d'identifier les points de divergences, la relecture des résultats n'a permis, de manière univoque, ni d'identifier des manquements dans l'utilisation des modèles mentaux, ni de lier ces manquements à un niveau d'expertise en particulier. Enfin, ne pouvant pas fournir une réponse claire à la seconde sous-question, la question de recherche ne peut se voir attribuer de réponse ferme. Toutefois, les résultats obtenus permettent d'émettre une nouvelle interrogation : **Les modèles mentaux entrant dans le processus de compréhension de programme ne sont-ils pas trop génériques pour permettre l'identification et l'attribution de manquement à une catégorie de développeur ?**

La difficulté de répondre à la question de recherche met en évidence les limites de la méthode utilisée dans ce mémoire. Traiter, avec un simple questionnaire en ligne, un sujet aussi général que celui des modèles mentaux entrant dans le processus de compréhension de programme, ne semble donc pas la méthode idéale. A la lecture des résultats, il a été montré que le nombre de réponses décroissait au fur et à mesure des questions posées. Le questionnaire, peut-être un peu trop long, a découragé les répondants. De plus, la forme des questions, une suite d'affirmations soumises à l'échelle de Likert, n'a pas rencontré le but souhaité, qui était de récolter, non seulement une appréciation des répondants sur les affirmations proposées, mais également de les pousser à compléter leur avis dans les réponses libres prévues à cet effet. Le format de récolte de données de type entretien individuel aurait été un meilleur choix pour obtenir des détails plus précis sur la manière d'aborder le sujet auprès des différents participants. Cependant, ce mémoire ayant comme objectif de proposer un profilage qui pourrait être utilisé dans d'autres études et situations, un grand nombre de participants était nécessaire, ce qui aurait été trop chronophage et incompatible avec le contexte de réalisation de ce mémoire.

Les résultats de cette étude, et en particulier la proposition de profilage, pourraient être utilisés dans de futures études traitant de modèles mentaux. Le profilage propose une réponse au problème d'hétérogénéité du public-cible identifié lors de la revue de la littérature. Cela permettrait de revoir certains résultats d'études sous un prisme commun et donc de détecter d'éventuelles similitudes. De plus, un exercice équivalent pourrait être mené sur des modèles mentaux moins génériques que les modèles mentaux entrant dans le processus de compréhension de programme. Cela permettrait de répondre à l'interrogation émise dans cette conclusion et permettrait de continuer la réflexion sur l'identification des manquements dans l'utilisation des modèles mentaux. Une autre piste d'étude serait d'identifier un problème précis, mais assez générique pour ne pas trop être dépendant d'un contexte particulier. Ce problème pourrait ensuite être soumis à des programmeurs de tous horizons, quelque

soit leur paradigme ou leur langage de programmation de prédilection, afin d'observer leurs réactions, leur manière d'aborder le problème et d'identifier les modèles mentaux en action. Enfin, traiter de manière générique le sujet des modèles mentaux, en ne se limitant pas à un programme ou une portion de programme en particulier, a montré ses limites, mais il serait intéressant de comparer les résultats de cette étude avec ceux d'une étude qui aurait les moyens de soumettre le questionnaire par le biais d'entretiens individuels. Les détails obtenus par ce type d'étude pourraient apporter un éclairage nouveau sur les différences dans l'application des modèles mentaux entre les novices et les experts et, éventuellement, identifier des manquements dans ces applications.

Références

Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. *Memory & cognition*, *9*(4), 422-433.

Adelson, B. (1984). When novices surpass experts : The difficulty of a task may increase with expertise. *Journal of Experimental Psychology : Learning, Memory, and Cognition*, *10*(3), 483.

Alardawi, A. S. & Agil, A. M. (2015). Novice comprehension of object-oriented OO programs : an empirical study, In *2015 World Congress on Information Technology and Computer Applications (WCITCA)*. IEEE.

Balijepally, V., Nerur, S. & Mahapatra, R. (2015). Task mental model and software developers' performance : an experimental investigation. *Communications of the Association for Information Systems*, *36*(1), 4.

Barfield, W. (1986). Expert-novice differences for software : Implications for problem-solving and knowledge acquisition. *Behaviour & Information Technology*, *5*(1), 15-29.

Bergantz, D. & Hassell, J. (1991). Information relationships in PROLOG programs : how do programmers comprehend functionality ? *International Journal of Man-Machine Studies*, *35*(3), 313-328.

Bidlake, L., Aubanel, E. & Voyer, D. (2020). Systematic literature review of empirical studies on mental representations of programs. *Journal of Systems and Software*, *165*, 110565.

Boone, H. N. & Boone, D. A. (2012). Analyzing likert data. *Journal of extension*, *50*(2), 1-5.

Brannick, M. T., Salas, E. & Prince, C. W. (1997). *Team performance assessment and measurement : Theory, methods, and applications*. Psychology Press.

Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International journal of man-machine studies*, *18*(6), 543-554.

Burkhardt, J.-M., Détienne, F. & Wiedenbeck, S. (1997). Mental representations constructed by experts and novices in object-oriented program comprehension, In *Human-Computer Interaction INTERACT'97*. Springer.

Burkhardt, J.-M., Détienne, F. & Wiedenbeck, S. (2002). Object-oriented program comprehension : Effect of expertise, task and phase. *Empirical Software Engineering*, *7*(2), 115-156.

Burkhardt, J.-M., Détienne, F. & Wiedenbeck, S. (1998). The effect of object-oriented programming expertise in several dimensions of comprehension strategies, In *Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No. 98TB100242)*. IEEE.

Butcher, K. R. (2006). Learning from text with diagrams : Promoting mental model development and inference generation. *Journal of educational psychology*, *98*(1), 182.

Canas, J. J., Bajo, M. T. & Gonzalvo, P. (1994). Mental models and computer programming. *International Journal of Human-Computer Studies*, *40*(5), 795-811.

Corritore, C. L. & Wiedenbeck, S. (1991). What do novices learn during program comprehension ? *International Journal of Human-Computer Interaction*, *3*(2), 199-222.

Corritore, C. L. & Wiedenbeck, S. (1999). Mental representations of expert procedural and object-oriented programmers in a software maintenance task. *International Journal of Human-Computer Studies*, *50*(1), 61-83.

Ehrlich, M.-F. & Tardieu, H. (1993). Modèles mentaux, modèles de situation et compréhension de textes. *Les Modèles Mentaux : approche cognitive des représentations*, 47-78.

Fix, V., Wiedenbeck, S. & Scholtz, J. (1993). Mental representations of programs by novices and experts, In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*.

Furlough, C. S. Et al. (2017). Mental Model Structures : Differences at Multiple Levels of Experience.

Hmelo-Silver, C. E. & Pfeffer, M. G. (2004). Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive science*, *28*(1), 127-138.

Holland, S., Griffiths, R. & Woodman, M. (1997). Avoiding object misconceptions, In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*.

Huang, J. & Cakmak, M. (2015). Supporting mental model accuracy in trigger-action programming, In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*.

Johnson, T. E., Lee, Y., Lee, M., O'Connor, D. L., Khalil, M. K. & Huang, X. (2007). Measuring sharedness of team-related knowledge : Design and validation of a shared mental model instrument. *Human Resource Development International*, *10*(4), 437-454.

Johnson, T. E., Top, E. & Yukselturk, E. (2011). Team shared mental model as a contributing factor to team performance and students' course satisfaction in blended courses. *Computers in Human Behavior*, *27*(6), 2330-2338.

Johnson-Laird, P. N. (1980). Mental models in cognitive science. *Cognitive science*, *4*(1), 71-115.

Johnson-Laird, P. N. (1983). *Mental models : Towards a cognitive science of language, inference, and consciousness*. Harvard University Press.

Kaczmarczyk, L. C., Petrick, E. R., East, J. P. & Herman, G. L. (2010). Identifying student misconceptions of programming, In *Proceedings of the 41st ACM technical symposium on Computer science education*.

Kaptein, M. C., Nass, C. & Markopoulos, P. (2010). Powerful and consistent analysis of likert-type rating scales, In *Proceedings of the SIGCHI conference on human factors in computing systems*.

Kieras, D. E. & Bovair, S. (1984). The role of a mental model in learning to operate a device. *Cognitive science*, *8*(3), 255-273.

Klimoski, R. & Mohammed, S. (1994). Team mental model : Construct or metaphor ? *Journal of management*, *20*(2), 403-437.

Letovsky, S. (1987). Cognitive processes in program comprehension. *Journal of Systems and software*, *7*(4), 325-339.

Mayer, R. E. (1994). Visual aids to knowledge construction : Building mental representations from pictures and words, In *Advances in psychology*. Elsevier.

Mayer, R. E. & Gallini, J. K. (1990). When is an illustration worth ten thousand words ? *Journal of educational psychology*, *82*(4), 715.

McKeithen, K. B., Reitman, J. S., Rueter, H. H. & Hirtle, S. C. (1981). Knowledge organization and skill differences in computer programmers. *Cognitive Psychology*, *13*(3), 307-325.

Meunier, J.-P. & Berten, A. (1995). A propos de Philip Johnson-Laird," L'ordinateur et l'esprit". *Recherches en communication*, *4*, 243-256.

Mills, C. B., Diehl, V. A., Birkmire, D. P. & Mou, L.-C. (1995). Reading procedural texts : Effects of purpose for reading and predictions of reading comprehension models. *Discourse Processes*, *20*(1), 79-107.

Mohammed, S., Klimoski, R. & Rentsch, J. R. (2000). The measurement of team mental models : We have no shared schema. *Organizational Research Methods*, *3*(2), 123-165.

Muhamad, S. (2012). Graduate employability and transferable skills : A review. *Advances in Natural and Applied Sciences*, *6*(6), 882-885.

Navarro-Prieto, R. & Canas, J. J. (2001). Are visual programming languages better ? The role of imagery in program comprehension. *International Journal of Human-Computer Studies*, *54*(6), 799-829.

Ofoleta, K. C. (2015). New graduates' knowledge of Software development works : what they need and how they learn.

Parkin, P. (2004). An exploratory study of code and document interactions during task-directed program comprehension, In *2004 Australian Software Engineering Conference. Proceedings.* IEEE.

Pennington, N. (1987a). Comprehension strategies in programming, In *Empirical Studies of Programmers : Second Workshop, 1987*.

Pennington, N. (1987b). Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, *19*(3), 295-341.

Quillin, K. & Thomas, S. (2015). Drawing-to-learn : a framework for using drawings to promote model-based reasoning in biology. *CBE—Life Sciences Education*, *14*(1), es2.

Ramalingam, V. & Wiedenbeck, S. (1997). An empirical study of novice program comprehension in the imperative and object-oriented styles, In *Papers presented at the seventh workshop on Empirical studies of programmers*.

Romero, P. & du Boulay, B. (2004). Structural knowledge and language notational properties in program comprehension, In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*. IEEE.

Schmidt, C., Kude, T., Heinzl, A. & Mithas, S. (2014). How agile practices influence the performance of software development teams : The role of shared mental models and backup.

Schwamb, K. B. (1990). Mental models : A survey. *URL : citeseer. nj. nec. com/schwamb90mental. html*.

Shneiderman, B. (1976). Exploratory experiments in programmer behavior. *International Journal of Computer & Information Sciences*, *5*(2), 123-143.

Shneiderman, B. & Mayer, R. (1979). Syntactic/semantic interactions in programmer behavior : A model and experimental results. *International Journal of Computer & Information Sciences*, *8*(3), 219-238.

Soloway, E., Adelson, B. & Ehrlich, K. (1988). Knowledge and processes in the comprehension of computer programs. *The nature of expertise*, 129-152.

Soloway, E. & Ehrlich, K. (1984). Empirical studies of programming knowledge. *IEEE Transactions on software engineering*, (5), 595-609.

Storey, M.-A., Fracchia, F. D. & Müller, H. A. (1999). Cognitive design elements to support the construction of a mental model during software exploration. *Journal of Systems and Software*, *44*(3), 171-185.

Sullivan, G. M. & Artino Jr, A. R. (2013). Analyzing and interpreting data from Likert-type scales. *Journal of graduate medical education*, *5*(4), 541-542.

Van den Bossche, P., Gijselaers, W., Segers, M., Woltjer, G. & Kirschner, P. (2011). Team learning : building shared mental models. *Instructional Science*, *39*(3), 283-301.

Van Dijk, T. A., Kintsch, W. Et al. (1983). Strategies of discourse comprehension.

Vessey, I. (1987). On matching programmers' chunks with program structures : An empirical investigation. *International Journal of Man-Machine Studies*, *27*(1), 65-89.

Von Mayrhauser, A. & Vans, A. M. (1995). Program comprehension during software maintenance and evolution. *Computer*, *28*(8), 44-55.

Von Mayrhauser, A. & Vans, A. M. (1996). Identification of dynamic comprehension processes during large scale maintenance. *IEEE Transactions on Software Engineering*, *22*(6), 424-437.

Von Mayrhauser, A. & Vans, A. M. (1998). Program understanding behavior during adaptation of large scale software, In *Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No. 98TB100242)*. IEEE.

von Mayrhauser, A. & Vans, A. M. (1994). Comprehension processes during large scale maintenance, In *Proceedings of 16th International Conference on Software Engineering*. IEEE.

Wasserman, J. A. & Koban, K. (2019). Bugs on the brain : A mental model matching approach to cognitive skill acquisition in a strategy game. *Journal of Expertise/June*, *2*(2).

Weiser, M. (1982). Programmers use slices when debugging. *Communications of the ACM*, *25*(7), 446-452.

Wiedenbeck, S., Fix, V. & Scholtz, J. (1993). Characteristics of the mental representations of novice and expert programmers : an empirical study. *International Journal of Man-Machine Studies*, *39*(5), 793-812.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S. & Corritore, C. L. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*(3), 255-282.

Annexe A - Questionnaire

Cette annexe présente la totalité des questions et des affirmations soumises à l'échelle de Likert proposées aux participants répondant au sondage effectué avec l'outil Drag'n Survey.

Le sondage fut proposé sous le titre :
Modèles mentaux - élicitation des techniques de conception et d'intégration de projets.

Partie I : Identification

L'identification du répondant propose huit questions à choix multiple (à réponse unique ou multiple) ainsi que une question à réponse libre.

1. Vous êtes :
   - ❍ Une femme
   - ❍ Un homme
   - ❍ Autre
2. Vous êtes âgé entre :
   - ❍ Moins de 18 ans
   - ❍ 18-25 ans
   - ❍ 25-35 ans
   - ❍ 35-45 ans
   - ❍ 45-55 ans
   - ❍ Plus de 55 ans
3. Votre plus haut niveau d'études :
   - ❍ Aucun
   - ❍ Secondaire suppérieur
   - ❍ Supérieur type court ou bachelier
   - ❍ Supérieur type long ou niveau universitaire
4. Fonction actuelle principale :
   - ❍ Etudiant/Stagiaire
   - ❍ Développement
   - ❍ Analyse et développement
   - ❍ Architecture
   - ✏ Autre : (réponse libre)

5. Nombre d'années d'expérience dans cette fonction principale :
   - ❍ Aucune (étudiant ou nouvel engagé)
   - ❍ 1 à 3 ans
   - ❍ 3 à 5 ans
   - ❍ 5 à 10 ans
   - ❍ 10 à 20 ans
   - ❍ Plus de 20 ans
6. Nombre d'années d'expérience dans le domaine informatique en général :
   - ❍ Aucune (étudiant ou nouvel engagé)
   - ❍ 1 à 3 ans
   - ❍ 3 à 5 ans
   - ❍ 5 à 10 ans
   - ❍ 10 à 20 ans
   - ❍ Plus de 20 ans
7. Nombre d'années d'ancienneté dans votre poste actuel (ou nombre d'années en tant qu'indépendant) :
   - ❍ Aucune (étudiant ou nouvel engagé)
   - ❍ 1 à 3 ans
   - ❍ 3 à 5 ans
   - ❍ 5 à 10 ans
   - ❍ 10 à 20 ans
   - ❍ Plus de 20 ans
8. Paradigmes de programmation pratiqués :
   - ❑ Impératif (procédural)
   - ❑ Déclaratif (fonctionnel ou logique)
   - ❑ Orienté objet
   - ❑ Concurrent
   - ❑ Visuel
   - ❑ Evènementiel
   - ❑ Basé web
9. Langages de programmation de prédilection :
   - ✏ Langage 1 (réponse libre)
   - ✏ Langage 2 (réponse libre)
   - ✏ Langage 3 (réponse libre)

Partie II : Approche globale

Description de la section fournie aux participants :

Cette partie se concentre sur l'approche personnelle que vous pouvez avoir lorsque que vous débutez un nouveau projet ou que vous rejoignez un projet en cours de développement.

Veuillez répondre aux affirmations suivantes et ensuite préciser vos réponses par une explication globale. Les valeurs des tableaux ci-dessous correspondent à :

- ☞ *2 : Tout à fait d'accord*
- ☞ *1 : Plutôt d'accord*
- ☞ *0 : Ni en désaccord ni d'accord*
- ☞ *-1 : Plutôt pas d'accord*
- ☞ *-2 : Pas du tout d'accord*

Lorsque je débute un nouveau développement :

	2	**1**	**0**	**-1**	**-2**
1. J'ai besoin d'un analyse complète des fonctionnalités	○	○	○	○	○
2. J'ai besoin d'une structure de données détaillée	○	○	○	○	○
3. J'ai besoin d'une définition des fonctionnalités principales (contrat d'utilisation)	○	○	○	○	○
4. J'ai besoin d'une définition d'écrans	○	○	○	○	○
5. J'ai besoin de connaître les standards de l'entreprise avant de commencer à imaginer la solution	○	○	○	○	○
6. J'ai besoin de faire le tour des technologies possibles avant de commencer à imaginer la solution	○	○	○	○	○
7. J'ai besoin d'un exemple d'un produit similaire (fonctionnellement ou techniquement)	○	○	○	○	○
8. Je tente de reproduire une solution que je maîtrise déjà en m'adaptant aux besoins du produit	○	○	○	○	○
9. Je tente de mettre en pratique la dernière solution que j'ai pu découvrir lors d'une formation ou d'une conférence	○	○	○	○	○

Pouvez-vous préciser votre approche dans le cadre d'un nouveau développement ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "Je suis entièrement d'accord avec l'affirmation 4 car j'ai besoin de visualiser la solution avant de ... "

Lorsque je rejoins un projet en cours de développement :

	2	**1**	**0**	**-1**	**-2**
1. J'ai besoin d'un analyse complète des fonctionnalités	○	○	○	○	○
2. J'ai besoin d'une structure de données détaillée	○	○	○	○	○
3. J'ai besoin d'une définition des fonctionnalités principales (contrat d'utilisation)	○	○	○	○	○
4. J'ai besoin d'une définition d'écrans	○	○	○	○	○
5. J'ai besoin de connaître les standards de l'entreprise avant de commencer à imaginer la solution	○	○	○	○	○
6. J'ai besoin d'analyser en détail les développements déjà en place pour me familiariser avec l'existant	○	○	○	○	○
7. J'ai besoin de démarrer sur des développements de type correction pour me familiariser avec l'existant	○	○	○	○	○
8. J'ai besoin de tester l'application existante (tests unitaires ou fonctionnels)	○	○	○	○	○
9. Je tente de reproduire une solution que je maîtrise déjà en m'adaptant auxbesoins du produit	○	○	○	○	○
10. Je tente de mettre en pratique la dernière solution que j'ai pu découvrir lors d'une formation ou d'une conférence	○	○	○	○	○

Pouvez-vous préciser votre approche dans le cadre d'une intégration à un projet existant ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "En ce qui concerne l'affirmation 7, cela dépend de la complexité de l'application ..."

Partie III : Communication

Description de la section fournie aux participants :

Cette partie se concentre sur les moyens utilisés pour communiquer et échanger de l'information avec les membres de votre équipe, avec d'autres équipes ou encore avec le monde extérieur.

Veuillez répondre aux affirmations suivantes et ensuite préciser vos réponses par une explication globale. Les valeurs des tableaux ci-dessous correspondent à :

- ☞ *2 : Tout à fait d'accord*
- ☞ *1 : Plutôt d'accord*
- ☞ *0 : Ni en désaccord ni d'accord*
- ☞ *-1 : Plutôt pas d'accord*
- ☞ *-2 : Pas du tout d'accord*

Au sein de mon équipe, je considère que :

	2	**1**	**0**	**-1**	**-2**
1. La communication verbale est la forme de communication la plus productive	○	○	○	○	○
2. Le transfert de connaissances par documentation externe est indispensable	○	○	○	○	○
3. Le code/les diagrammes se suffisent à eux-mêmes et n'ont pas besoin de documentation supplémentaire	○	○	○	○	○
4. Chaque membre de l'équipe est libre d'élaborer sa propre solution du moment que cette dernière est suffisamment documentée	○	○	○	○	○
5. La bonne communication passe principalement par l'élaboration d'une solution collégiale	○	○	○	○	○
6. En cas de problème, rechercher une solution seul est une perte de temps (mieux vaut directement s'en référer à l'avis de l'équipe)	○	○	○	○	○
7. Une nouvelle approche/solution doit obligatoirement passer par l'approbation de tous (de la majorité)	○	○	○	○	○

Pouvez-vous expliquer en quoi la communication au sein de votre équipe a une influence sur votre travail et quels sont les éléments que vous jugez indispensable à la réussite d'un projet ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "Je ne suis pas totalement d'accord avec l'affirmation 1 car la communication verbale ... "

Pour communiquer avec d'autres équipes, je considère que :

	2	**1**	**0**	**-1**	**-2**
1. Une documentation globale est indispensable et suffisante	○	○	○	○	○
2. Des réunions inter-équipes sont les seules sources d'informations utiles pour la bonne communication entre les équipes	○	○	○	○	○
3. Des sessions de travail et de formations communes garantissent la réussite de travaux conjoints	○	○	○	○	○
4. Une structure similaire entre les projets est indispensable pour faciliter la communication	○	○	○	○	○
5. Il est indispensable d'identifier une unité centralisatrice et décisionnelle pour acter les décisions et veiller à leur respect	○	○	○	○	○
6. Tout type de problème et solution rencontré au sein d'une équipe doit être transmis aux autres équipes	○	○	○	○	○

Pouvez-vous expliquer en quoi la communication inter-équipe a une influence sur votre travail et quels sont les éléments que vous jugez indispensable à la réussite d'un projet ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "Je suis moyennement d'accord avec l'affirmation 6 car ..."

Pour une bonne communication avec le monde extérieur, je considère que :

	2	**1**	**0**	**-1**	**-2**
1. Une documentation technique est primordiale (de mon projet vers l'extérieur)	○	○	○	○	○
2. Une explication par l'exemple est la seule documentation qui permet de comprendre le fonctionnement réel d'une opération (de mon projet vers l'extérieur)	○	○	○	○	○
3. Rien n'est plus instructif qu'une bonne FAQ (foire aux questions) plutôt qu'un manuel rempli d'évidences (de mon projet vers l'extérieur)	○	○	○	○	○
4. Une présentation / conférence est suffisante pour comprendre et inclure un nouveau composant	○	○	○	○	○
5. A partir d'un certain niveau de maturité, les acquis sont suffisants pour utiliser correctement un nouveau composant, les FAQ suffisent pour préciser les points de détail	○	○	○	○	○
6. Une formation est indispensable pour appréhender toutes les subtilités d'un composant, même si certaines parties ne sont pas indispensables, avant de l'inclure dans un projet	○	○	○	○	○

Pouvez-vous expliquer en quoi la manière de communiquer avec l'extérieur influence votre travail, que ce soit une communication entrante ou sortante ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "Il est faux de penser qu'une FAQ ... "

Partie IV : Technique

Description de la section fournie aux participants :

Cette partie se concentre sur les aspects techniques de mise en œuvre pour la réalisation d'un projet.

Veuillez répondre aux affirmations suivantes et ensuite préciser vos réponses par une explication globale. Les valeurs du tableau ci-dessous correspondent à :

- ☞ *2 : Tout à fait d'accord*
- ☞ *1 : Plutôt d'accord*
- ☞ *0 : Ni en désaccord ni d'accord*
- ☞ *-1 : Plutôt pas d'accord*
- ☞ *-2 : Pas du tout d'accord*

Je considère que :

	2	1	0	-1	-2
1. C'est dans les vieux pots qu'on fait les meilleures soupes. Si une technologie est utilisée depuis des années, il ne sert à rien de tenter de nouvelles choses	○	○	○	○	○
2. La base d'une solution doit toujours reposer sur l'utilisation d'éléments connus (design pattern, style architectural, base d'un projet précédent, ...)	○	○	○	○	○
3. Le type de projet détermine le type de langage de programmation à utiliser	○	○	○	○	○
4. Une solution custom vaut mieux que l'utilisation d'un outil validé, largement utilisé mais plus contraignant	○	○	○	○	○
5. Un outil "Black Box" est à proscrire, rien ne doit limiter la visibilité d'une solution	○	○	○	○	○
6. Une solution n'est terminée que si elle est entièrement documentée et testée et le code nettoyé	○	○	○	○	○
7. Une bonne prévention des bugs passe par l'assemblage de bouts de code simples, quitte à multiplier le nombre de ces bouts de code	○	○	○	○	○
8. Une solution qui a fait ses preuves dans d'autres projets sera toujours aussi efficace dans le projet courant, moyennant une adaptation minime	○	○	○	○	○
9. La seule limite à l'élaboration d'une solution est l'imagination de ses concepteurs	○	○	○	○	○
10. Une couverture de code élevé est le signe principal d'un code de qualité	○	○	○	○	○
11. Développer de manière défensive est révélateur d'un manque de confiance envers soi-même et les autres	○	○	○	○	○
12. La redondance, le code spaghetti ou les pâtés de code sont toujours évitables	○	○	○	○	○
13. Les tests unitaires c'est une perte de temps, leur maintenance est trop coûteuse par rapport au bénéfice qu'ils apportent	○	○	○	○	○

Pouvez-vous détailler vos réponses en précisant quels sont les éléments que vous considérez comme important à prendre en compte ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "Je ne suis pas d'accord avec l'affirmation 9 car l'imagination n'est pas le seul facteur à prendre en compte, il faut aussi considérer ... "

Partie V : Méthodologie

Description de la section fournie aux participants :

Cette dernière partie se concentre sur les aspects méthodologiques et conceptuels supportant la réalisation d'un projet.

Veuillez répondre aux affirmations suivantes et ensuite préciser vos réponses par une explication globale. Les valeurs du tableau ci-dessous correspondent à :

- ☞ *2 : Tout à fait d'accord*
- ☞ *1 : Plutôt d'accord*
- ☞ *0 : Ni en désaccord ni d'accord*
- ☞ *-1 : Plutôt pas d'accord*
- ☞ *-2 : Pas du tout d'accord*

Je considère que :

	2	**1**	**0**	**-1**	**-2**
1. Avancer par petits incréments facilite le développement mais pénalise la compréhension globale du projet	○	○	○	○	○
2. La base, c'est de commencer par la structure complète puis de détailler petit bout par petit bout	○	○	○	○	○
3. Un bon travail ne peut être fait que lors qu'il se base sur une analyse complète et validée	○	○	○	○	○
4. Faire les tests avant l'implémentation, c'est comme mettre son slip au-dessus de son pantalon : c'est ridicule, même quand on s'appelle Superman	○	○	○	○	○
5. La réussite d'un projet passe essentiellement par les interactions et le dialogue, il est donc indispensable que tous parlent le même langage	○	○	○	○	○
6. Une implémentation est une bonne implémentation si et seulement si elle fait ce qui est demandé, les standards de conception sont juste une aide optionnelle	○	○	○	○	○
7. Ordre et discipline sont les maîtres-mots d'une conceptualisation réussie, il y a toujours moyen de ne pas déroger aux règles, quel qu'en soit le prix	○	○	○	○	○
8. A deux, c'est toujours mieux. Que ce soit avant, pendant ou après un développement ou une conception, le retour d'une autre personne est indispensable	○	○	○	○	○

Pouvez-vous détailler vos réponses en précisant quels sont les éléments que vous considérez comme important à prendre en compte ? Vous pouvez vous reposer sur les affirmations proposées. Exemple : "Je ne suis pas d'accord

avec l'affirmation 7 car il est parfois préférable de sortir des sentiers battus ..."