Chao : Un framework pour le développement de systèmes supportant l'orchestration d'activités sur tablettes en classe

Patrick Wang

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 août 2006

Présentée par

PATRICK WANG

Thèse dirigée par **PIERRE TCHOUNIKINE**
et co-encadrée par **MATTHIEU QUIGNARD**

préparée au sein **Laboratoire d'Informatique de Grenoble**
et de **École Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Chao : Un framework pour le développement de systèmes d'orchestration d'EIAH sur tablettes en classe

Thèse soutenue publiquement le **5 juillet 2016**,
devant le jury composé de :

Mme Mireille BETRANCOURT
Professeure des Universités, Université de Genève, Présidente

M. Serge GARLATTI
Professeur des Universités, Télécom Bretagne, Rapporteur

M. Thierry NODENOT
Professeur des Universités, IUT Bayonne, Rapporteur

M. Pierre TCHOUNIKINE
Professeur des Universités, Université Grenoble-Alpes, Directeur de thèse

M. Matthieu QUIGNARD
Chargé de recherche, CNRS, ENS Lyon, Co-encadrant de thèse

我从来没想过Patrick有一天可以成为博士。
Mon père à l'un de ses amis.

REMERCIEMENTS

La thèse... Cet exercice est parfois assimilé à un travail en solitaire alors qu'il est en fait parsemé de rencontres toutes plus enrichissantes les unes que les autres. Les quelques petits mots qui suivent ne sont qu'une bien pauvre tentative pour exprimer toute ma reconnaissance envers les personnes avec qui j'ai pu partager mon voyage. J'espère que vous savez à quel point je vous porte haut dans mon cœur.

Mes premiers remerciements s'adressent à Pierre et à Matthieu pour avoir encadré cette thèse et fait en sorte que celle-ci se déroule dans les meilleurs conditions possibles. C'est grâce votre patience, votre disponibilité, votre rigueur, vos encouragements, et votre enthousiasme que je suis devenu le chercheur (et aussi la personne) que je suis aujourd'hui. Quelques bosses sont venues accidenter le parcours, mais nous les avons surmontées ensemble sans encombre. La randonnée, entamée trois ans plus tôt depuis le parking du campus, se conclue finalement d'une bien belle manière.

Je suis aussi bien conscient de la chance que j'ai eue de faire partie de l'équipe MeTAH. Vous avez été ma famille d'accueil, et vous m'avez fait me sentir comme chez moi. C'est sans aucun doute avec une grande nostalgie que je me remémorerai tous les moments qui ont fait de mon passage ici une expérience inoubliable : les discussions de tous les jours à midi, les séminaires au vert, les repas d'équipe de fin d'année (solaire ou lunaire) ou encore les verres pris à la piscine ou à Ève. Mais plus que tout, ce sont vous, les membres de cette équipe, qui vont me manquer. Je pense en particulier à mes amis doctorants et post-doctorants : Ben, Catherine, Mohameth-François, Nathalie, Reinaldo, Sébastien L., Sébastien J., et Péricles. Je m'en voudrais aussi de ne pas mentionner Anne, Cédric, Nadine, et Viviane. Merci d'avoir été là, tout simplement.

Cette expérience de la thèse n'aurait pas été la même sans mes activités d'enseignement. Je voudrais donc remercier Alexia pour son accueil plus que chaleureux au sein de l'ENEPS et pour les nombreuses discussions que nous avons eu après les TP de programmation. De la même façon, je souhaiterais remercier François pour son professionnalisme. Je ne sais pas si j'aurais été capable de préparer les TP de Mathématiques sans son aide.

En Chine, on dit souvent que lorsqu'il n'y a plus rien, il y a toujours la famille. Si j'en suis ici aujourd'hui, c'est évidemment grâce à la mienne. C'est pourquoi je voudrais remercier mes parents pour nous avoir ouvert, à ma sœur, mon frère et moi-même, le champ des possibles. Une tendre pensée aussi pour Huihan et François : vous qui me supportez depuis si longtemps, votre petit frère a grandi et il tâchera de moins vous enquiquiner par la suite. Je profite finalement de cette occasion pour vous le dire encore une fois : Papa, Maman, Huihan, François, je vous aime et merci pour tout.

J'ai aussi une pensée toute particulière pour mon père, qui s'est énormément sacrifié pour que sa famille puisse mener la vie qu'elle mène désormais. Je sais que les premières années ont été difficiles, peut-être même trop difficiles à supporter pour un seul homme. J'espère qu'en ce jour tu es fier de moi.

Enfin, mes derniers mots te sont destinés Chao. Je ne sais comment te remercier pour tout le soutien que tu m'as apporté et que tu continues encore de m'apporter. Il y a eu pas mal de hauts, et puis pas mal de bas aussi. Mais tout cela en valait la peine, tu ne trouves pas ? Le moins que l'on puisse dire, c'est que cette année 2016 restera à jamais ancrée dans nos mémoires. De quoi sera fait le futur ? J'ai bien un embryon d'idée (voire même un peu plus que ça) et la seule chose dont je suis sûr, c'est que j'ai hâte d'y être à tes côtés...

Patrick Wang

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLES

1 INTRODUCTION

Contenu du chapitre

1.1 PRÉSENTATION GÉNÉRALE DES TRAVAUX

1.1.1 *Domaine de recherche*

Les travaux de recherche menés au cours de cette thèse se situent dans le cadre de l'utilisation d'Environnements Informatiques pour l'Apprentissage Humain (EIAH) en classe. Ce domaine de recherche étudie comment l'utilisation d'environnements informatiques peut venir en aide (1) à des apprenants pour supporter et structurer leurs activités d'apprentissage et/ou (2) à des enseignants pour supporter et structurer leurs interventions au cours de ces activités.

Au sein du domaine des EIAH, nous nous intéressons au contexte technologique de l'utilisation de tablettes tactiles en classe. En particulier, nous nous intéressons à des situations issues du « *one-to-one Technology-Enhanced Learning* » (Chan et al., 2006) où chaque élève se voit attribué une tablette. Il s'agit alors de tirer profit d'une telle configuration pour créer de nouvelles situations d'apprentissage. Cette pratique rencontre un intérêt fort, comme l'en atteste le développement d'un grand nombre de projets de recherche et d'applications éducatives. Pourtant, peu de travaux de recherche s'intéressent à la mise en œuvre de ces applications en classe et sur le rôle de l'enseignant dans ce type de situation. Cette observation est surprenante puisque la décision quant à l'adoption et l'utilisation d'EIAH en classe revient principalement à l'enseignant (Ifenthaler & Schweinbenz, 2013).

Au sein de ces situations d'apprentissage, nous nous intéressons plus particulièrement aux enseignants et à comment ces-derniers gèrent ces situations. Nous ferons référence à ceci comme l'*orchestration* d'une situation d'apprentissage en classe. Dillenbourg (Dillenbourg, 2013, p. 485) propose la définition suivante : « L'orchestration fait référence à la façon dont un enseignant gère en temps réel des activités se déroulant sur plusieurs plans [c'est-à-dire sur un plan individuel, collectif, et en classe entière] dans un contexte à plusieurs degrés de contrainte ».

Dans cette thèse, nous utiliserons le terme de *technologie d'orchestration* pour faire référence à des logiciels conçus pour aider un enseignant à orchestrer une situation d'apprentissage utilisant des EIAH en classe. Une technologie d'orchestration comprend typiquement des outils permettant de visualiser les actions et productions des élèves lors de la réalisation d'une activité, et d'en modifier son cours si nécessaire.

Il existe un certain nombre de travaux qui se sont conclus par le développement de technologies supportant un enseignant dans l'orchestration d'une situation spécifique et définie par avance. Du fait de leurs spécificités, ces technologies ne sont pas aptes à être directement réutilisées dans des situations différentes.

Au cours de cette thèse, nous nous sommes penchés sur cet aspect avec pour objectif de proposer un framework logiciel pouvant servir de base pour le développement d'une technologie d'orchestration. Un tel framework, moyennant un travail d'instanciation pour une application-élève [1], serait alors en mesure de supporter l'orchestration de scénarios mis en œuvre par cette application-élève.

1.1.2 *Travaux réalisés*

Cette thèse vise à étudier les problèmes conceptuels et techniques que l'on peut rencontrer lors de la construction d'une librairie d'outils logiciels pouvant servir de base pour le développement de technologies d'orchestration.

Pour cela, nous avons dans un premier temps conçu et développé une première version du framework à l'aide d'un exemple *fil rouge*, en l'occurrence une application-élève pour mettre en œuvre sur tablettes des dictées négociées (application que nous avons par ailleurs conçue et développée en parallèle avec l'aide d'enseignants du primaire). Cette première version mettait l'accent sur l'aide à l'enseignant dans le suivi d'une dictée négociée en classe.

Dans un second temps, nous avons continué ce travail en affinant les concepts et les outils logiciels impliqués dans le développement du framework. Ce travail s'est notamment nourri de l'instanciation du framework pour une deuxième application-élève : SimBûchettes. Cette application permet de mettre en œuvre des exercices de mathématiques, et nous l'avons choisie parce qu'elle a été développée par un tiers [2]. Nous pouvions ainsi tester d'instancier le framework pour une application que nous n'avions pas conçue nous-même.

Enfin, nous avons complété ce travail à l'aide d'une instanciation pour une troisième application-élève : Topeka. De la même façon que pour SimBûchettes, nous avons utilisé Topeka pour compléter notre

1. Dans cette thèse, une application-élève correspond à l'application utilisée par les élèves au cours de l'activité d'apprentissage.

2. Plus précisément, par une stagiaire de Master 1 au sein de notre équipe.

travail de conceptualisation et de développement des outils logiciels et parce que cette application a aussi été développée par une personne extérieure à notre équipe de recherche [3].

Les résultats de ces travaux sont un modèle d'orchestration, des modèles d'interface pour supporter l'enseignant dans l'orchestration d'un scénario pédagogique en classe, et le framework logiciel CHAO (qui signifie orCHestration of Activities in classrOoms) qui a pour objectif de servir de base pour le développement de technologies d'orchestration.

Dans le cadre de la thèse, nous avons mené trois travaux à visée expérimentale pour évaluer nos contributions. Pour cela, nous avons utilisé les trois applications-élève mentionnées plus tôt. Ces travaux ont porté sur (1) l'évaluation de l'utilité et de l'efficacité des outils conçus pour supporter un enseignant dans le suivi d'un scénario mis en œuvre par l'application-élève de dictée négociée, (2) l'évaluation de l'utilisabilité et l'efficacité des outils conçus pour pouvoir modifier en temps réel un scénario mis en œuvre par l'application-élève SimBûchettes, et (3) la quantification du travail qu'un informaticien doit fournir pour instancier le framework pour les applications-élève SimBûchettes et Topeka.

1.2 ORGANISATION DU DOCUMENT

Le CHAPITRE 2 présente le domaine de recherche de l'orchestration. Nous y rappelons les définitions formulées par différents travaux de recherche et présentons le modèle $5+3$ de Prieto et collègues (Prieto, Holenko Dlab, et al., 2011). En particulier, nous détaillons les pistes méthodologiques de recherche en orchestration dessinées par ce modèle. Puis, nous expliquons la distinction entre technologie d'orchestration et technologie orchestrable avant d'analyser des principes de conception spécifiés dans la littérature. Enfin, nous présentons des exemples de technologies mettent en œuvre ces principes, et nous concluons ce chapitre en expliquant le positionnement adopté durant cette thèse.

Le CHAPITRE 3 présente le contexte précis dans lequel s'inscrit notre problématique de recherche : l'absence d'un chaînon technologique faisant le lien entre applications à but éducatif sur tablettes et technologies d'orchestration pour l'enseignant. Ce chapitre explicite notre problématique et détaille la méthodologie suivie afin de combler ce manque. Nous donnons aussi une brève description des trois applications-élève que nous avons utilisées au cours de cette thèse pour tester et évaluer le framework : la dictée négociée, SimBûchettes, et Topeka.

Dans le CHAPITRE 4, nous décrivons les contributions de cette thèse, c'est-à-dire le framework CHAO et ses modèles sous-jacents.

3. En l'occurrence, par des ingénieurs de Google.

Dans un premier temps, nous présentons le modèle d'orchestration d'un scénario pédagogique en classe construit au cours de cette thèse. Ce modèle d'orchestration comprend (1) des actions menées par un enseignant pour préparer et gérer un scénario pédagogique, et (2) des informations facilitant la tâche de l'enseignant dans le suivi en temps réel du déroulement de ce scénario. Pour construire ce modèle, nous nous sommes appuyés sur les travaux de Tchounikine qui identifient trois actions d'orchestration distinctes : les actions de *primo-scripting*, les actions de *runtime scripting*, et les actions de *monitoring* (Tchounikine, 2013). Puis, nous présentons les modèles d'interface conçus pour permettre à un enseignant de réaliser ces actions sur une tablette tactile qu'il utiliserait comme support d'orchestration. Enfin, nous détaillons la conception et le développement du framework Chao et son processus d'instanciation.

Le Chapitre 5 présente les travaux à visée expérimentale que nous avons menés pour évaluer nos propositions. Ces évaluations ont porté sur l'utilité, l'utilisabilité, et l'efficacité des outils et interfaces de suivi et de modification de scénario pédagogique en temps réel, et sur le travail d'instanciation du framework Chao. Afin de mieux expliquer ces évaluations, ce chapitre présente également plus en détail les différentes instances du framework.

Enfin, dans le Chapitre 6, nous faisons une synthèse des travaux effectués en précisant le spectre d'application de nos contributions. Nous analysons aussi ces travaux par rapport à ceux identifiés dans le chapitre d'état de l'art, et proposons des perspectives de recherche pouvant s'appuyer sur nos résultats.

2

L'ORCHESTRATION DE SCÉNARIOS UTILISANT DES EIAH EN CLASSE

Contenu du chapitre

2.1 DÉFINITIONS DE L'ORCHESTRATION

Le terme d'orchestration a été utilisé dans un premier temps pour illustrer le rôle d'un enseignant dans sa classe, tel un chef d'orchestre coordonnant le jeu de ses musiciens. Par exemple, Dillenbourg et collègues proposent la définition suivante :

> « *[Orchestration is] the process of productively coordinating supportive interventions across multiple learning activities occuring at multiple social levels.* » (Dillenbourg, Järvelä & Fischer, 2009, p. 12)

Par la suite, Dillenbourg affine cette définition en précisant que l'orchestration d'une situation d'apprentissage en classe nécessite que l'enseignant soit réactif au déroulement de cette situation. Les actions entreprises par l'enseignant le sont donc en temps réel :

> « *Orchestration refers to how a teacher manages in real-time multi-layered activities in a multi-constraints context.* » (Dillenbourg, 2013, p. 485)

Cependant, le rôle d'un enseignant ne se limite pas à la gestion de la situation en classe et comporte aussi la préparation des activités. En tenant compte de cette nouvelle dimension, Kollar et Fischer formulent la définition suivante :

> « *Orchestrating TEL means the process of creating, adapting and enacting a technology-enhanced learning scenario under complex classroom conditions.* » (Kollar & Fischer, 2013, p. 508)

Cette définition étend le champ d'application de l'orchestration aux actions d'un enseignant se déroulant à la fois avant et pendant une activité en classe.

Quelle que soit la définition adoptée, l'orchestration d'une situation d'apprentissage en classe est une tâche complexe pour un enseignant, et l'utilisation d'un EIAH peut contribuer à y ajouter encore un niveau de difficulté. En effet, Dillenbourg et Tchounikine évoquent dans un premier temps les difficultés rencontrées en lien avec le besoin de flexibilité lors de la conception d'un scénario pédagogique utilisant des EIAH (Dillenbourg & Tchounikine, 2007). Dans un second temps, Prieto et collègues (Prieto, Sharma & Dillenbourg, 2015; Prieto, Sharma, Wen, et al., 2015) mettent en évidence des épisodes de charge cognitive élevée de l'enseignant lors d'interactions de régulation en classe entière (en référence à la notion de charge d'orchestration définie par Cuendet et collègues comme « *the effort necessary for the teacher to conduct learning activities* » (Cuendet et al., 2013, p. 558)).

Dans une tentative de formuler une modélisation de l'orchestration, Prieto et collègues (Prieto, Holenko Dlab, et al., 2011) ont construit le framework $5 + 3$ illustré en FIGURE 1.

Ce modèle identifie cinq caractéristiques de l'orchestration et reprend les définitions données plus tôt (les trois éléments restants seront analysés par la suite en SECTION 2.2) :

- *Design/Planning* : ce point fait référence à une phase en amont des activités, durant laquelle un enseignant construit un scénario pédagogique ;
- *Regulation/Management* : ce point fait référence aux actions de régulation d'un enseignant pendant le déroulement des activités ;
- *Awareness/Assessment* : ce point fait référence au besoin de l'enseignant d'estimer le déroulement de l'activité avant d'entreprendre des actions de régulation ;
- *Adaptation/Flexibility/Intervention* : ce point fait référence à la nécessité de concevoir des scénarios adaptables ou d'avoir des technologies flexibles qui peuvent être adaptées selon les interventions de l'enseignant ;
- *Teachers/Other actors* : ce point fait référence au fait que l'enseignant joue un rôle essentiel dans l'orchestration de sa classe ; nous ne sommes donc pas dans une situation où un système

tutoriel intelligent serait en charge de gérer les activités d'un élève.

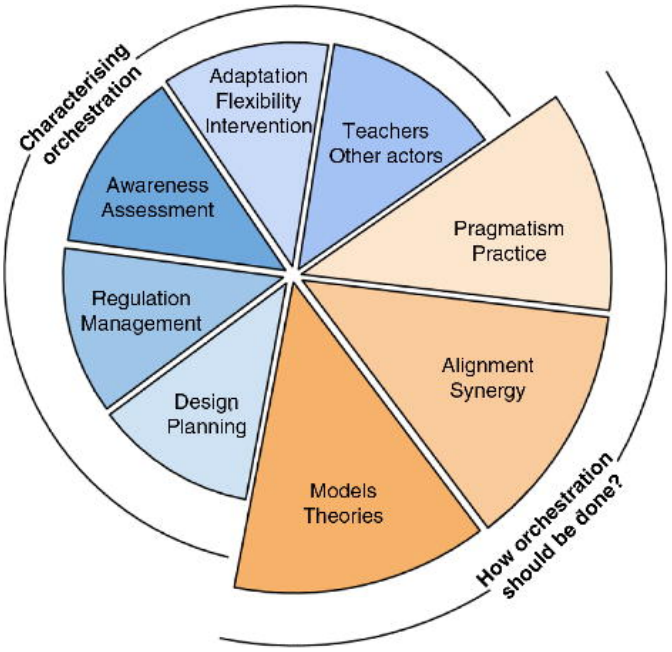


FIGURE 1 – Le modèle d'orchestration $5+3$ de (Prieto, Holenko Dlab, et al., 2011).

Respecter ce modèle et concevoir une technologie d'orchestration est difficile. C'est pourquoi nous allons reprendre et analyser, dans la section suivante, les trois éléments du modèle $5+3$ présentant des pistes méthodologiques de recherche en orchestration.

2.2 MÉTHODOLOGIE DE RECHERCHE EN ORCHESTRATION

Les trois éléments du modèle $5+3$ visent à apporter des pistes méthodologiques pour réaliser des travaux de recherche sur l'orchestration de scénarios pédagogiques en classe.

Par le *pragmatisme*, Prieto et collègues notent la volonté des chercheurs de rendre leurs résultats et travaux accessibles à des enseignants et à leurs classes. Cela signifie de concevoir des technologies destinées à être utilisées dans des conditions *réelles*. Ce point de vue est aussi repris par Chan, qui propose de passer d'une *vision de laboratoire* à une *vision de terrain* (Chan, 2013).

Par la *synergie*, les auteurs mettent en avant l'un des traits les plus souvent mentionnés contribuant à des situations d'apprentissage bien orchestrées. Cette caractéristique stipule que pour tirer le meilleur parti d'une situation d'apprentissage en classe utilisant des EIAH, il est nécessaire que les acteurs et les technologies poursuivent des objectifs complémentaires. Par exemple, l'enseignant et les technolo-

gies peuvent chercher à créer des conditions d'apprentissage structurées et captivantes, tandis que les élèves doivent faire preuve d'intérêt pour ces situations.

Par les *modèles*, Prieto et collègues soulignent la nécessité de construire des modèles homogènes décrivant l'orchestration de situations en classe afin de diriger les futurs travaux de ce domaine. En effet, les auteurs constatent que les travaux sur l'orchestration en classe résultent en la construction de modèles d'orchestration hétérogènes. Cette hétérogénéité provient à la fois des différentes définitions de l'orchestration dans la littérature, mais aussi des différents contextes de mise en œuvre des scénarios. En guise d'illustration, les auteurs référencent : l'orchestration d'un module d'ingénierie pour étudiants de Master en utilisant des étapes d'observation, de réflexion, et d'articulation (Chamberlain et al., 2001) ; l'orchestration de discussions en classe entière au sujet des mathématiques dans l'enseignement primaire et secondaire (Stein et al., 2008) ; et l'orchestration de scénarios d'apprentissage collaboratif (Weinberger et al., 2007).

2.3 TECHNOLOGIES D'ORCHESTRATION

Les définitions de l'orchestration mentionnées en SECTION 2.1 peuvent laisser planer le doute quant à ce qu'est une technologie d'orchestration. Est-ce le système supportant un enseignant dans l'orchestration d'un scénario pédagogique en classe ? Est-ce le système permettant aux élèves de réaliser ce scénario, tout en facilitant le travail d'orchestration de l'enseignant ? Est-ce un mélange des deux ?

Tchounikine propose de clarifier cette ambigüité en distinguant *technologie orchestrable* et *technologie d'orchestration* (Tchounikine, 2013).

Une technologie orchestrable est une technologie permettant de mettre en œuvre un scénario pédagogique, et qui offre la possibilité à l'enseignant d'en modifier ou d'adapter certains de ses paramètres en cours d'utilisation (par exemple, des paramètres de rétroaction ou des paramètres décrivant le scénario). Dans la suite de ce document, nous utiliserons aussi le terme d'*application-élève* pour nous référer à l'application sur tablette tactile utilisée par les élèves.

Une technologie d'orchestration est une technologie qui réalise ou permet à un enseignant de réaliser des actions d'orchestration. C'est-à-dire, par exemple, une technologie supportant un enseignant dans la gestion (c'est-à-dire la définition, la modification, et le suivi) d'un scénario en classe.

La conception d'une technologie orchestrable ou d'une technologie d'orchestration est difficile. Cette difficulté provient en partie du fait que les technologies doivent être flexibles pour faire face à un contexte aussi imprévisible qu'une classe (Roschelle, Dimitriadis & Hoppe, 2013). Pour guider ce travail, un certain nombre de travaux

ont conduit à énoncer des principes de conception que nous allons détailler dans la section qui suit.

2.4 PRINCIPES DE CONCEPTION

Les principes que nous allons mentionner dans cette section font référence à la conception d'une technologie d'orchestration ou d'une technologie orchestrable. De prime abord, nous pouvons penser que les principes de conception d'une technologie orchestrable ne sont pas pertinents dans le cadre de cette thèse. Cependant, une technologie d'orchestration ne peut pas supporter un enseignant dans l'orchestration d'une situation en classe si les élèves n'utilisent pas une technologie orchestrable. C'est pourquoi nous proposons de décrire tout de même ces principes, en gardant en tête l'idée qu'une technologie orchestrable peut aussi influencer la conception d'une technologie d'orchestration.

Nous avons sélectionné cinq travaux de recherche énonçant des listes de principes de conception à suivre pour le développement de technologies d'orchestration et de technologies orchestrables pour un usage en classe :

— Dillenbourg & Jermann, 2010 ;
— Dillenbourg, 2013 ;
— Prieto, Holenko Dlab, et al., 2011 ;
— Prieto, 2015 ;
— Kharrufa et al., 2013.

Nous développons ci-dessous une analyse faisant le lien entre les principes de conception mentionnés dans ces articles et les cinq éléments du modèle 5 + 3 décrit en FIGURE 1.

2.4.1 *Principes en lien avec les acteurs de l'orchestration*

Les définitions mentionnées en SECTION 2.1 impliquent que l'enseignant reste l'acteur principal de l'orchestration. C'est pourquoi Dillenbourg et Jermann utilisent les termes de *leadership* et *contrôle* (Dillenbourg & Jermann, 2010).

Que ce soit avec ou sans technologie d'orchestration, le leadership fait référence au fait que l'enseignant est celui qui dirige l'activité de ses élèves, qui interagit directement avec eux, et qui leur fournit des conseils en cas de besoin.

Le principe de contrôle précise que l'enseignant doit garder le contrôle sur la technologie d'orchestration qu'il utilise. Cela signifie que, si la technologie d'orchestration prend des décisions automatiques (comme par exemple donner un nouvel exercice à un élève), l'enseignant doit être en mesure d'inhiber ou de revenir sur ces décisions s'il souhaite en prendre d'autres.

Enfin, Kharrufa et collègues s'intéressent aussi aux élèves comme acteurs de l'orchestration. Pour cela, ils proposent de faciliter la collaboration entre les élèves (Kharrufa et al., 2013), ce qui se rapproche d'un principe de conception de technologie orchestrable. Ce dernier point ne faisant pas l'objet de la thèse, nous l'évoquons ici uniquement à titre indicatif.

2.4.2 *Principes en lien avec la mise en œuvre d'un scénario pédagogique*

Une première recommandation est de tenir compte des machines utilisées et du contexte d'utilisation. Ce principe implique donc d'analyser la pertinence des appareils utilisés ou des scénarios pédagogiques mis en œuvre dans le cadre d'une activité se déroulant en classe. Par exemple, préparer une activité collaborative sur ordinateurs fixes durant laquelle les élèves peuvent être amenés à se déplacer et former des groupes n'est pas judicieux.

Une deuxième recommandation est de concevoir une technologie structurant les activités des élèves de façon à guider les élèves dans la réalisation du scénario, soit le principe de *guidage* (Dillenbourg & Jermann, 2010; Kharrufa et al., 2013). Dans le cadre d'activités collaboratives, cette recommandation est particulièrement importante puisque la qualité des interactions entre les élèves influe directement sur la qualité des apprentissages (Dillenbourg, Järvelä & Fischer, 2009).

Enfin, une dernière recommandation est de concevoir des scénarios pédagogiques dont les productions peuvent être réutilisés lors de séances non-informatisées (Kharrufa et al., 2013). En effet, l'utilisation de technologies en classe reste encore minoritaire. Il s'agit donc d'intégrer ces scénarios pédagogiques utilisant des EIAH dans des modules plus conséquents, et de profiter des travaux réalisés par les élèves au cours de ces scénarios dans les séances suivantes (Dillenbourg & Jermann, 2010).

2.4.3 *Principes en lien avec les actions de régulation*

Les définitions de l'orchestration mentionnent toutes les interventions d'un enseignant pour gérer et réguler le déroulement d'un scénario en classe. C'est pourquoi Dillenbourg met l'accent sur le fait qu'orchestrer est une action physique : c'est le principe de *physicalité* (Dillenbourg, 2013). Dillenbourg propose alors d'étudier toutes les dimensions physiques liées à la conception d'une technologie d'orchestration. Cela peut comprendre la taille des machines utilisées (dans ce cas, fournir une tablette aux enseignants en charge de l'orchestration peut paraître intéressant pour sa mobilité) ou encore la disposition de la classe (si, par exemple, l'utilisation d'un tableau blanc interactif est envisagée).

Pour faciliter les actions de régulation d'un enseignant, les auteurs des cinq travaux de recherche analysés s'accordent pour énoncer le principe de *linéarité* qui s'applique à la conception d'une technologie orchestrable. Ce principe propose que le scénario pédagogique mis en œuvre doit être linéaire et divisé en plusieurs phases, dans lequel les élèves réalisent tous en même temps une tâche identique ou similaire. L'enseignant peut alors mieux se repérer dans le déroulement du scénario pédagogique et peut profiter des intervalles de temps entre deux phases pour adapter la suite du scénario ou fournir des indications avant de démarrer une nouvelle phase.

D'autre part, faire en sorte que tous les élèves travaillent de façon simultanée sur une même tâche réduit aussi la charge d'orchestration d'un enseignant. Cela peut aussi se révéler particulièrement pratique si l'enseignant décide, en cours de route, d'étaler un scénario sur plusieurs séances.

2.4.4 *Principes en lien avec le suivi d'un scénario*

Une des tâches de l'enseignant pendant le déroulement d'un scénario est de s'assurer que celui-ci se développe sans encombre. C'est pourquoi les cinq travaux de recherche mettent tous l'accent sur la nécessité de fournir des moyens à l'enseignant de visualiser l'activité de ses élèves, en mentionnant le principe de *visualisation*. Un enseignant peut alors mieux cibler ses actions de régulation en synthétisant les informations fournies par les technologies d'orchestration.

Bien que les articles concordent pour affirmer ce besoin d'évaluer le déroulement des activités en classe, la réalisation de ce service et les objectifs recherchés diffèrent.

Une première réalisation possible est de ne fournir ces informations qu'à l'enseignant. C'est ce que Kharrufa et collègues appellent *l'espace privé de l'enseignant* (Kharrufa et al., 2013). Cette possibilité peut être justifiée, par exemple, par le fait que la pertinence des actions de l'enseignant ont des conséquences quant aux apprentissages des élèves (Onrubia & Engel, 2011). Cependant, une nouvelle tendance est de rendre accessible ces informations à tous les participants, c'est-à-dire aussi bien à l'enseignant qu'à ses élèves. De cette façon, un élève est capable de comparer son avancement avec celui de ses camarades, et les groupes peuvent évaluer la participation de chacun de leurs membres.

En respectant ce principe de conception, Prieto (Prieto, 2015) évoque aussi deux caractéristiques en lien avec le suivi d'un scénario. La première caractéristique est la possibilité de sauvegarder et de recréer un historique des actions des élèves. La seconde caractéristique est la possibilité, grâce à cet historique, de synthétiser plus rapidement les informations pour préparer une éventuelle phase de correction des exercices.

2.4.5 *Principes en lien avec la flexibilité des scénarios et technologies*

Les définitions de l'orchestration évoquent le besoin de gérer des activités en classe pendant leur déroulement. Le scénario pédagogique ou les technologies mises en œuvre doivent donc être suffisamment flexibles pour tenir compte des événements justifiant d'effectuer des modifications. À ce titre, les principes de conception liés à la flexibilité affectent trois dimensions.

La première dimension est liée à la flexibilité du scénario pédagogique. Dillenbourg et Tchounikine traitent de cette dimension (Dillenbourg & Tchounikine, 2007). Une manière de concevoir des scénarios pédagogiques flexibles est d'identifier les contraintes intrinsèques et extrinsèques au scénario pédagogique. Les contraintes intrinsèques sont liées aux principes du scénario pédagogique. Les contraintes extrinsèques sont introduites par des facteurs externes (par exemple, technologiques avec la décision d'utiliser une tablette pour sa mobilité plutôt qu'un ordinateur, ou institutionnels pour respecter le règlement d'une école).

La seconde dimension résulte directement de la première, et concerne la flexibilité des technologies mises en œuvre pour réaliser ces scénarios pédagogiques. La difficulté ici repose sur le besoin de concevoir une technologie dont le comportement et les paramètres puissent être adaptés à la volée, sans pour autant dénaturer le scénario pédagogique en cours (Tchounikine, 2008).

La troisième dimension est liée à l'une des ressources les plus critiques pour un enseignant en classe : le temps (Dillenbourg, 2013). Est-il envisageable de démarrer une nouvelle activité sur le dernier quart d'heure ? Comment l'enseignant peut-il raccourcir une tâche en cours pour que cette dernière ne dépasse pas le temps imparti d'une séance en classe ? Dans ce cas, le principe de flexibilité propose que la conception d'un scénario pédagogique ou d'une technologie d'orchestration permette de gérer les contraintes temporelles liées à une séance en classe. Par exemple, en offrant la possibilité de reprendre une activité d'une séance à l'autre ou de modifier certains paramètres d'une tâche pour la raccourcir.

2.4.6 *Principe de minimalisme*

Un principe de conception est mentionné par Dillenbourg et peut s'appliquer à tous les éléments du modèle d'orchestration $5 + 3$: le principe de *minimalisme* (Dillenbourg, 2013). Il qualifie d'ailleurs ce principe de « méta-principe » (Dillenbourg, 2013, p. 491). Pour que les technologies d'orchestration soient utilisées et acceptées par les enseignants, il faut qu'elles restent le plus simple possible. Par cette caractéristique, Dillenbourg suggère que les technologies d'orchestration ont aussi pour objectif de minimiser la charge d'orchestration des

enseignants. Cela passe par essayer de réduire au plus la complexité d'utilisation de ces technologies, sans pour autant affecter la qualité des outils et informations proposés.

2.5 EXEMPLES DE TECHNOLOGIES

Afin d'illustrer les principes de conception listés en SECTION 2.4, nous proposons dans cette section d'analyser certains travaux mettant en œuvre des technologies orchestrables et des technologies d'orchestration.

2.5.1 *Exemples de technologies orchestrables*

TinkerLamp (Dillenbourg, Zufferey, et al., 2011; Do-Lenh et al., 2012) est un exemple de technologie orchestrable. Utilisée dans des cours de logistique, cette technologie met en œuvre un projecteur, une caméra, et une feuille de papier au contenu spécifique (une TinkerSheet) qui permettent aux élèves d'organiser des entrepôts en plaçant des jetons ou des étagères miniatures sur la TinkerSheet. Le travail des élèves est filmé et projeté sur un grand écran pour que l'enseignant puisse suivre le déroulement de l'activité. Cette technologie offre aussi à l'enseignant la flexibilité nécessaire pour adapter son comportement en utilisant des TinkerKeys : des cartes sur lesquelles sont imprimés des codes QR et qui peuvent contrôler l'activité des élèves. Par exemple, en utilisant une carte spécifique, un enseignant peut mettre toute la classe en pause ou autoriser un élève à évaluer l'organisation de son entrepôt en simulant l'activité du personnel remplissant les étagères. Cette technologie reprend plusieurs principes de conception, dont les principes de visualisation, de contrôle, de physicalité, de flexibilité, et aussi de minimalisme (avec l'utilisation de cartes).

CK3 (Fong, Cober, Moher, et al., 2015) est une application utilisée dans le cadre de l'apprentissage par démarche d'investigation, et consiste en un environnement de partage de notes. Les élèves produisent ces notes avec l'aide de tablettes puis ces notes sont affichées sur un tableau blanc interactif (Fong, Cober, Madeira, et al., 2013). En prenant connaissance des notes rédigées par ses élèves, l'enseignant peut intervenir pour introduire de nouveaux concepts, recentrer le travail de ses élèves, et animer des débats en confrontant deux concepts opposés. La gestion d'une activité se fait donc à la fois oralement et avec l'aide du tableau en déplaçant et regroupant des notes. Dans cet exemple, les principes de visualisation, de contrôle, et de flexibilité ont été suivi. Nous pouvons cependant noter que dans ce cas, l'enseignant reste l'acteur principal de l'orchestration mais ne contrôle pas entièrement la situation puisque des outils de gestion lui font défaut (comme mettre en pause le travail des élèves).

2.5.2 *Exemples de technologies d'orchestration*

SceDer (Niramitranon, 2009) est un système conçu pour supporter les enseignants dans l'orchestration d'un scénario d'apprentissage collaboratif en classe. Il s'agit donc d'une technologie d'orchestration, qui offre à l'enseignant la possibilité d'éditer un scénario pédagogique (avec l'outil SceDer Authoring) et de suivre le déroulement du scénario pédagogique grâce à l'outil GS-SceDer, qui reprend l'application de création et de partage de notes Group Scribbles (Roschelle, Tatar, et al., 2007). Bien que les résultats d'utilisabilité et d'utilité de GS-SceDer semblent positifs, les enseignants impliqués dans les expérimentations évoquent le besoin de pouvoir mettre en pause le travail des élèves et soulignent le désagrément de devoir toujours retourner sur leurs ordinateurs pour interagir avec le scénario pédagogique (Niramitranon et al., 2010). Les principes de conception suivis dans cet exemple sont les principes de flexibilité et de visualisation.

Un exemple de technologie d'orchestration évitant ces défauts est MTClassroom (Martinez-Maldonado, Dimitriadis, et al., 2013). Des élèves travaillent en groupe sur des tables tactiles, et l'enseignant possède son propre écran de contrôle sur sa tablette : MTDashboard. Grâce à MTDashboard, l'enseignant possède les outils pour interagir avec les tables tactiles (pour les mettre en pause ou pour envoyer des messages) et peut prendre connaissance des informations quant à la participation de chaque élève d'un groupe dans la réalisation du travail. De plus, l'utilisation d'une tablette offre à l'enseignant la mobilité nécessaire pour se déplacer dans sa classe. Les auteurs ont conçu ce système en suivant quatre principes : l'enseignant est l'acteur principal de l'orchestration (en lien avec le principe de contrôle), le système supporte la synchronisation des activités (en lien avec le principe de linéarité), le système supporte la gestion et le suivi des activités de la classe (en lien avec le principe de suivi d'un scénario), et le système produit des indicateurs informant de la progression des élèves (en lien avec le principe de visualisation).

2.6 SYNTHÈSE ET POSITIONNEMENT

Les définitions de l'orchestration données en SECTION 2.1 ont montré qu'il existe entre les chercheurs des visions légèrement différentes pour caractériser les actions d'orchestration d'un enseignant. Le modèle $5+3$ présenté constitue une tentative de conceptualisation des travaux de recherche en orchestration. Ce modèle, au cours d'un travail de validation, a d'ailleurs été approuvé par un panel d'experts du domaine (Prieto, Dimitriadis, et al., 2015). Notre positionnement se rapproche de la caractérisation de l'orchestration par le modèle $5+3$ et de la définition énoncée par Kollar et Fischer et qui prend en

compte à la fois la création et la gestion d'un scénario pédagogique en classe (Kollar & Fischer, 2013).

En analysant SceDer et MTClassroom, nous nous apercevons que ces technologies d'orchestration ont été construites pour supporter un enseignant dans l'orchestration d'un scénario mis en œuvre par une application-élève spécifique (d'un côté Group Scribbles et de l'autre Collaborative Concept Mapping Application qui est l'application installée sur les tables tactiles (Martinez-Maldonado, Kay & Yacef, 2010)). Ces travaux sont assez proches de notre vision des technologies d'orchestration. Cependant, nous soulevons la question de la réutilisation de ces technologies pour supporter des enseignants dans l'orchestration d'autres applications-élève. En effet, les auteurs de ces deux travaux n'abordent pas spécifiquement cette question, ce qui peut laisser supposer que ces technologies ne sont pas facilement réutilisables dans d'autres situations.

Les cinq éléments du modèle $5 + 3$ mettent en avant les caractéristiques importantes de l'orchestration. Ces caractéristiques sont souvent bien présentes dans les travaux de recherche mentionnés précédemment. Pourtant, nous pouvons observer l'absence d'un travail de modélisation permettant de concevoir et développer des technologies d'orchestration à un niveau abstrait. En effet, bien que SceDer propose un langage de modélisation destiné à l'édition d'un scénario pédagogique, ce langage ne permet pas de modéliser l'orchestration dans son ensemble. D'autre part, MTClassroom et MTDashboard ne sont que les résultats d'un travail de développement fondés sur certains principes de conception identifiés dans la littérature.

Ce travail de thèse vise à combler ce manque de modèle pour le développement de technologies d'orchestration. Pour cela, nous décidons de nous intéresser plus particulièrement à l'orchestration de scénarios pédagogiques en classe où les élèves utilisent des tablettes.

3

PROBLÉMATIQUE ET MÉTHODOLOGIE

Contenu du chapitre

3.1 CONTEXTE

La quantité de travaux de recherche impliquant l'utilisation de terminaux mobiles en classe dénote un intérêt pour l'introduction de ce type d'appareil dans un milieu institutionnel. Cependant, ces travaux ne traitent souvent pas des problématiques de mise en œuvre et de suivi de ce nouveau type d'activité par les enseignants. Nous pouvons citer par exemple les travaux de (Alvarez, Alarcon & Nussbaum, 2011; Boticki, Looi & Wong, 2011; C. Cheong, Bruno & F. Cheong, 2012; Echeverria et al., 2011; Jahnke et al., 2015; Wong & Looi, 2011; Zurita & Nussbaum, 2004).

De façon similaire, les technologies d'orchestration mentionnées en SECTION 2.5.2 ont été conçues et développées pour supporter l'orchestration d'une situation spécifique, ce qui peut constituer un obstacle à leur réutilisation dans des contextes différents.

Bien que la littérature fasse état de principes de conception à suivre (recensés en SECTION 2.4), le chaînon technologique faisant le lien entre les applications mobiles éducatives et les systèmes d'orchestration est manquant. Plusieurs raisons peuvent expliquer l'absence de modèles et d'outils génériques. D'une part, il est difficile de concevoir des outils génériques compte-tenu de la diversité des pratiques et des attentes de chaque enseignant. D'autre part, le fait que ces outils soient génériques implique un travail de conception et de programmation spécifique pour chaque application-élève.

Dans cette thèse, nous nous sommes intéressés à la conception de modèles et d'outils informatiques génériques dont les objectifs sont de faciliter le développement d'une application d'orchestration (technologie d'orchestration) pour venir en aide à un enseignant dans l'orchestration d'un scénario pédagogique utilisant des tablettes en classe

(ce scénario pédagogique étant mise en œuvre par une application-élève). Pour cela, et plutôt que de construire une application d'orchestration générique, l'approche adoptée est de concevoir et développer un framework, c'est-à-dire un ensemble de composants logiciels fournissant des éléments de base pour le développement d'une technologie d'orchestration.

3.2 PROBLÉMATIQUE

La problématique qui structure ce travail de thèse est la suivante :

> « Est-il possible de concevoir un framework logiciel qui soit facile à instancier et qui, une fois instancié, supporte un enseignant dans l'orchestration d'un scénario en classe utilisant des tablettes ? »

D'un point de vue conceptuel, deux questions découlent de cette problématique. D'une part, comment modéliser l'orchestration de façon à expliciter les actions qu'un enseignant réalise lorsqu'il prépare ou modifie une activité en classe et sur quelles informations se base-t-il pour modifier cette activité ? D'autre part, et à partir du modèle d'orchestration construit, comment concevoir des modèles d'interface utilisateur qui soient simples d'utilisation et qui, une fois instanciés, peuvent supporter un enseignant dans l'orchestration d'un scénario en classe ?

D'un point de vue technique, il s'agit de concevoir un framework logiciel qui implémente une suite d'outils d'orchestration génériques. Ce framework doit présenter les deux caractéristiques suivantes : (1) il doit être facile à instancier par l'informaticien qui souhaite développer une technologie d'orchestration, et (2) les outils et informations d'orchestration proposés doivent supporter de façon satisfaisante un enseignant.

3.3 MÉTHODOLOGIE

Une première décision méthodologique que nous avons prise est de doter l'enseignant en charge de l'orchestration d'une tablette. Ce choix est en accord avec deux principes de conception : le principe de physicalité avec la mobilité proposée par les tablettes (Dillenbourg, 2013), et le principe d'espace privé pour la gestion des activités en classe (Kharrufa et al., 2013). Les photos présentées en FIGURE 2 ont été prises lors d'une expérimentation que nous avons menée dans une classe, et illustre bien l'intérêt pour un enseignant d'utiliser une tablette. Dans cette figure, nous pouvons voir que l'enseignante est en mesure de se déplacer dans sa classe tout en gardant un œil sur sa tablette l'informant du déroulement des activités.

Nous avons ensuite conduit un travail analytique pour construire une modélisation de l'orchestration d'un scénario pédagogique en

FIGURE 2 – À gauche, l'enseignante peut suivre le déroulement de l'activité grâce à l'espace privé que représente sa tablette. À droite, la tablette lui permet de se déplacer et de venir en aide à ses élèves.

classe. Pour concevoir ce modèle, nous nous sommes appuyés sur les résultats de deux travaux de recherche. Le premier résultat est un ensemble de notions génériques permettant la modélisation de nombreux scénarios pédagogiques (Kobbe et al., 2007). Le second résultat concerne la modélisation de l'orchestration en termes d'actions de *primo-scripting* (c'est-à-dire l'édition *a priori* d'un scénario pédagogique), *runtime scripting* (c'est-à-dire la modification de ce scénario durant son déroulement), et *monitoring* (c'est-à-dire le suivi du déroulement du scénario) (Tchounikine, 2013).

Puis, nous avons construit des modèles d'interface utilisateur qui, une fois instanciés, présentent des caractéristiques de simplicité d'utilisation. En ce qui concerne les écrans de construction et de modification d'un scénario pédagogique, nous avons basé nos travaux sur des résultats précédents suggérant que des interfaces utilisateurs en forme de table sont utilisables et efficaces pour l'édition d'un scénario pédagogique (Sobreira & Tchounikine, 2015). Pour les interfaces de suivi des activités, nous avons travaillé avec des enseignants du primaire. Pour cela, nous avons conçu des prototypes et avons recueilli leurs impressions quant à l'utilisabilité et l'efficacité des écrans proposés. Ce processus, itéré à plusieurs reprises, nous a permis d'obtenir des informations précises par rapport aux besoins des enseignants.

L'ensemble de ces travaux préliminaires nous a guidé lors du développement du framework afin qu'il présente les deux caractéristiques mentionnées en SECTION 3.2 (soit la facilité d'instanciation et l'implémentation d'outils supportant de façon satisfaisante un enseignant).

Nous avons évalué le framework sur ces deux axes en menant des expérimentations dans plusieurs classes. Nous avons pour cela construit une infrastructure informatique et réseau réunissant les composants nécessaires pour assurer le fonctionnement du framework. Cette infrastructure, composée d'un routeur et de différents serveurs, est décrite en SECTION 4.4.1. Elle permet notamment d'être indépendant du matériel informatique présent dans les écoles.

Pour mener ces travaux à visée expérimentale, nous avons aussi utilisé trois applications-élèves sur tablettes en guise de technologies orchestrables. Nous allons tout d'abord expliciter les raisons qui nous ont amenés à choisir ces trois applications, et les présentons plus en détail dans la SECTION 3.4.

La première application-élève permet de réaliser une dictée négociée. La dictée négociée est un exercice pédagogiquement intéressant, mais difficile à mettre en œuvre et à gérer en classe. Nous avons utilisé cet exercice pour concevoir de façon itérative les interfaces utilisateurs de suivi d'un scénario pédagogique. Suite à cela, nous avons développé une application de dictée négociée, instancié le framework pour supporter l'orchestration de cet exercice, et mené des expérimentations pour évaluer l'utilisabilité et l'efficacité des interfaces conçues.

La seconde application-élève que nous avons utilisée s'appelle SimBûchettes. Cette application a été conçue par une personne tierce de notre équipe (en stage de Master) dans le cadre d'un autre projet de recherche, et permet à des élèves de primaire de réaliser des exercices de numération. Ces exercices présentent des paramètres de rétroaction qui sont modifiables par l'enseignant en fonction de l'activité des élèves. Nous avons choisi cette application pour deux raisons. D'une part pour évaluer les outils de régulation proposés par le framework. D'autre part pour évaluer (1) le travail d'instanciation du framework avec une application que nous n'avions pas conçus nous-même, et (2) le travail d'adaptation d'une application-élève que nous n'avions pas conçue pour rendre cette dernière orchestrable.

La troisième application-élève que nous avons utilisée s'appelle Topeka [1]. Cette application propose à son utilisateur de répondre à des quiz thématiques. Nous avons choisi cette application parce qu'elle a été développée en dehors de notre équipe de recherche (à savoir par des ingénieurs de Google) et parce que son code source est accessible librement sur GitHub [2]. Cette application n'a pas été testée dans le cadre d'expérimentations en classe et a juste servi pour évaluer le côté logiciel de nos contributions. En instanciant notre framework pour l'orchestration de Topeka et en modifiant le comportement de cette application-élève pour permettre son orchestration, nous avons complété les éléments de réponse quant à la question de la facilité d'instanciation du framework et des travaux à mener du côté de l'application-élève. En effet, nous souhaitions évaluer ce critère avec deux applications développées en dehors du cadre de cette thèse. SimBûchettes ayant été conçue au sein de notre équipe de recherche, nous avons jugé utile d'effectuer ce travail d'instanciation avec une application dont le développement nous est complètement externe.

1. https://play.google.com/store/apps/details?id=com.chromecordova.Topeka
2. https://github.com/googlesamples/android-topeka

Dans la suite de ce document, nous utilisons les termes CHAO-Dictée négociée, CHAO-SimBûchettes, et CHAO-Topeka pour nous référer aux instances du framework CHAO pour l'orchestration des scénarios mis en œuvre par chacune de ces applications-élève.

3.4 APPLICATIONS-ÉLÈVE POUR L'INSTANCIATION DU FRAMEWORK

Les trois applications-élève que nous avons utilisées au cours de cette thèse sont : l'application de dictée négociée, l'application SimBûchettes, et l'application Topeka. Cette thèse n'étant pas centré sur le travail des élèves, nous nous contentons de donner une description rapide de chacune de ces applications-élève. Nous y décrivons le déroulement de chaque exercice mis en œuvre par ces applications, ce que font les élèves durant ces exercices en interagissant avec leur application-élève, et ce que les enseignants sont amenés à faire lorsqu'ils préparent et orchestrent une situation fondée sur ces exercices.

3.4.1 *La dictée négociée*

La dictée négociée [3] est habituellement un exercice de type papier-crayon se déroulant en trois phases :

1. Une phase individuelle, similaire à une dictée classique où l'enseignant dicte un texte et les élèves le retranscrivent sur papier ;
2. Une phase collective où les élèves, par groupe, doivent retravailler le même texte et s'accorder pour produire une réponse commune ;
3. Une phase d'institutionnalisation durant laquelle l'enseignant reprend en classe entière les productions de chaque groupe et les corrige.

Pour les besoins de cette thèse, nous avons développé une application sur tablette permettant aux élèves de réaliser une dictée négociée. À ce titre, cette application-élève permet de mettre en œuvre un seul scénario pédagogique possible, qui comprend la réalisation de la phase individuelle et de la phase collective.

Durant la phase individuelle, les élèves profitent de l'utilisation de tablettes pour écouter des pistes audio contenant la dictée et pour la saisir à l'écran (voir la FIGURE 3).

La phase collective se déroule en deux étapes. Un premier écran propose aux élèves de considérer des mots spécifiques de la dictée (dans la FIGURE 4, représentés par les mots en rouge de la zone 1). En cliquant sur l'un de ces mots, l'interface de vote (zone 2 de la

3. Des descriptions plus détaillées de la dictée négociée sont données par (Cellier, 2004; Cher, 2005; Lachaud-Beauchene, 2006)

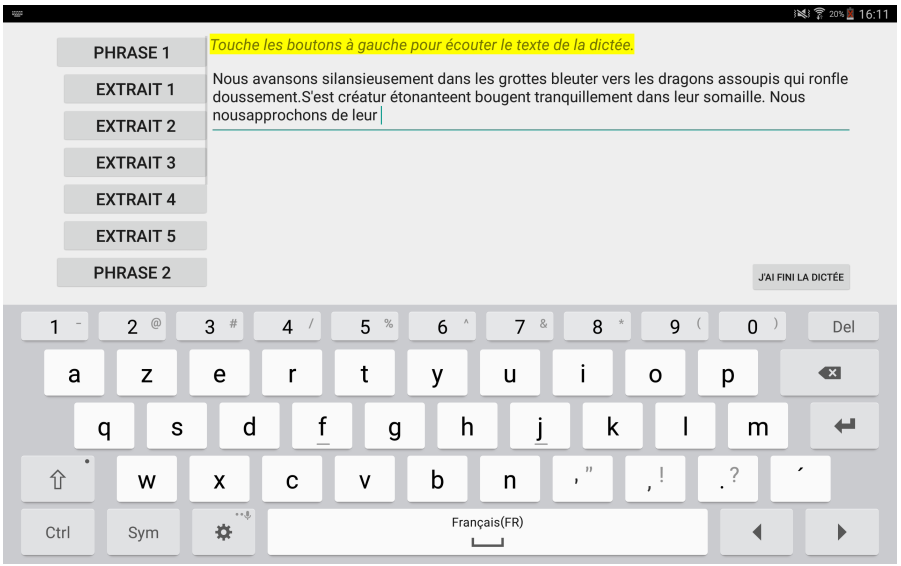


FIGURE 3 – En écoutant les différentes pistes audio situées sur la gauche de l'écran, un élève tape le texte de la dictée durant la phase individuelle de la dictée négociée.

FIGURE 4) met à jour son contenu et propose aux élèves de voter pour l'orthographe qu'ils pensent être correcte ou d'en proposer une autre.

Après ce vote, un deuxième écran demande aux élèves de justifier leurs choix en leur proposant de compléter un texte à trou (voir la FIGURE 5). La production d'une justification passe par une étape de discussion (ou de négociation) durant laquelle les élèves ayant voté pour une même orthographe doivent se mettre d'accord afin de proposer une justification commune. Lorsque les justifications sont validées, les élèves continuent leur travail en votant pour le mot suivant, et ainsi de suite.

L'enseignant en charge de l'orchestration de cet exercice se retrouve confronté à plusieurs difficultés de gestion. Durant la phase individuelle, il s'agit principalement de s'assurer que les élèves saisissent le texte de la dictée et le relisent. Une fois la phase individuelle terminée, l'enseignant doit constituer les groupes et spécifier quels sont les mots à négocier avant de démarrer la phase collective. Au cours de la phase collective, l'enseignant doit suivre le déroulement de l'exercice, intervenir auprès des groupes pour leurs prodiguer des conseils, et éventuellement modifier la composition de certains groupes si nécessaire. Dans cet exemple, il s'agit essentiellement de supporter un enseignant dans ses actions d'édition de scénario pédagogique (constitution des groupes et *runtime scripting*) et de suivi des activités (*monitoring*).

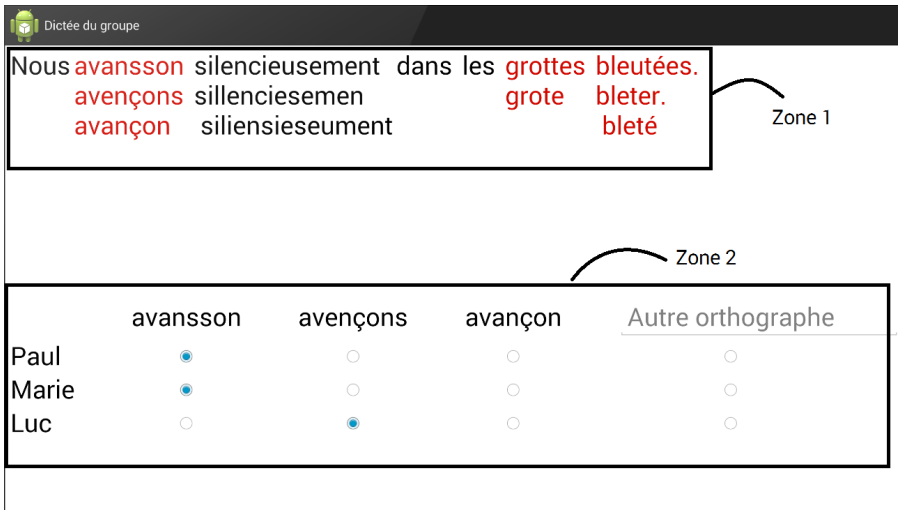


FIGURE 4 – Les trois élèves d'un même groupe retrouvent les différentes orthographes qu'ils ont produites au cours de la phase individuelle et doivent voter pour celle qu'ils jugent être la correcte.

FIGURE 5 – L'écran de justification des votes de la phase collective de la dictée négociée.

3.4.2 *SimBûchettes*

SimBûchettes est une application pour tablette développée au sein de notre équipe de recherche. Elle permet à des élèves de faire des exercices de mathématiques sur la thématique de la numération à l'école primaire. Un scénario pédagogique mis en œuvre par l'application SimBûchettes correspond à une succession d'exercices différents que doivent réaliser les élèves.

Lors de l'utilisation de cette application, les élèves se voient demander de faire des exercices mettant en jeu leurs connaissances sur la numération décimale, et en particulier la mise en pratique du prin-

cipe de position et du principe décimal. Tempier donne les définitions suivantes (Tempier, 2016) :

— Le principe de position stipule que la place d'un chiffre dans un nombre correspond à l'ordre de grandeur de ce chiffre (en référence aux unités, dizaines, centaines, etc...) ;
— Le principe décimal stipule qu'un élément d'un ordre de grandeur est égal à dix éléments d'ordre de grandeur immédiatement inférieur (par exemple, une dizaine est égale à dix unités).

La FIGURE 6 illustre un exemple d'exercice. L'écran propose aux élèves de représenter des nombres à l'aide de bûchettes qui peuvent être déplacées dans des boîtes représentant les unités, dizaines, et centaines. Pour ce faire, l'élève glisse et dépose des éléments de la zone 1 vers la zone 2. La zone 3 correspond à une zone de groupement. Cette zone permet aux élèves de créer des groupements de dix (respectivement des groupements de cent) bûchettes à l'aide de bûchettes (respectivement à l'aide de groupements de dix bûchettes). L'opération inverse est aussi possible, c'est-à-dire de défaire des groupements de dix ou cent bûchettes, pour en retirer respectivement dix unités ou dix groupements de dix bûchettes.

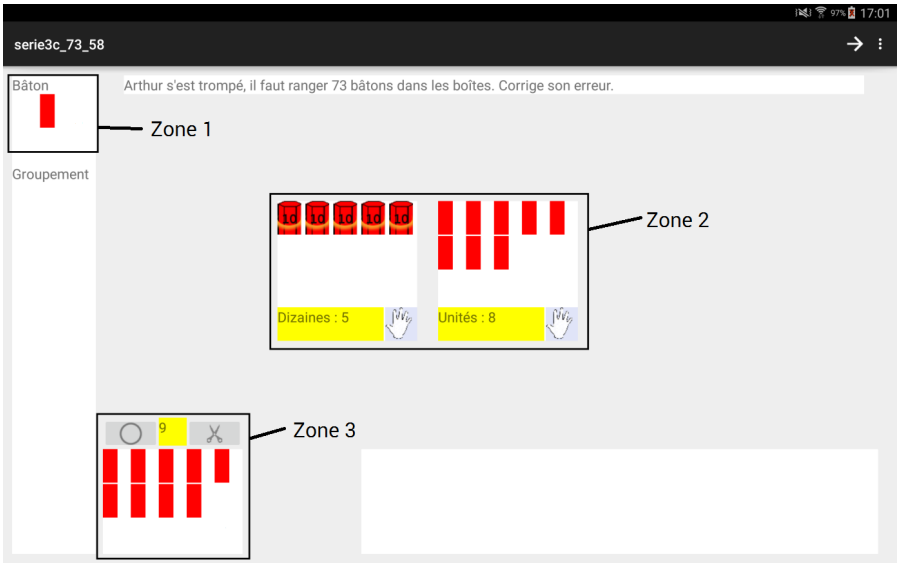


FIGURE 6 – Un exemple d'exercice avec SimBûchettes.

Pour guider le travail des élèves, des rétroactions sont implémentées en lien avec les principes mentionnés précédemment. Ces règles de rétroaction sont ajustables par l'enseignant. Par exemple, des règles de rétroaction peuvent permettre ou empêcher les élèves de faire des actions ne respectant pas les principes de position et décimal (par exemple, déposer 10 bûchettes dans la boîte des unités).

Le travail d'orchestration de l'enseignant consiste dans un premier temps à préparer une succession d'exercices pour chaque élève (*primo-scripting*). Puis, si besoin, cet enchaînement peut être modifié

en cours de réalisation en proposant des exercices mieux adaptés aux connaissances des élèves ou en ajustant les règles de rétroaction pour l'exercice en cours (*runtime scripting*). Ici, ce sont donc essentiellement des actions d'édition de scénario pédagogique, de suivi du scénario, et de régulation des activités des élèves qu'il s'agit de supporter.

3.4.3 *Topeka*

Topeka est une application Android développée par Google. Elle propose à son utilisateur de répondre à des quiz en lien avec huit thématiques différentes. De façon similaire à l'application SimBûchettes, un scénario pédagogique mis en œuvre par Topeka consiste à proposer aux élèves une succession de quiz sur des thématiques différentes.

Lors de son démarrage, l'application demande à l'utilisateur de saisir des identifiants, puis l'amène sur un écran de sélection d'une thématique. Ce choix se fait parmi huit options : « *Food & Drink* », « *General Knowledge* », « *History* », « *Geography* », « *TV & Movies* », « *Entertainment* », et « *Sports* ».

Chacune de ces thématiques comportent par la suite dix questions, dont les modalités de réponse varient. En effet, la question peut être à choix multiple (comme illustré par la FIGURE 7), peut inviter l'utilisateur à saisir une réponse dans un champ de texte, ou encore peut demander à l'utilisateur de saisir une réponse à l'aide d'un curseur (comme illustré par la FIGURE 8).

Avec cette application, le travail de l'enseignant consiste à préparer une succession de quiz pour ses élèves (*primo-scripting*). Puis, en visualisant les réponses des élèves (*monitoring*), l'enseignant peut envisager de changer les modalités de réponses de certaines questions (*runtime scripting*). Ici, les actions à supporter concernent aussi l'édition d'un scénario pédagogique, le suivi de son déroulement, et sa modification.

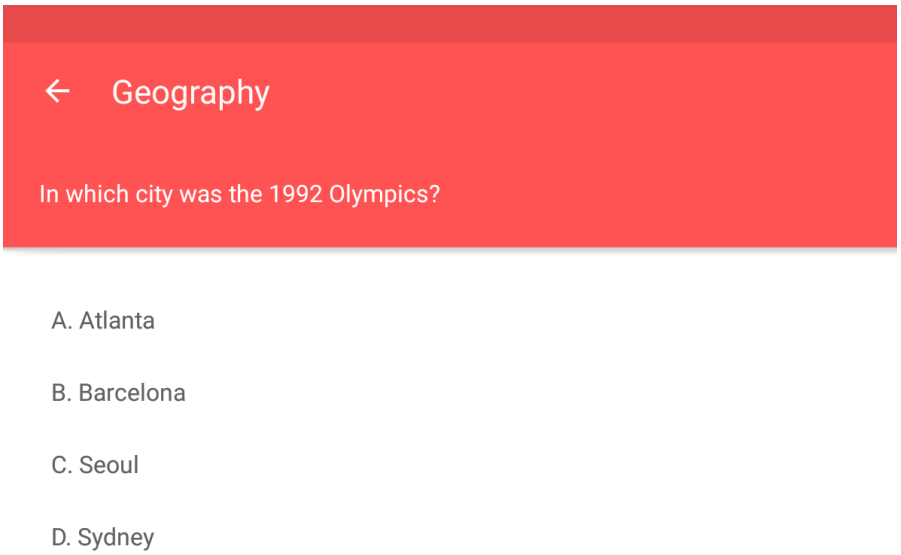


FIGURE 7 – Un exemple de question à choix multiple de la catégorie « *Geography* ».

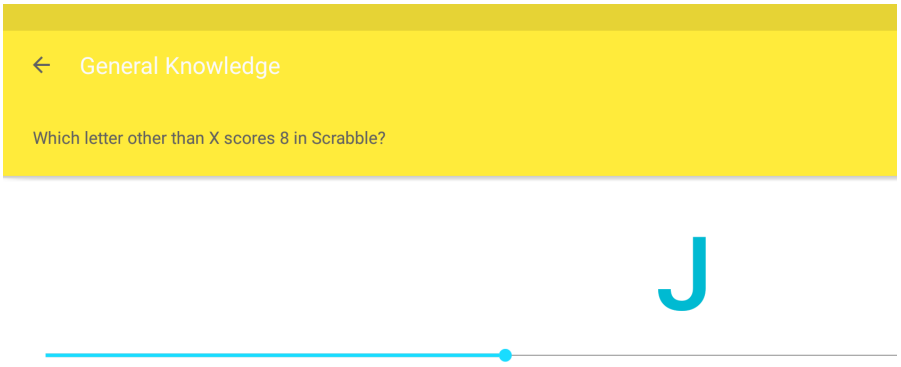


FIGURE 8 – Un exemple de question utilisant un curseur pour sélectionner sa réponse de la catégorie « *General Knowledge* ».

4

CONSTRUCTION DU FRAMEWORK CHAO

Contenu du chapitre

4.1 PRÉSENTATION GÉNÉRALE

4.1.1 *Précisions quant aux objectifs des travaux*

Dans cette thèse, nous cherchons à concevoir un modèle et un framework logiciel. Pour clarifier le terme de framework, nous utilisons la définition suivante :

> « *Un framework est un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.* » (*Framework — Wikipédia* 2016)

Soit une application-élève orchestrable, c'est-à-dire, une application que nous pouvons adapter pour (1) récupérer des informations sur l'activité des élèves et (2) permettre à l'enseignant de la piloter depuis sa tablette. La mise en place d'une situation d'orchestration utilisant cette application-élève et le framework CHAO consiste à :

1. Modéliser le scénario mis en œuvre par l'application-élève à partir du modèle d'orchestration associé au framework CHAO et présenté en SECTION 4.2.3 ;
2. Adapter l'application-élève pour pouvoir synthétiser des informations quant au déroulement de l'activité d'un élève et les communiquer à la tablette de l'enseignant ;
3. Instancier le framework CHAO pour supporter l'orchestration de ce scénario, c'est-à-dire piloter l'application-élève (*primo-scripting* et *runtime scripting*) et visualiser des informations quant à son déroulement (*monitoring*).

4.1.2 *Composants logiciels et graphiques du framework*

Pour clarifier le travail réalisé au cours de cette thèse, nous présentons ici une synthèse des composants du framework CHAO. L'objectif de cette section est de lister les éléments structurants du framework afin de mieux comprendre les principes de sa conception, que nous décrivons dans la suite de ce chapitre.

Le framework CHAO implémente des éléments génériques de deux types :

1. Des composants logiciels qui correspondent à des classes et des méthodes génériques pour le traitement et l'acheminement des données (par exemple, en déclarant les classes structurant un scénario pédagogique et en implémentant les méthodes pour pouvoir le créer et le modifier) ;
2. Des interfaces graphiques abstraites pour structurer les interfaces utilisateurs et la représentation des données sur l'écran de la tablette de l'enseignant (par exemple, en définissant

une représentation spécifique pour le scénario pédagogique et pour les informations de *monitoring* décrivant son déroulement).

Pour concevoir les composants logiciels du framework, nous avons construit un modèle d'orchestration qui s'appuie sur les travaux de caractérisation d'un scénario pédagogique (cette partie est détaillée dans la SECTION 4.2). Le framework implémente donc des composants logiciels qui représentent les éléments de ce modèle d'orchestration.

En lien avec ces éléments logiciels, nous avons conçu des interfaces abstraites. Ces interfaces abstraites définissent l'organisation des éléments graphiques sur l'écran de la tablette de l'enseignant. Pour concevoir ces interfaces abstraites, nous nous sommes appuyés sur des travaux antérieurs de l'équipe qui suggèrent qu'une interface en forme de table est facile à utiliser par un enseignant pour la création d'un scénario pédagogique (Sobreira & Tchounikine, 2015). Nous avons alors repris cette structuration pour l'ensemble des interfaces abstraites proposées par le framework.

Ces composants logiciels et graphiques ont été développés en Java en utilisant le SDK Android (pour *Software Development Kit*), puisque les tablettes à notre disposition au cours de cette thèse fonctionnaient avec ce système d'exploitation.

4.1.3 *Cas d'utilisation du framework*

L'utilisation de ce framework nécessite l'intervention de trois types d'acteurs : un ingénieur pédagogique, un informaticien, et un enseignant (voir FIGURE 9). Ces trois rôles ne sont pas incompatibles entre eux, c'est-à-dire qu'une seule personne peut par exemple jouer à la fois le rôle d'ingénieur pédagogique et d'informaticien. Dans le cadre de cette thèse, nous avons rempli ces deux rôles et avons laissé l'enseignant en charge de l'orchestration des situations d'apprentissage dans sa classe.

La FIGURE 9 présente un diagramme UML de cas d'utilisation dans lequel une application-élève est d'ores et déjà identifiée.

L'ingénieur pédagogique réalise la tâche de modélisation du scénario pédagogique mis en œuvre par l'application-élève en s'appuyant sur le modèle d'orchestration que nous présentons en FIGURE 12. Cela consiste à identifier les types de composant utilisés (par exemple, « Acteur » ou « Tâche ») et les informations de *monitoring* permettant le suivi du scénario. La SECTION 4.2 procure plus d'informations sur ce sujet.

L'informaticien utilise le travail de modélisation de l'ingénieur pédagogique et instancie le framework en conséquence. De la même façon, et parce que l'application-élève n'est pas forcément orchestrable par défaut, l'informaticien est chargé d'apporter les modifica-

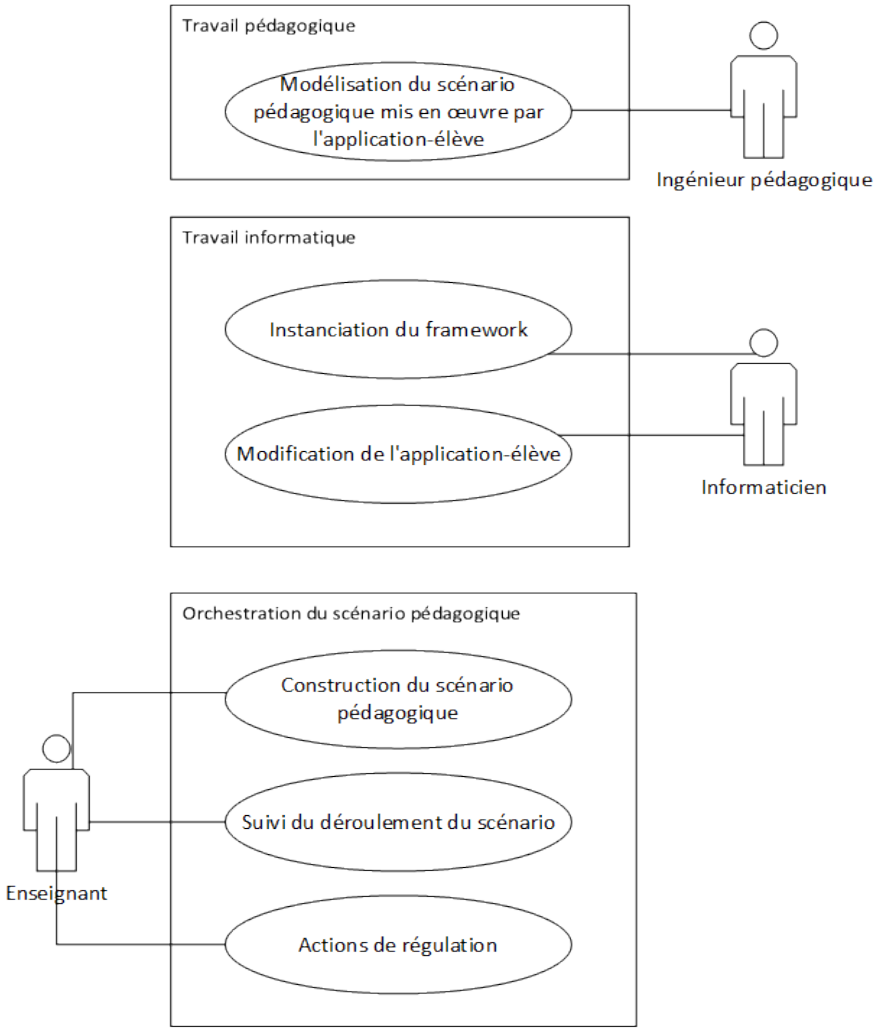


FIGURE 9 – Les interventions de la part de l'ingénieur pédagogique, de l'informaticien, et de l'enseignant dans l'instanciation et l'utilisation du framework.

tions nécessaires pour que celle-ci puisse communiquer et interagir avec une instance du framework, et vice-versa.

L'enseignant est en charge de l'orchestration du scénario mis en œuvre par l'application-élève. C'est lui qui a accès aux outils d'orchestration proposés par une instance du framework. Avec ces outils, il est en mesure de construire un scénario pédagogique, de suivre le déroulement de ce scénario, et de réaliser des actions de régulation comme démarrer, modifier, ou interrompre un exercice sur la tablette d'un élève. En ce sens, il peut être considéré comme l'utilisateur final d'une instance du framework.

En reprenant chacune de ces actions, en les détaillant, et en les plaçant sur une échelle de temps, l'utilisation d'une instance du framework passe par les étapes suivantes :

1. Un acteur (ingénieur pédagogique, enseignant, ou informaticien) identifie une application-élève.
2. L'ingénieur pédagogique modélise le scénario pédagogique mis en œuvre par cette application-élève. En spécifiant ce modèle, il identifie aussi les informations à afficher sur l'écran de la tablette de l'enseignant.
3. L'informaticien s'appuie sur le travail de l'ingénieur pédagogique pour instancier les composants logiciels et graphiques du framework.
4. L'informaticien modifie l'application-élève pour permettre les communications avec l'instance du framework et pour produire des informations en relation avec le déroulement de l'activité.
5. L'enseignant met en œuvre un scénario pédagogique grâce à l'application-élève modifiée et orchestre cette situation en utilisant l'instance du framework.

4.2 MODÈLE D'ORCHESTRATION

4.2.1 *Scénario pédagogique, type de composant, et composant*

Nous nous appuyons sur les travaux de Kobbe et collègues (Kobbe et al., 2007) pour définir les termes de scénario pédagogique, type de composant, et composant :

> Un *scénario pédagogique* est une spécification des activités que doivent réaliser des acteurs, ainsi que des ressources et outils qu'ils ont à leur disposition pour les assister dans ces activités.
>
> Un *type de composant* est une notion abstraite utilisée pour structurer un scénario pédagogique.
>
> Un *composant* est le résultat de l'instanciation d'un type de composant.

Par exemple, dans leurs travaux, Kobbe et collègues identifient cinq types de composant pour spécifier un scénario pédagogique : participant, activité, rôle, ressource, et groupe. Dans ce cas, un participant p_i impliqué dans un scénario pédagogique est un composant de ce scénario.

Enfin, nous faisons une différence entre composants *disponibles* et composants *impliqués* dans la construction d'un scénario pédagogique. Un composant est dit impliqué à partir du moment où l'enseignant décide de s'en servir dans un scénario pédagogique. S'il décide de s'en passer, un composant est alors simplement qualifié de disponible. Par exemple, une ressource r_i peut être un composant disponible mais, dans un scénario donné, l'enseignant peut décider de ne pas s'en servir.

4.2.2 *Représentation et modélisation d'un scénario pédagogique*

Nous décidons de reprendre le principe de représentation d'un scénario pédagogique en table pour sa simplicité (en accord avec le principe de minimalisme) et pour la possibilité de définir autant de types de composant que nécessaire pour spécifier un scénario.

Par exemple, les trois tableaux ci-dessous montrent les représentations possibles en forme de table d'un scénario mis en œuvre par les applications-élève de dictée négociée, SimBûchettes, et Topeka. Les types de composant utilisés pour structurer ces scénarios pédagogiques sont « Acteur », « Phase » ,« Tâche »,« Ressource », et « Production ».

Acteur	Phase	Tâche	Ressource	Production
Les élèves e_1, e_2, e_3, et e_4	Individuelle	Écouter la dictée en entier	Les pistes audio	N/A
Les élèves e_1, e_2, e_3, et e_4	Individuelle	Faire la dictée individuelle	Les pistes audio	La dictée des élèves
Les groupes d'élèves g_1 et g_2	Collective	Négocier l'orthographe des mots	La dictée du groupe, et les mots à négocier m_1, m_2, et m_3	L'orthographe d'un mot négocié, et la justification d'un mot négocié

TABLE 1 – Représentation d'un scénario de dictée négociée en forme de table.

Acteur	Tâche	Ressource	Production
Les élèves e_1, e_2, e_3, et e_4	Faire l'exercice E_1	Principe de position contraint	Le nombre de bûchettes dans chaque boîte
Les élèves e_1, e_2, e_3, et e_4	Faire l'exercice E_2	Principe décimal contraint	Le nombre de bûchettes dans chaque boîte

TABLE 2 – Représentation d'un scénario de SimBûchettes en forme de table.

Acteur	Tâche	Production
Les élèves e_1, e_2, e_3, et e_4	Répondre au quiz Q_1	Les réponses au quiz
Les élèves e_1, e_2, e_3, et e_4	Répondre au quiz Q_2	Les réponses au quiz

TABLE 3 – Représentation d'un scénario de Topeka en forme de table.

Cette représentation d'un scénario pédagogique en forme de table amène à la structuration suivante : les intitulés de chaque colonne correspondent aux types de composant utilisés, et les éléments contenus dans les cellules correspondent aux composants. Par exemple, « Acteur » est un type de composant, et e_1, e_2, e_3, et e_4 sont quatre composants de ce type.

D'autre part, cette représentation nous amène aussi à considérer ce que représente une ligne d'une telle table. C'est pourquoi nous proposons la définition suivante :

> Une *ligne de scénario* est une mise en association de composants de plusieurs types, dont l'objectif est de spécifier une étape attendue du scénario. Elle est donc composée de zéro ou plusieurs composants de chaque type.

La FIGURE 10 présente le modèle complet de scénario pédagogique que nous utilisons dans nos travaux. Ce modèle repose sur la structuration en table et sur la notion de ligne de scénario. Nous y retrouvons les trois types de composants « Acteur », « Tâche », et « Production » qui sont les trois types récurrents quel que soit le scénario pédagogique décrit. De plus, ce modèle peut être instancié en y ajoutant des types de composant complémentaires si nécessaire (comme c'est le cas par exemple pour la dictée négociée avec les types de composant « Phase » et « Ressource »).

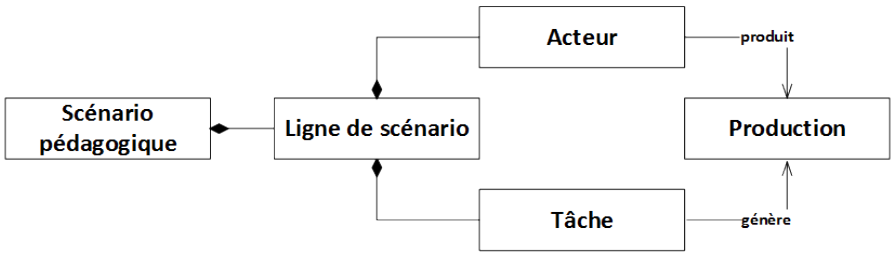


FIGURE 10 – Modélisation UML d'un scénario pédagogique sous la forme de lignes de scénario.

À titre d'illustration, nous avons représenté un scénario pédagogique de dictée négociée en utilisant l'éditeur ediT2 développé au cours de la thèse de Sobreira (Sobreira, 2014). Le résultat de cette représentation est fourni en FIGURE 11.

Le scénario construit comporte quatre lignes de scénario qui peuvent être lues de la manière suivante :

1. L'ensemble d'élèves $\{e_2, e_7, e_1, e_5, e_4, e_3\}$ est associé à la tâche *Écrire la dictée* de la phase *Phase individuelle* avec comme ressource la *Piste audio*.
2. La paire d'élèves $\{e_1, e_5\}$ est associée à la tâche *Négocier* de la phase *Phase collective* avec comme ressources l'ensemble {*Dictée* e_1, *Mot à négocier* m_3, *Mot à négocier* m_2, *Dictée* e_5}.

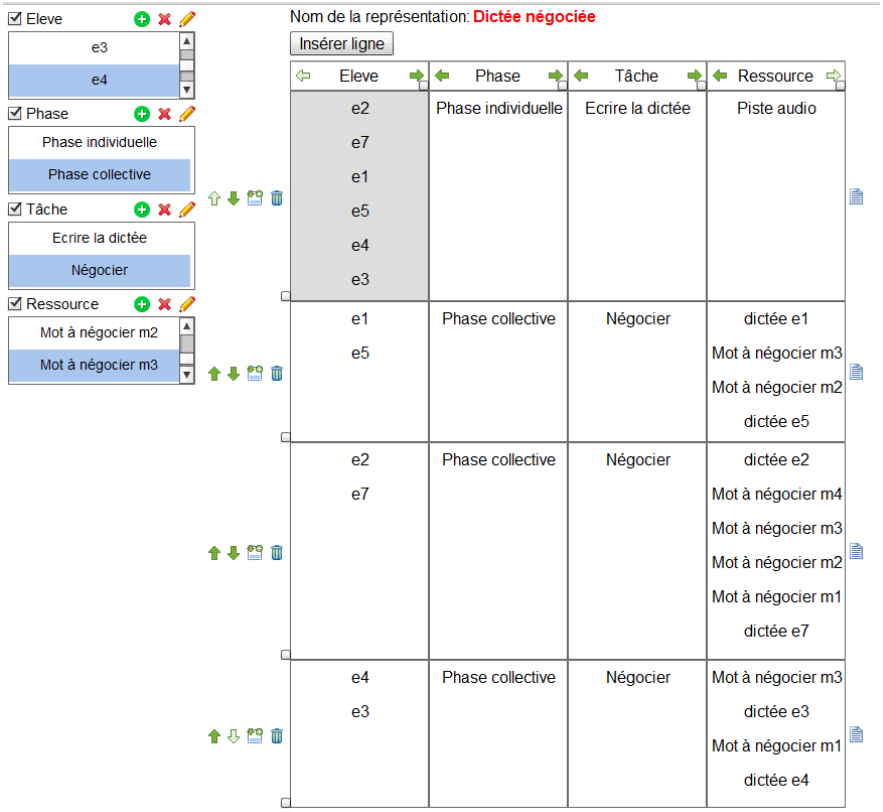


FIGURE 11 – Un scénario de dictée négociée représenté avec ediT2, un éditeur de scénario pédagogique.

3. La paire d'élèves $\{e_2, e_7\}$ est associée la tâche *Négocier* de la phase *Phase collective* avec comme ressources l'ensemble {*Dictée* e_2, *Mot à négocier* m_4, *Mot à négocier* m_3, *Mot à négocier* m_2, *Mot à négocier* m_1, *Dictée* e_7}.
4. La paire d'élèves $\{e_4, e_3\}$ est associée à la tâche *Négocier* de la phase *Phase collective* avec comme ressources l'ensemble {*Mot à négocier* m_3, *Dictée* e_3, *Mot à négocier* m_1, *Dictée* e_4}.

4.2.3 *Le modèle d'orchestration*

Pour construire notre modèle d'orchestration, nous reprenons la modélisation d'un scénario pédagogique présentée en SECTION 4.2.2 et lui ajoutons des éléments du modèle d'orchestration proposé par Tchounikine (Tchounikine, 2013). Ce modèle distingue trois types d'actions d'orchestration : les actions de *primo-scripting*, les actions de *runtime scripting*, et les actions de *monitoring*. Par la suite, nous utiliserons l'acronyme PRM (*Primo-scripting*, *Runtime scripting*, *Monitoring*) pour nous référer à ce modèle. Le résultat de ce travail de modélisation est illustré en FIGURE 12.

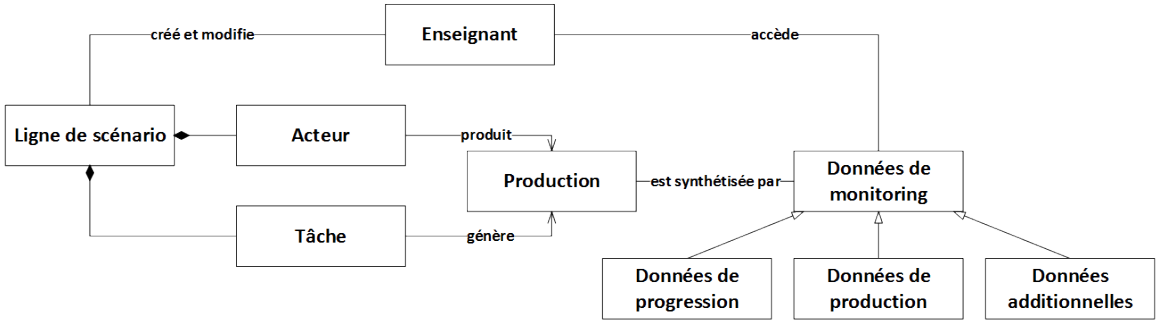


FIGURE 12 – Modèle d'orchestration d'un scénario pédagogique

Les actions de *primo-scripting* sont définies comme les actions de création du scénario pédagogique initial. Ces actions correspondent donc à une phase de conception dans laquelle l'enseignant construit un scénario pédagogique en spécifiant des lignes de scénario. Cela signifie créer des lignes et remplir ses cellules avec des composants impliqués dans le scénario. Ces actions sont représentées par la partie gauche du modèle.

Les actions de *runtime scripting* sont les actions que doit réaliser un enseignant lorsque le scénario nécessite d'être modifié en cours de session. L'enseignant est alors amené à reconsidérer la structure et la composition du scénario pédagogique alors même que les activités sont toujours en cours. Cela signifie créer ou supprimer des lignes de scénario, et/ou modifier leurs contenus en ajoutant ou retirant des composants. De la même façon que pour le *primo-scripting*, les actions de *runtime scripting* sont représentées par la partie gauche du modèle.

Les actions de *monitoring* correspondent à des actions visant à surveiller le déroulement du scénario. Cela consiste à prendre connaissance du travail des élèves au cours des différentes tâches qu'ils ont à réaliser. Pour informer l'enseignant de l'avancement du scénario, nous proposons de construire et de fournir à un enseignant des données de *monitoring* à partir de l'analyse des productions des élèves pour une tâche (comme représenté par la partie droite du modèle). Cela signifie qu'une donnée de *monitoring* décrit la réalisation de la tâche en cours pour un élève, et il y a donc autant de données de *monitoring* qu'il y a d'associations entre un élève et une tâche dans le scénario pédagogique. Nous distinguons trois catégories de données de *monitoring* :

— Des données de progression qui sont définies par la relation suivante :

$$\mathrm{Prog}_{\text{élève}}(T) = \frac{\mathrm{Nb}_{action,\text{élève}}(T)}{\mathrm{Nb}_{\text{référence}}(T)},$$

où,

— T correspond à une tâche ;

  - $\text{Prog}_{\text{élève}}(T)$ correspond à l'information de progression d'un élève pour la tâche T ;
  - $\text{Nb}_{\text{action,élève}}(T)$ correspond au nombre d'actions réalisées par l'élève jusqu'à présent pour construire sa production pour la tâche T ;
  - $\text{Nb}_{\text{référence}}(T)$ correspond au nombre d'actions de référence nécessaires pour réaliser la tâche T.
- Des données de production reflétant, en temps réel, la production d'un acteur pour une tâche ;
- Des données additionnelles à définir au cas par cas.

Pour illustrer ces trois types d'informations, prenons l'exemple du suivi de la phase individuelle de la dictée négociée. La production d'un élève est le texte de la dictée saisi par cet élève. Nous donnons une description des données de *monitoring* ci-dessous :

- Les données de progression d'un élève se présentent sous la forme d'une barre de progression symbolisant le rapport entre le nombre de caractères du texte saisi par cet élève et le nombre de caractères total de la dictée sans faute.
- Les données de production d'un élève affichent, de façon explicite, le texte saisi par cet élève.
- Les données additionnelles d'un élève affichent le nombre de fois que cet élève a écouté chaque piste audio (revoir la FIGURE 3), ou le nombre de modifications apportées à son texte et qui peut être représentatif d'une activité de relecture.

Pour synthétiser, une fois le scénario pédagogique construit et lancé, l'enseignant suit son déroulement grâce aux informations de *monitoring*. Ceci peut l'amener à des actions de régulation de type *runtime scripting* (c'est-à-dire des modifications du scénario pédagogique via des outils informatiques) et/ou à intervenir oralement auprès de ses élèves (c'est-à-dire en se passant du support technologique).

4.3 MODÈLES D'INTERFACE

Le modèle d'orchestration que nous avons construit nous a permis d'identifier les types d'actions d'orchestration proposés à l'enseignant. Dans cette section, nous allons présenter les modèles d'interface construits pour permettre la réalisation de ces actions. En guise d'illustration, nous présentons aussi les interfaces issues de ces modèles que nous avons développées lors de l'instanciation du framework pour supporter l'orchestration de SimBûchettes.

4.3.1 *Modèle d'interface de primo-scripting*

L'interface de *primo-scripting* doit permettre à un enseignant d'éditer un scénario pédagogique. Pour cela, nous avons décider d'utiliser un affichage en forme de table, comme mentionné en SECTION 4.2.2.

La FIGURE 13 présente le modèle d'interface que nous avons conçu. En déplaçant les composants de la partie supérieure vers la zone de création de lignes de scénario, un enseignant peut spécifier et construire une ligne de scénario.

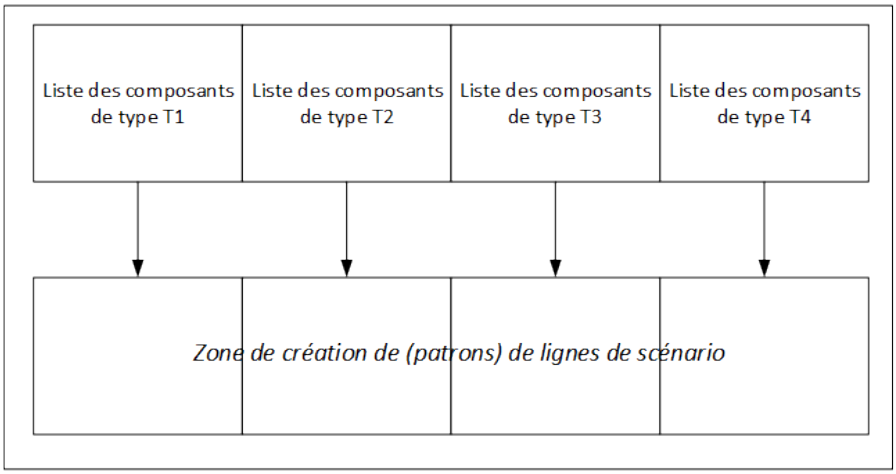


FIGURE 13 – Modèle d'interface de *primo-scripting*.

La création d'un scénario pédagogique passe alors par la création de lignes de scénario. Ce travail peut être fastidieux s'il y a beaucoup de lignes de scénario à construire. Par exemple, l'écran de *runtime scripting* de la FIGURE 15 affiche un scénario pédagogique pour SimBûchettes résultant d'une situation réelle testée en classe. Les lignes de scénario à l'écran (huit lignes sont visibles entièrement sur cet écran) ne constituent qu'un tiers du total de lignes utilisées pour décrire le scénario mis en œuvre. Ceci signifie que, pour éditer ce scénario, l'enseignant doit créer une vingtaine de lignes.

Pour répondre à ce problème, nous avons déclaré des composants abstraits de la forme « Chaque [type de composant] » (la FIGURE 14 montre les composants « Chaque élève » et « Chaque paramètre »). En utilisant ces composants, l'enseignant peut déclarer des *patrons de lignes de scénario* que nous définissons de la façon suivante :

> Un *patron de lignes de scénario* est une ligne de scénario dont le contenu permet de décrire et construire une ou plusieurs lignes de scénario.

Pour illustrer ce modèle d'interface de *primo-scripting*, la FIGURE 14 présente l'interface qu'un enseignant peut avoir à sa disposition en utilisant CHAO-SimBûchettes. Nous y trouvons quatre types de composants (« *Actors* », « *Phases* », « *Tasks* », et « *Parameters* »), les listes de composants associés situées en haut de l'écran, et la zone de création de lignes de scénario. Cette figure utilise les noms d'élèves impliqués lors d'expérimentations, nous les avons donc rendus anonymes partout où cela était nécessaire.

Dans l'exemple illustré en FIGURE 14, l'enseignant a construit un patron de lignes de scénario qui signifie : chaque élève est associé à

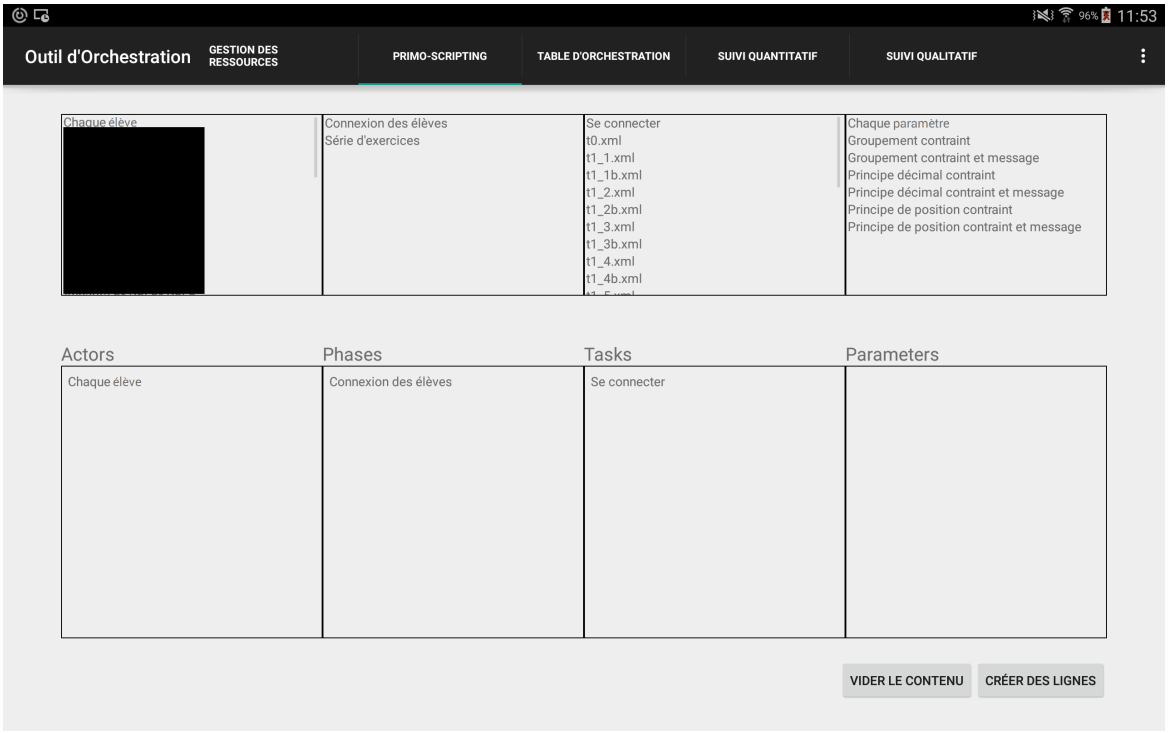


FIGURE 14 – Instance du modèle d'interface de *primo-scripting* utilisée dans CHAO-SimBûchettes.

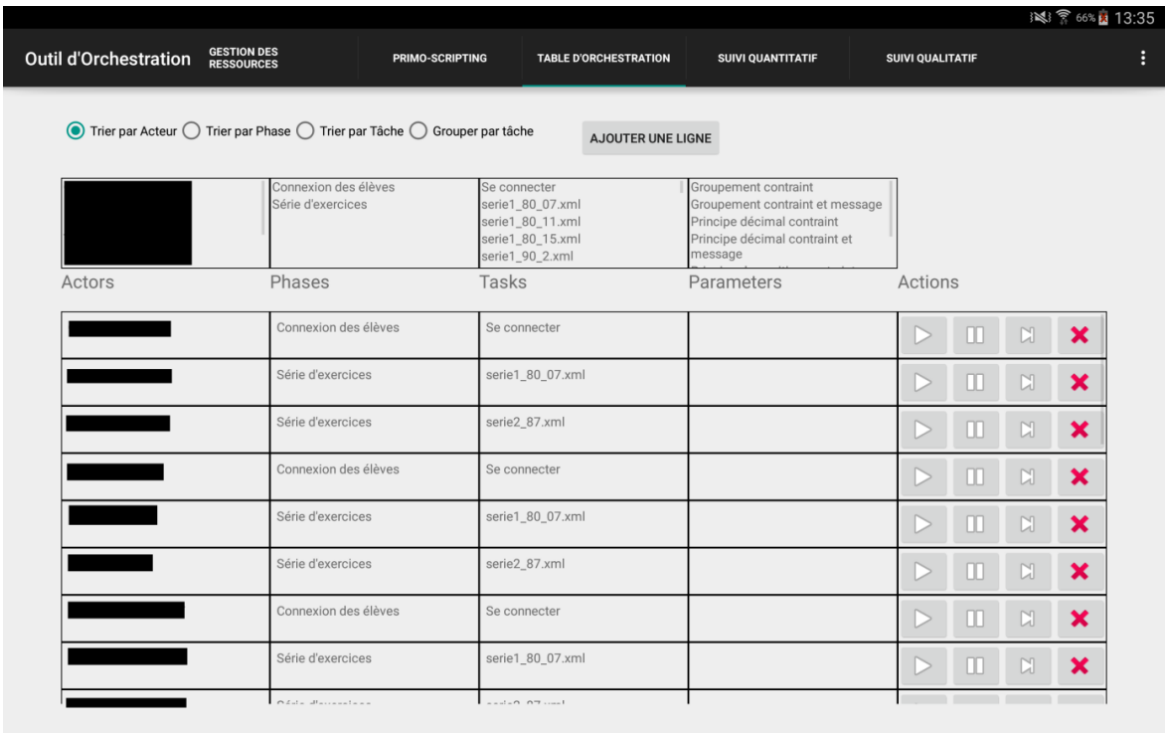


FIGURE 15 – Un exemple de scénario pédagogique contenant un grand nombre de lignes.

la phase « Connexion des élèves » et à la tâche « Se connecter ». En appuyant sur le bouton « Créer des lignes », l'enseignant construit

autant de lignes de scénario qu'il y a d'élèves impliqués dans ce scénario.

L'écran présenté en FIGURE 14 est une instance du modèle d'interface, c'est-à-dire une interface dont le contenu a été adapté pour l'orchestration d'un scénario pédagogique pour SimBûchettes. L'implémentation du modèle d'interface par le framework représente en fait un écran littéralement vide, puisqu'aucun type de composant (et *a fortiori* aucun composant) n'est spécifié par défaut par le framework.

4.3.2 *Modèle d'interface de runtime scripting*

L'interface de *runtime scripting* doit permettre à un enseignant de (1) visualiser le scénario pédagogique construit, (2) de le modifier en manipulant les composants à sa disposition, et (3) de répercuter ces modifications sur le comportement de l'application-élève.

Ce modèle d'interface, présenté en FIGURE 16, possède des points communs avec le modèle d'interface de *primo-scripting*. En effet, nous pouvons retrouver une zone consacrée aux listes de composants et une autre consacrée aux lignes de scénario. De plus, des boutons d'actions sont ajoutés en fin de chaque ligne de scénario. C'est en actionnant ces boutons que l'enseignant rend effectifs les changements appliqués au scénario et modifie le comportement de l'application-élève. Par exemple, nous avions mentionné en SECTION 4.1.3 la possibilité de démarrer un exercice, le mettre en pause, ou encore de modifier certains paramètres de cet exercice.

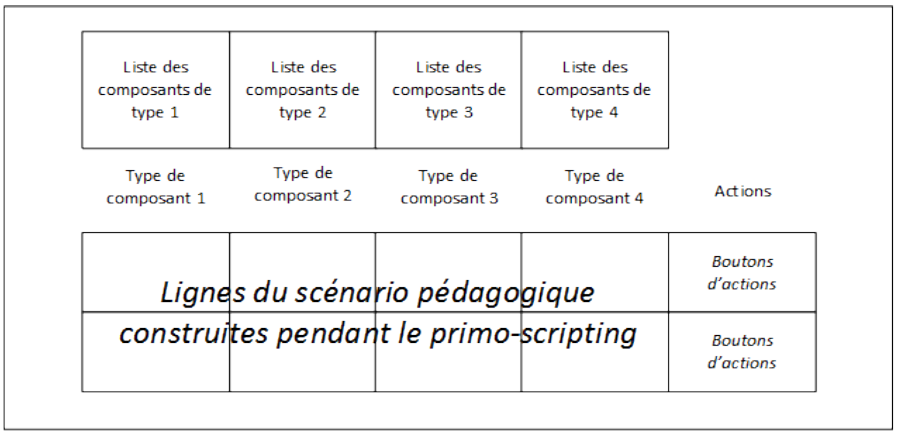


FIGURE 16 – Le modèle d'interface de *runtime scripting* sous forme de *table d'orchestration*, qui affiche les lignes du scénario pédagogique et propose des boutons d'actions pour modifier les applications-élève.

La FIGURE 17 est une instance de l'interface de table d'orchestration proposée par le modèle. Nous y retrouvons les composants que l'enseignant peut manipuler, les lignes de scénario créées pendant le *primo-scripting*, et quatre boutons d'actions (B1, B2, B3, et B4) permet-

tant de (1) démarrer une tâche, (2) de la mettre en pause, (3) de la reprendre, et (4) de supprimer une ligne de la table.

En utilisant cette interface, un enseignant peut par exemple mettre en pause le travail d'un élève grâce au bouton « Mettre en pause », ajouter un paramètre de rétroaction en glissant et déposant un composant de type « Paramètre » dans la cellule correspondante, et relancer l'application-élève en actionnant le bouton « Reprendre ».

De la même façon que pour le *primo-scripting*, l'implémentation du modèle d'interface de *runtime scripting* par le framework affiche un écran vide puisque le framework, par défaut, ne définit pas de type de composant, composant, ligne de scénario, ou bouton d'action.

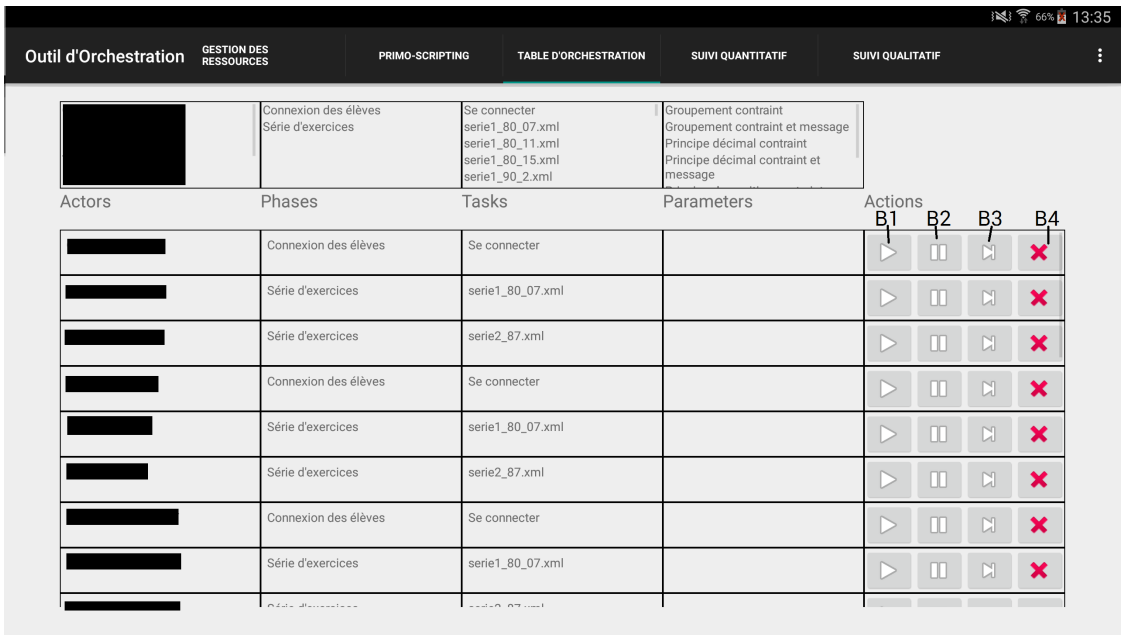


FIGURE 17 – L'instance du modèle d'interface de *runtime scripting* sous forme de table d'orchestration utilisée dans CHAO-SimBûchettes

4.3.3 *Modèle d'interface de monitoring*

L'interface de *monitoring* doit permettre à un enseignant de visualiser les données de progression, de production, ainsi que les données additionnelles décrivant le déroulement du scénario pédagogique. En raison de la taille des écrans des tablettes, nous avons décidé de présenter les données de progression et les données de production sur deux interfaces différentes. Les données additionnelles sont quant à elles accessibles de façon indirecte à partir de ces deux écrans.

Malgré cette séparation en deux interfaces, nous avons conçu un modèle d'interface de *monitoring* unique, visible en FIGURE 18. Ce modèle d'interface reprend une structure en forme de table, où chaque ligne fournit trois informations : l'acteur concerné, la tâche qu'il est en train de réaliser, et les informations de *monitoring* (de progression

ou de production) concernant la réalisation de cette tâche. Enfin, un enseignant peut accéder aux données additionnelles de réalisation d'une tâche par un acteur en cliquant sur la ligne correspondante du tableau (ou plus précisément, en touchant la ligne sur l'écran tactile de sa tablette).

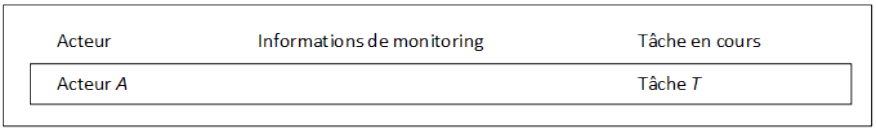


FIGURE 18 – Le modèle d'interface présente les informations de *monitoring* de la tâche en cours pour chaque acteur.

Les FIGURES 19 et 20 présentent les instances du modèle d'interface pour afficher respectivement les données de production et de progression des élèves. Ces deux écrans mentionnent un suivi « qualitatif » et « quantitatif », termes utilisés lors d'une première phase de conception du modèle d'orchestration et qui ont été par la suite remplacés respectivement par « informations de production » et « informations de progression ».

Dans la FIGURE 19, nous pouvons retrouver les trois colonnes « Acteurs », « Données de production », et « Tâche en cours ». Les informations de production (zone 1) sont utilisées pour afficher le contenu des différentes zones illustrées en FIGURE 6 et sont structurées de la sorte :

— La zone 2 présente les symboles correspondant aux groupements de cent bûchettes, aux groupes de dix bûchettes, et aux bûchettes. Ces symboles indiquent ce à quoi réfèrent les données numériques sur chaque ligne.
— La zone 3 affiche le nombre de bûchettes ou de groupements de bûchettes se trouvant dans les zones de stockages (qui correspondent aux zones 1 et 3 de la FIGURE 6).
— La zone 4 affiche le nombre de bûchettes ou de groupements de bûchettes se trouvant dans les boîtes correspondant aux unités, dizaines, ou centaines (qui correspondent à la zone 2 de la FIGURE 6).

Dans la FIGURE 20, les données de production font place aux données de progression synthétisées par le nombre d'actions qu'un élève a entrepris pour réaliser son exercice. En fonction des règles de rétroaction définies pour un exercice, ces actions peuvent être acceptées (zone 1) ou refusées (zone 2). Par exemple, une règle de rétroaction peut empêcher un élève de déposer plus de neuf bûchettes dans la boîte des unités.

Les actions des élèves sont de quatre types (zone 3) :

— Déplacement, qui correspond à l'action de déplacer un élément (c'est-à-dire une bûchette, un groupement de dix, ou un groupement de cent) d'une zone quelconque à une autre.

FIGURE 19 – L'instance du modèle d'interface de *monitoring* pour l'affichage des données de production pour CHAO-SimBûchettes.

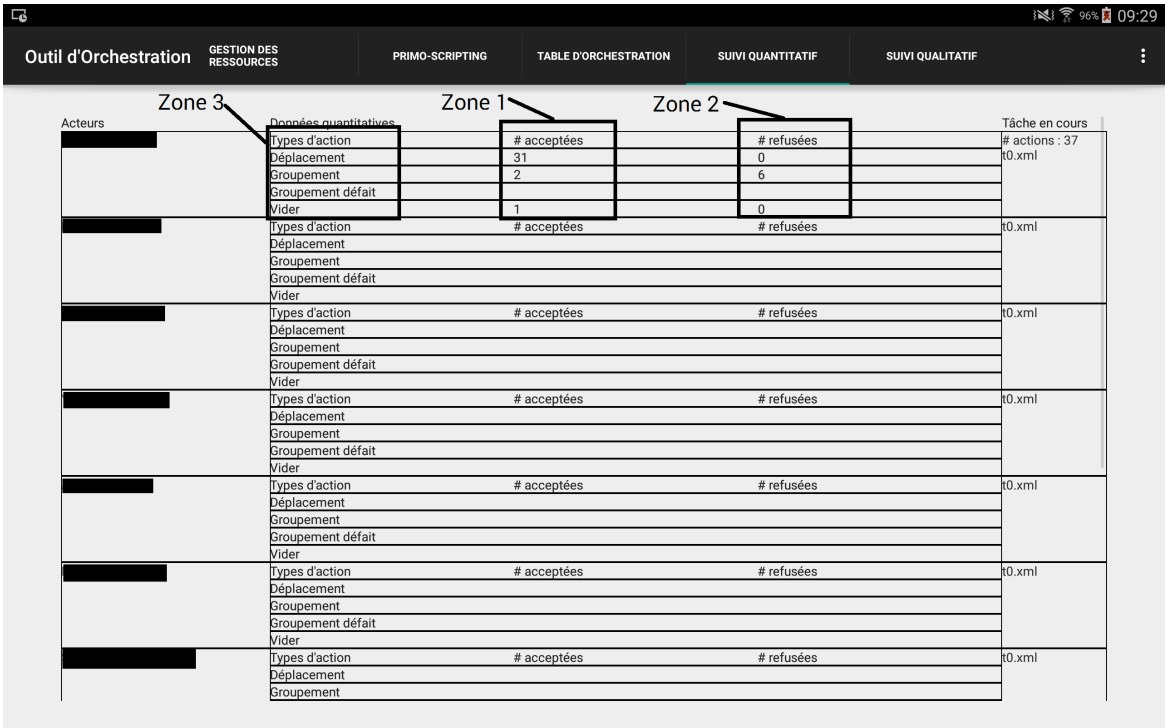


FIGURE 20 – L'instance du modèle d'interface de *monitoring* pour l'affichage de données de progression pour CHAO-SimBûchettes.

— Groupement, qui correspond à l'action de créer des groupements de dix bûchettes à partir de dix bûchettes, ou de créer des groupements de cent bûchettes à partir de dix groupements de dix bûchettes.
— Groupement défait, qui correspond à l'action de défaire un groupement de cent bûchettes en dix groupements de dix bûchettes, ou de défaire un groupement de dix bûchettes en dix bûchettes.
— Vider, qui correspond à l'action de supprimer définitivement un élément du reste de l'exercice.

Les données additionnelles de *monitoring* sont à définir à chaque instanciation du framework CHAO. Pour CHAO-SimBûchettes, ces données affichent l'historique des actions d'un élève lors de la réalisation d'un exercice. Ces informations, qui indiquent la date de l'action, le type d'action, et si l'action est acceptée ou refusée, sont visibles en FIGURE 21.

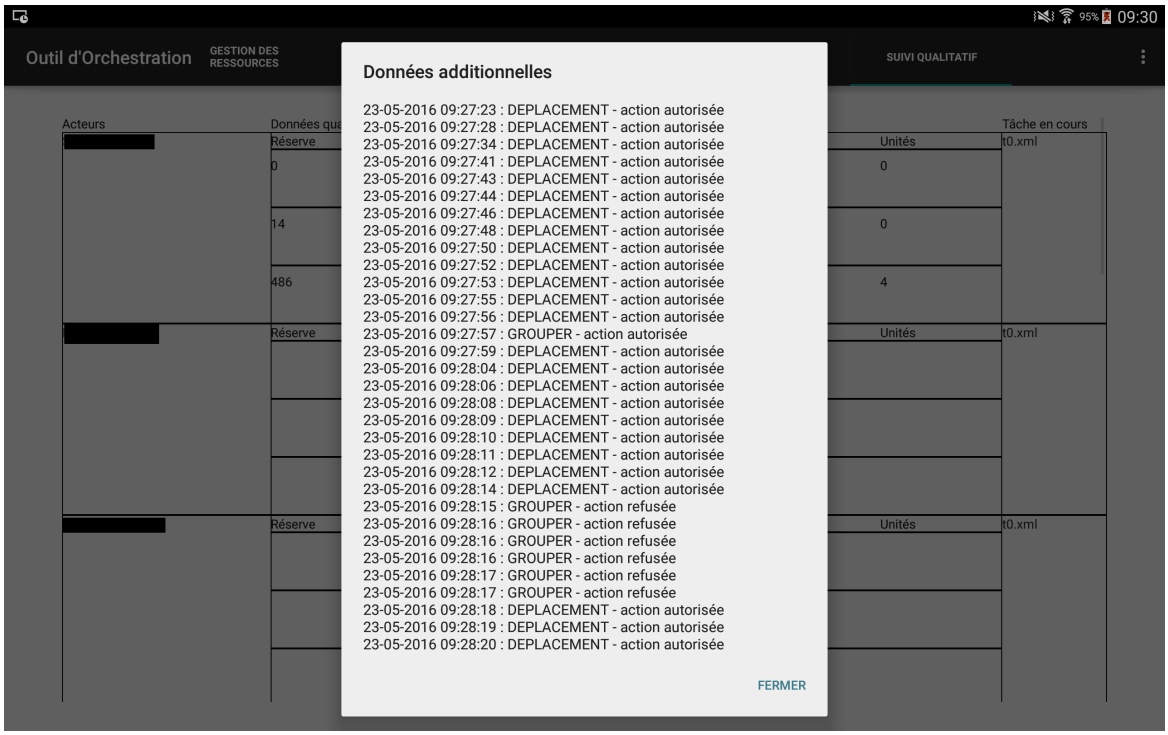


FIGURE 21 – Des données additionnelles affichées dans une fenêtre pop-up.

Le modèle d'interface de *monitoring* implémenté par le framework présente aussi des écrans vides. Seuls les intitulés des colonnes sont affichés, c'est-à-dire « Acteurs », « Données de progression (respectivement de production) », et « Tâche en cours ». La structuration et la représentation des données de *monitoring* restent donc à définir au moment de l'instanciation du framework.

4.4 INFRASTRUCTURE ET ARCHITECTURE INFORMATIQUE DU FRAMEWORK

Nous avons présenté, dans la section précédente, les modèles conçus et utilisés lors de la conception du framework CHAO. L'objectif de cette section est maintenant de décrire l'infrastructure informatique mise en place pour assurer le bon fonctionnement du framework.

4.4.1 *Infrastructure informatique déployée*

4.4.1.1 *Description générale*

L'infrastructure informatique utilisée est illustrée en FIGURE 22 et est composée des éléments suivants :

- Une tablette pour l'enseignant et chacun de ses élèves ;
- Trois serveurs :
  - MQTT pour permettre les communications inter-tablettes ;
  - HTTP pour héberger l'interface de programmation REST ;
  - MySQL pour héberger une base de données ;
- Une interface de programmation de type REST, faisant le lien entre les tablettes et le serveur de base de données ;
- Un routeur, pour créer un réseau local indépendant.

Ces éléments sont décrits de façon plus détaillée dans les sections suivantes.

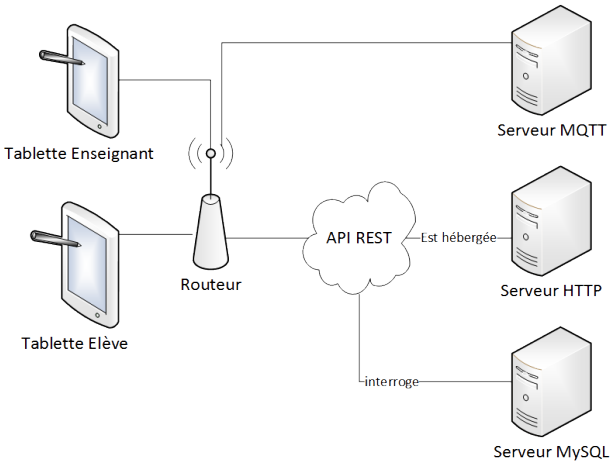


FIGURE 22 – L'infrastructure informatique nécessite l'installation de trois serveurs et le développement d'une API.

4.4.1.2 *Mise en place d'un serveur MQTT*

MQTT (Message Queueing Telemetry Transport) est le protocole utilisé pour assurer la communication entre les tablettes au cours des activités. Nous avons utilisé ce protocole de communication parce

qu'il a été conçu pour être performant même en cas de faible bande passante et parce qu'il permet de différencier facilement les messages selon leurs contenus (*MQTT* 2014). Ce protocole est basé sur le principe du *publier-souscrire*, dont le fonctionnement est présenté en FIGURE 23.

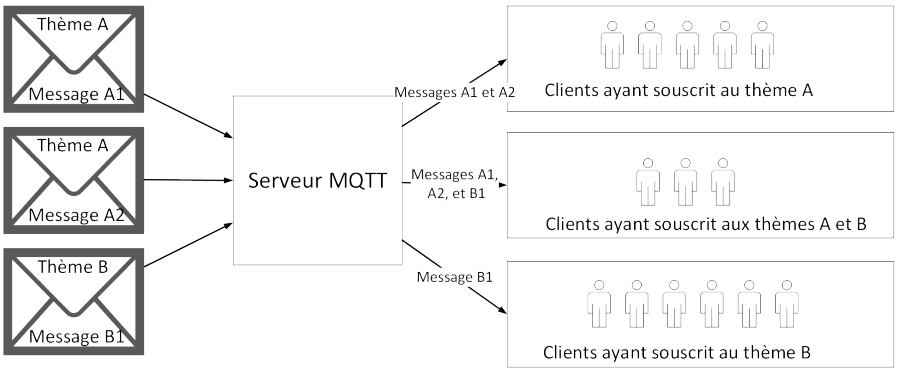


FIGURE 23 – Fonctionnement du protocole MQTT.

Un message MQTT possède deux composantes : un thème et un message. Lorsqu'un client (par exemple la tablette-enseignante ou une tablette-élève) rédige un message, ce client peut lui définir n'importe quel thème. Seuls les clients abonnés à ce thème recevront ce message. Un client abonné à plusieurs thèmes recevra donc tous les messages associés. Dans le cadre de notre framework, nous avons utilisé les thèmes `INSTRUCTIONS`, `PROGRESS`, et `PRODUCTIONS`.

4.4.1.3 *Mise en place d'un serveur MySQL*

Le stockage de données présente plusieurs intérêts. Cela permet de récupérer des informations quant à l'utilisation des technologies par l'enseignant et par les élèves. Cela permet aussi de pouvoir enregistrer les lignes de scénario créées pour en faire des points de sauvegarde. Ce deuxième aspect est important pour la reprise du scénario suite à un problème technique ou si l'activité s'étale sur plusieurs séances.

Plusieurs méthodes de stockage de données s'offraient à nous. Les deux principales options étaient de sauvegarder les données directement sur la tablette dans le répertoire où est installé le framework, ou de les sauvegarder dans une base de données à distance.

Nous avons choisi d'opter pour la deuxième solution pour les raisons suivantes :

— Le fait de délocaliser les données permet aux enseignants d'avoir accès aux informations stockées à tout moment, et pas nécessairement à travers l'utilisation d'une instance du framework CHAO.

— Dans le cas où les données étaient sauvegardées directement dans la tablette-enseignante, des modifications apportées au

code source du framework – voire sa suppression de la tablette – peuvent écraser les informations récupérées jusqu'à présent et entraîner une perte de ces informations de façon irréversible.

Nous utilisons un serveur MySQL, dans lequel est hébergée une base de données qui contient, dans notre implémentation, des tables de trois types :

— Celles représentant les composants de scénario (chaque type de composant est représenté par une table) ;
— Celles représentant les traces d'activités et de manipulations des élèves et de l'enseignant (de la même façon, une table par type de trace d'activité) ;
— Celles représentant les lignes de scénario créées pour spécifier le scénario pédagogique, qui permettent de reprendre l'activité à partir d'un point de sauvegarde précédent.

De plus, il est aussi possible de définir des fonctions et des procédures stockées dans le cas de requêtes effectuées fréquemment. Nous pouvons, par exemple, créer des procédures stockées pour l'ajout de nouvelles traces d'activités des élèves.

4.4.1.4 *Mise en place d'un service web*

Le développement d'un service web est nécessaire pour qu'une tablette puisse interagir avec la base de données. Ce service web prend la forme d'une API de type REST, et est hébergée par un serveur HTTP. Ce service effectue donc des requêtes pour écrire ou lire des informations stockées en base de données. Son fonctionnement est illustré en FIGURE 24.

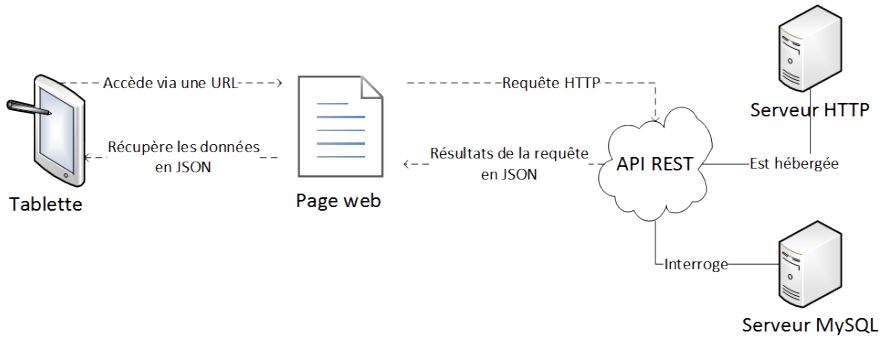


FIGURE 24 – Description du fonctionnement du service web REST.

Pour utiliser cette API, un client doit accéder à une page web via une URL de la forme :
`http://IP_ADDRESS:PORT/OrchestrationAPI/webresources/{resources}`
où :

— `IP_ADDRESS` correspond à l'adresse IP du serveur HTTP ;
— `PORT` correspond au port réseau pour la communication avec ce serveur ;

— `OrchestrationAPI/webresources` est une portion d'URL créée par l'API à partir de son nom ;
— `{resources}` est une variable dont la valeur est une portion d'URL qui indique quel type de ressource le client souhaite manipuler.

Pour récupérer la liste des élèves stockés dans la base de données, un exemple d'URL à construire est :
`http://localhost:1883/OrchestrationAPI/webresources/component/student`

4.4.2 *Architecture logicielle MVP*

Nous avons travaillé et développé le framework pour des tablettes fonctionnant avec le système d'exploitation Android, qui est lui-même développé à partir du langage de programmation Java.

La création d'une nouvelle application Android s'accompagne de la génération de *packages* et de fichiers qui offrent de façon intrinsèque une séparation entre les éléments de logique et les éléments graphiques.

La FIGURE 25 donne un aperçu simplifié de la structure d'une application Android. On y trouve deux répertoires principaux :

— Le répertoire `java` contient tous les fichiers de type Java. Ces fichiers implémentent les fonctionnalités du framework et sont responsables, par exemple, de la mise à jour dynamique des interfaces graphiques, de la gestion des classes métiers, et des échanges de données entre les différents modules du framework.
— Le répertoire `res` (pour *resources* en Anglais) contient un ensemble de fichiers XML qui définissent de façon statique les paramètres de l'application, avec en particulier les éléments définissant la mise en page des interfaces.

```
- java/         // Contient les fichiers .java
|   - package/
|   - package/
|   - ...
- res/          // Ressources, au format XML, utilisées par l'application
|   - drawable/
|   - layout/
|   - values/
|       - dimens.xml
|       - strings.xml
|       - styles.xml
```

FIGURE 25 – Arborescence des dossiers lors de la création d'une application Android.

Nous avons adopté l'architecture Modèle - Vue - Présentation (ou MVP) dont les principes sont illustrés en FIGURE 26. Nous avons décidé d'utiliser cette architecture parce qu'elle concorde avec la structuration d'une application Android.

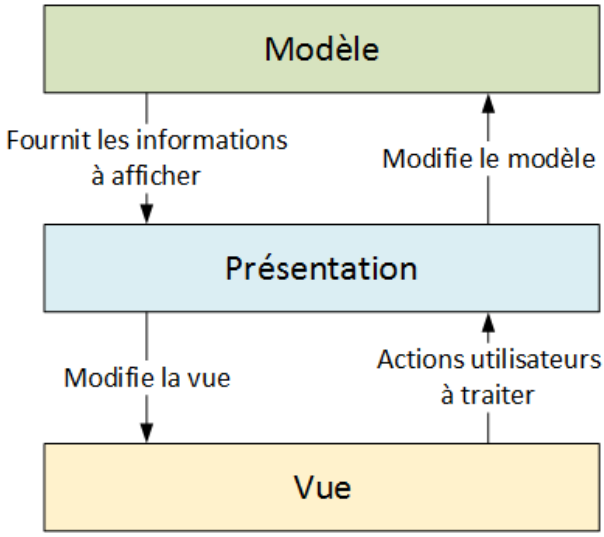


FIGURE 26 – Architecture MVP et échanges de données entre chaque couche.

Le modèle MVP comporte trois couches :

— La couche Modèle contient les classes métiers. Dans notre cas, cette couche contient les classes *métiers* utilisées pour représenter les types de composant et les classes *utilitaires* utilisées pour communiquer avec les serveurs MQTT et HTTP ;

— La couche Vue joue le rôle d'interface avec l'utilisateur. Les éléments de cette couche peuvent être catégorisés en éléments de mise en page (ou *layout*) et éléments graphiques (ou *widgets*). Les éléments graphiques sont alors disposés sur les éléments de mise en page pour créer les interfaces utilisateurs et afficher les données et objets avec lesquels l'utilisateur peut interagir.

— La couche Présentation fait le lien entre les modèles et les vues. Une spécification particulière est que les éléments des couches Vue et Présentation sont couplés, c'est-à-dire qu'il y a exactement une présentation pour un élément de type *layout*. Ce sont les présentateurs qui formatent les données à afficher par les vues, et qui traitent les interactions entre l'utilisateur et l'interface graphique.

L'utilisation de l'architecture MVP présente plusieurs intérêts. Cela permet de structurer le code source du framework de façon précise en catégorisant les objets selon leurs rôles. Cette structuration facilite aussi la prise en main du code source du framework par un informaticien externe.

Dans le cadre de cette thèse, l'architecture logicielle MVP permet surtout de décrire en quoi consistent les modifications à apporter lors de l'instanciation du framework CHAO. En spécifiant le rôle de chaque composant, l'architecture MVP offre une première estimation du travail de programmation à fournir. Par exemple, la couche Modèle est

instanciée en déclarant des classes représentant les types de composant et en modifiant les classes utilitaires ; la couche Présentation est instanciée pour mettre à jour les éléments spécifiques du scénario présents dans les couches Modèle et Vue ; la couche Vue est instanciée en déclarant de nouveaux éléments graphiques pour afficher les informations propres au scénario mis en œuvre.

4.5 IMPLÉMENTATION TECHNIQUE DU FRAMEWORK

Un framework logiciel a pour vocation d'être instancié et éventuellement adapté. Sa structure et son implémentation technique constituent donc des aspects importants à détailler pour comprendre les fonctionnalités proposées. C'est l'objet de cette section.

4.5.1 *Vue d'ensemble*

Nous avons décrit l'infrastructure informatique utilisée et l'architecture logicielle du framework. Nous allons désormais détailler son implémentation technique. Pour cela, nous proposons d'illustrer notre description du framework CHAO avec des copies d'écran de son instance pour l'orchestration de l'application Topeka.

Comme mentionné auparavant, un enseignant orchestre une situation en ayant recours tour à tour à des actions de *primo-scripting*, de *runtime scripting*, et de *monitoring*. Cela signifie qu'il doit pouvoir accéder aux outils d'orchestration à tout moment, depuis la préparation du scénario jusqu'à sa mise en œuvre. Nous avons donc décidé de créer une interface utilisateur pour chaque outil, à laquelle nous pouvons accéder en balayant l'écran de la tablette de droite à gauche. Ainsi, et comme affiché en FIGURE 27, le premier écran est celui de la gestion des composants (c'est aussi l'écran actif, comme le montre la barre soulignant « *Script Components* »). En balayant l'écran vers la gauche (c'est-à-dire en faisant défiler l'écran vers la droite), l'enseignant peut accéder à l'écran de *primo-scripting*, puis à l'écran de *runtime scripting*, et enfin aux écrans de *monitoring* affichant les données de progression et de production.

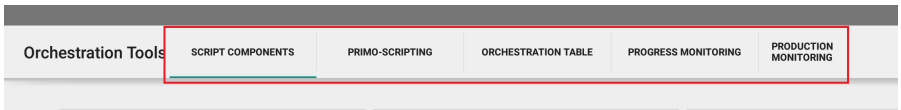


FIGURE 27 – La navigation entre les modules se fait en balayant l'écran ou en touchant un des titres d'écran situés dans la partie encadrée.

La FIGURE 28 illustre de façon détaillée les différents modules qui composent le framework, ainsi que les flux de données entre chaque module. Nous avons aussi repris le code couleur utilisé pour distin-

guer les différentes couches de l'architecture MVP afin d'apporter plus de clarté quant au rôle de ces composants logiciels.

Nous y retrouvons les éléments composant l'infrastructure informatique construite pour le déploiement du framework en classe : les serveurs MQTT, MySQL, et HTTP, et le service web REST.

La couche Modèle est constituée de quatre modules que nous pouvons ranger dans deux catégories différentes :

— Les éléments *métiers*, composés des classes métiers dont le rôle est de décrire les composants de scénario et de représenter le scénario pédagogique à orchestrer ;
— Les éléments *utilitaires*, composés des classes `MQTT` pour la gestion des échanges inter-tablettes, `JsonParser` pour la gestion des échanges avec l'API, et `JsonHelper` pour le traitement des données échangées.

Du fait de l'utilisation de l'architecture MVP, les éléments des couches Présentation et Vue sont couplés. En lien avec la structuration des écrans présentée en FIGURE 27, nous pouvons retrouver :

— Le module de gestion des composants, utilisé pour sélectionner les composants impliqués dans la création du scénario pédagogique ;
— Le module de *primo-scripting*, pour la création d'un scénario pédagogique ;
— Le module de *runtime scripting*, pour modifier le scénario pédagogique et réaliser des actions de régulation ;
— Le module de *monitoring*, pour afficher sur deux écrans différents les informations de *monitoring* liées aux progressions et aux productions des acteurs.

La suite de cette section est consacrée aux descriptions des composants logiciels suivants :

— SECTION 4.5.2 : Les classes métiers ;
— SECTION 4.5.3 : Les classes utilitaires ;
— SECTION 4.5.4 : Le module de gestion des composants ;
— SECTION 4.5.5 : Le module de *primo-scripting* ;
— SECTION 4.5.6 : Le module de *runtime scripting* ;
— SECTION 4.5.7 : Le module de *monitoring*.

Nous concluons cette section en proposant une description chiffrée de l'implémentation du framework, avec en particulier le nombre de classes créées pour chaque couche du modèle MVP ainsi que le nombre total de lignes de code écrites pour le développement du framework CHAO.

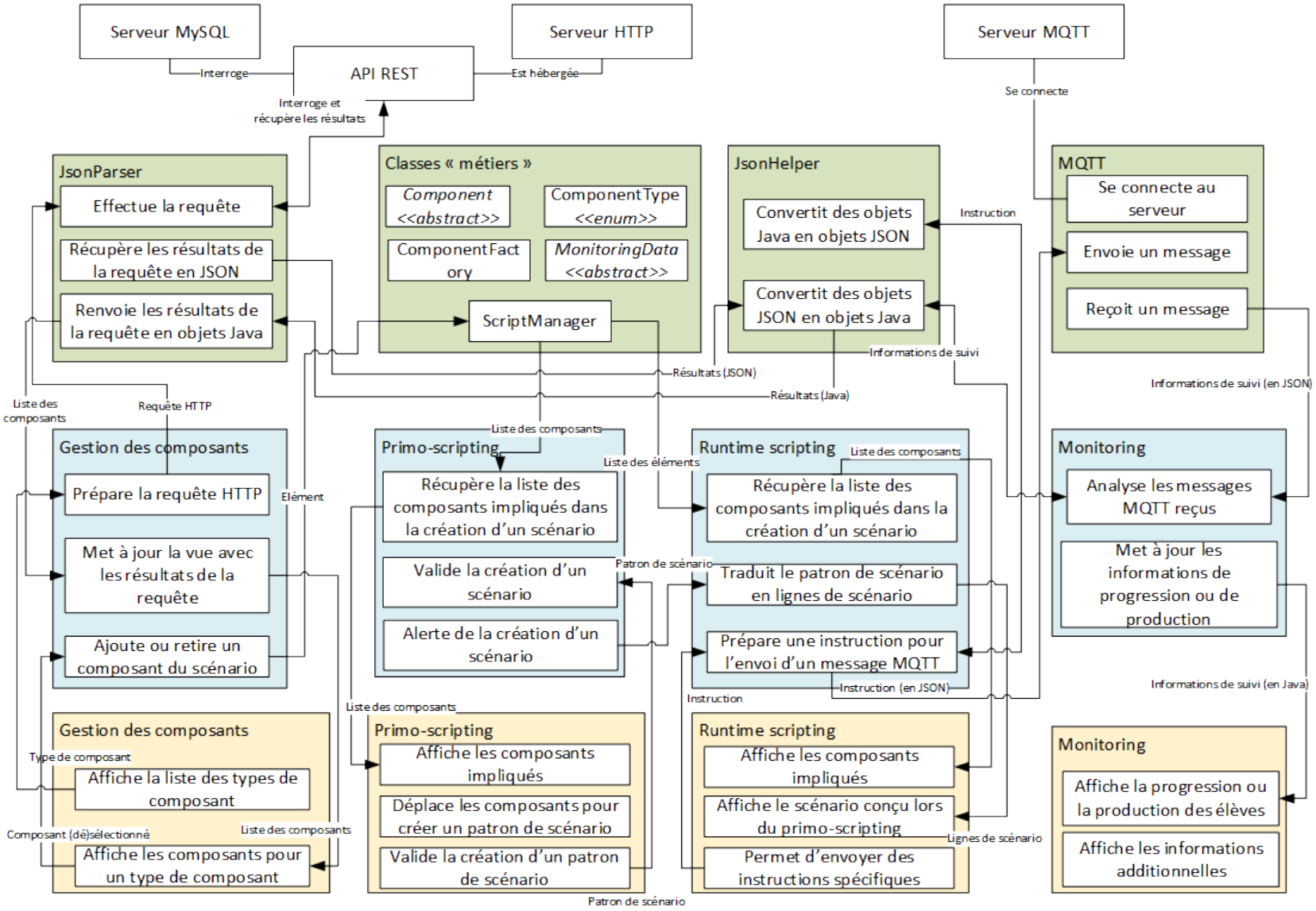


FIGURE 28 – Représentation détaillée des modules constituant le framework, en reprenant les couches de l'architecture MVP.

4.5.2 *Classes métiers*

Les classes métiers décrivent le scénario pédagogique orchestré. Le framework implémente d'ores et déjà les classes présentes en FIGURE 29.

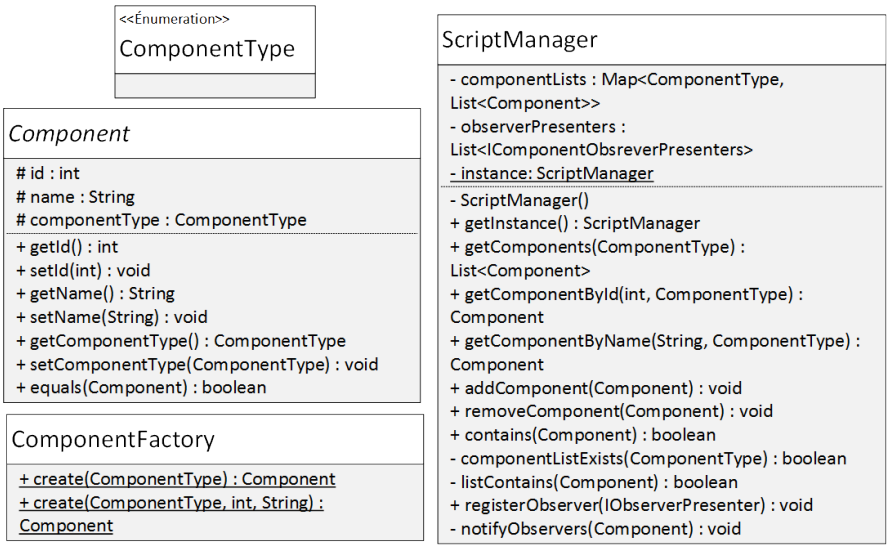


FIGURE 29 – Diagramme UML de classes décrivant l'implémentation des classes métiers.

Nous pouvons y trouver quatre classes principales :

— Une énumération `ComponentType`, qui liste les types de composant utilisés pour structurer le scénario pédagogique.

— Une classe abstraite `Component`, pour pouvoir instancier les composants utilisés lors de l'édition du scénario pédagogique. Après instanciation du framework, il y a autant de classes filles héritant de `Component` qu'il y a d'éléments dans l'énumération `ComponentType`, et les noms de ces classes filles sont identiques aux noms des éléments de l'énumération.

— Une classe `ComponentFactory`, qui implémente le patron de conception *Fabrique* et qui permet d'instancier des objets de type `Component` en utilisant le principe de polymorphisme de la programmation orientée objet.

— Un singleton `ScriptManager`, dont l'intérêt est de gérer les composants impliqués dans la construction du scénario. De plus, ce singleton implémente le patron de conception *Observateur* afin de notifier les modules de *primo-scripting* et de *runtime scripting* de l'ajout ou du retrait d'un composant de la liste des composants impliqués dans la construction du scénario.

Puisque les informations de *monitoring* sont à définir lors de chaque instantiation, nous n'avons pas créé d'interface ni de classe abstraite pour les décrire.

4.5.3 *Classes utilitaires*

Parmi les classes utilitaires, nous pouvons distinguer les cinq classes représentées en FIGURE 30 :

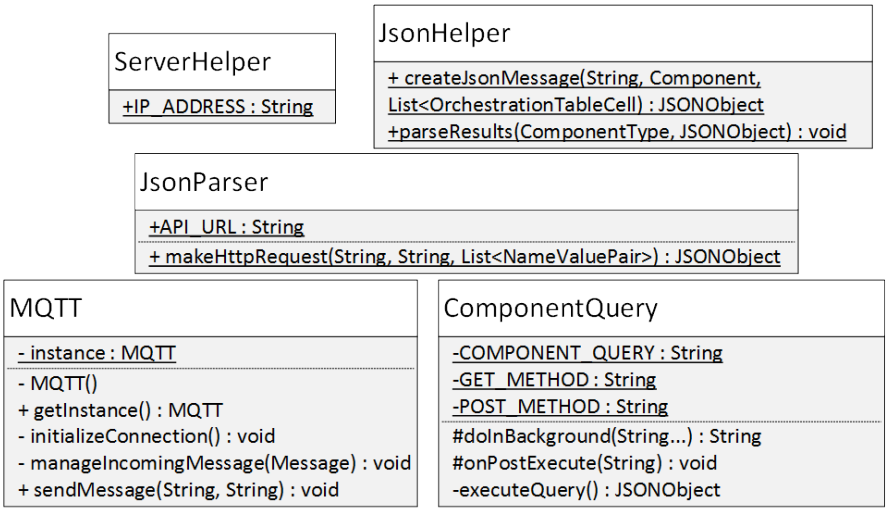


FIGURE 30 – Diagramme UML de classes décrivant l'implémentation des classes utilitaires.

— La classe `ServerHelper`, qui ne contient qu'une ligne sauvegardant l'adresse IP de la machine hébergeant les serveurs HTTP, MQTT, et MySQL.
— La classe `JsonHelper`, qui ne déclare initialement que deux méthodes sans les implémenter. Une méthode sert à créer un message MQTT au format JSON. Une autre méthode est utilisée pour traduire les résultats d'une requête en base de données en objets Java. De manière générale, cette classe implémente les méthodes de traitement et de conversion d'objets et nécessite d'être complétée à chaque instanciation.
— La classe `JsonParser`, qui implémente une méthode statique permettant de faire une requête HTTP pour échanger avec le service web. Cette méthode nécessite de fournir une URL et de spécifier la méthode (GET ou POST) et les éventuels paramètres complétant l'URL.
— Le singleton `MQTT`, qui permet de se connecter au serveur MQTT, d'envoyer un message en mentionnant son thème et son contenu, et de recevoir les messages arrivants qui sont ensuite traités par le `JsonHelper`.
— La classe `ComponentQuery`, qui permet d'effectuer une requête asynchrone en créant un nouveau *thread*. Lorsque cette requête a récupéré des résultats, la méthode `onPostExecute()` spécifie les traitements à réaliser sur ces résultats.

4.5.4 *Module de gestion des composants*

4.5.4.1 *Principes de l'interface utilisateur*

Le module de gestion des composants existe pour (1) récupérer l'ensemble des composants disponibles et sauvegardés en base de données, et (2) pour offrir une interface utilisateur permettant de visualiser ces composants disponibles et d'y sélectionner ceux qui sont impliqués dans l'édition du scénario pédagogique.

La FIGURE 31 illustre l'interface utilisateur de ce module. La zone 1 présente, sous la forme de boutons, la liste des types de composant utilisés pour structurer le scénario pédagogique mis en œuvre par l'application-élève Topeka. La zone 2 présente, sous la forme de cases à cocher, les composants disponibles en base de données pour un type spécifié.

En actionnant le bouton « *Task* », la classe `ComponentQuery` est appelée pour effectuer une requête HTTP et récupérer les données présentes dans la table SQL sauvegardant les composants de ce type. Cela entraîne l'affichage des composants disponibles et nous retrouvons les huit thématiques proposées par Topeka. En cochant (respectivement décochant) une de ces cases, le `ScriptManager` exécute la méthode `addComponent()` (respectivement `removeComponent()`) qui ajoute (respectivement retire) un composant de la liste des composants impliqués dans le scénario.

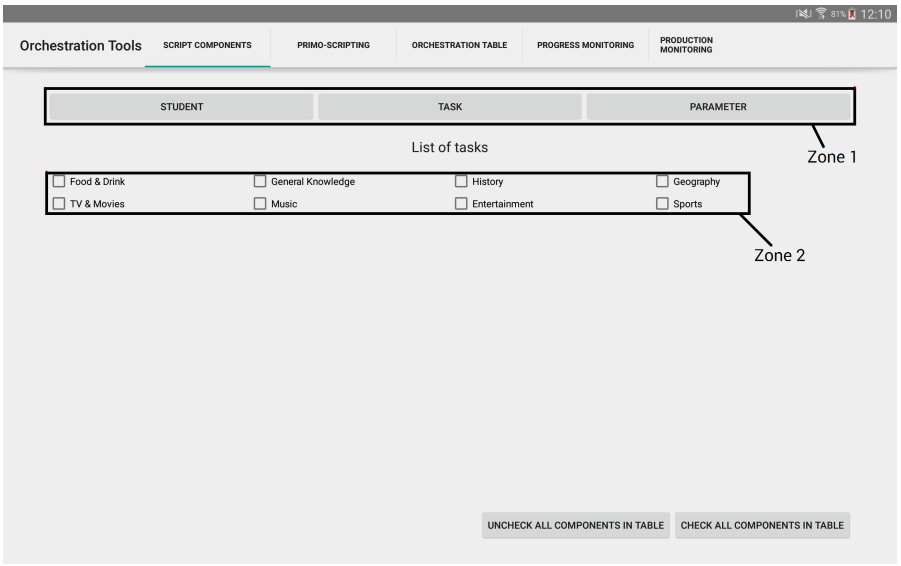


FIGURE 31 – Vue du module de gestion des composants de CHAO-Topeka.

Cette interface présente un intérêt pour le framework CHAO grâce son caractère générique. En effet, elle est facilement adaptable pour tout type de scénario pédagogique : il suffit de modifier l'énumération `ComponentType` pour définir d'autres types de composant et de s'assurer que la requête HTTP renvoie les résultats attendus.

4.5.4.2 *Implémentation technique*

La FIGURE 32 présente les quatre classes créées pour implémenter le module de gestion des composants :

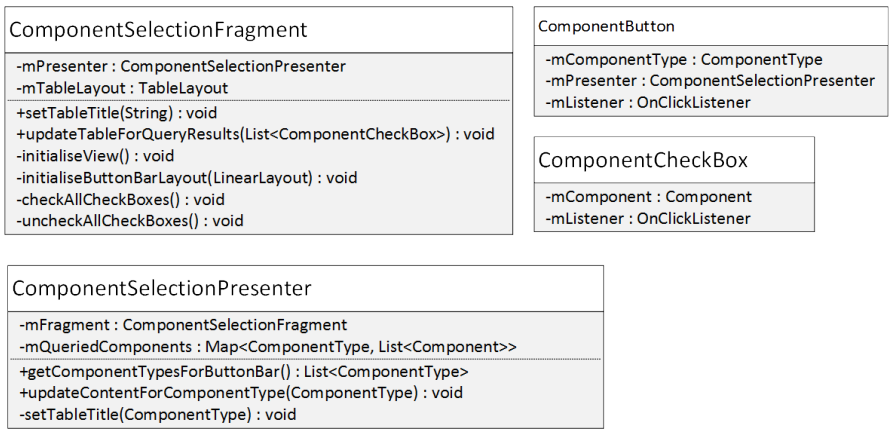


FIGURE 32 – Diagramme UML de classes décrivant l'implémentation des classes utilisées pour le module de gestion des composants.

— La classe `ComponentSelectionFragment` correspond à l'élément Vue du modèle MVP pour ce module. C'est pourquoi nous trouvons des méthodes pour initialiser l'interface utilisateur, des méthodes pour mettre à jour cette interface, et des méthodes pour traiter les interactions avec cette interface.
— La classe `ComponentButton` permet d'afficher les boutons qui se trouvent dans la zone 1 de la FIGURE 31. Cette classe possède trois attributs :
  — `mComponentType` qui précise à quel type de composant un objet de cette classe fait référence ;
  — `mPresenter` qui est le résultat de la composition d'un élément Présentation, et qui offre la possibilité d'appeler la méthode publique `updateContentForComponentType()`.
  — `mListener` qui implémente le comportement de ces boutons. Dans notre cas, il s'agit d'appeler la méthode de la présentation : `updateContentForComponentType()`.
— La classe `ComponentSelectionPresenter` correspond à l'élément Présentation du modèle MVP pour ce module. Grâce à la méthode `getComponentTypesForButtonBar()`, les éléments de l'énumération `ComponentType` sont récupérés puis passés à la Vue pour créer les `ComponentButton`. Grâce à la méthode `updateContentForComponentType()`, cette classe peut faire appel à la classe utilitaire `ComponentQuery` pour effectuer une requête HTTP.

4.5.5 *Module de primo-scripting*

4.5.5.1 *Principe de l'interface utilisateur*

Comme spécifié en SECTION 4.3.1, le modèle d'interface de *primo-scripting* présente deux zones distinctes permettant de (1) afficher les composants impliqués dans la construction du scénario pédagogique et (2) construire des patrons de lignes de scénario.

L'écran illustré en FIGURE 33 illustre l'instance de ce modèle d'interface pour CHAO-Topeka. La zone 1 est divisée en autant de cellules qu'il y a de types de composant déclarés. Une cellule contient les composants impliqués dans le scénario relativement à son type de composant. La zone 2 représente la zone de création de patrons de lignes de scénario.

Une telle interface offre un mécanisme de spécification de patrons de lignes de scénario en glissant et déposant des composants de la zone 1 vers la zone 2. Ce mécanisme est indépendant des composants déplacés, ce qui explique son intérêt dans le cadre du framework CHAO.

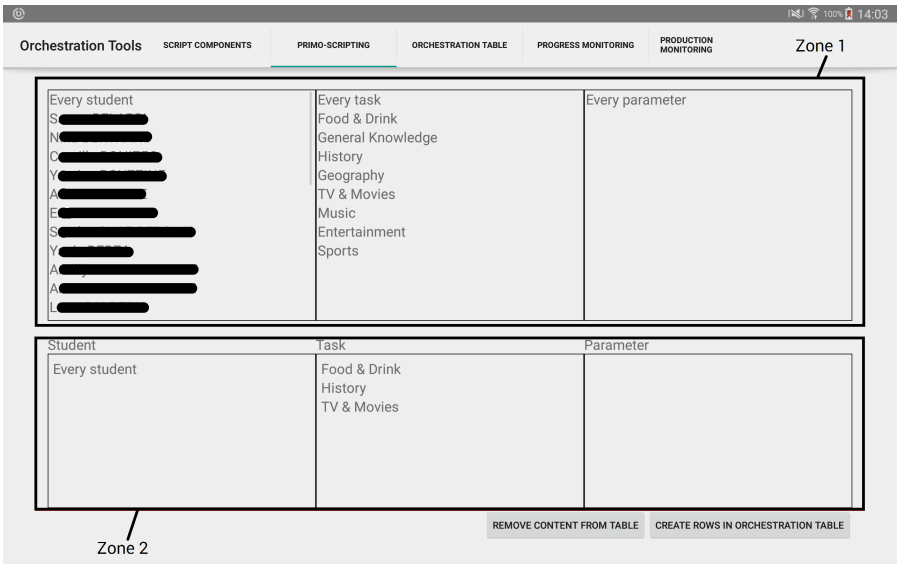


FIGURE 33 – Vue du module de *primo-scripting* CHAO-Topeka.

Afin de fluidifier la construction d'un scénario pédagogique, l'interface présente deux fonctionnalités complémentaires proposées par le framework :

— La déclaration des composants abstraits pour faciliter la création de patrons de lignes de scénario (« *Every Student* », « *Every Task* », et « *Every Parameter* ») ;
— La création d'un bouton permettant de vider le contenu de la zone 2, au lieu de retirer les composants les uns après les autres par des actions de glisser-déposer.

4.5.5.2 *Implémentation technique*

L'implémentation du module de *primo-scripting* repose sur la déclaration de deux classes spécifiques au module de *primo-scripting*, et de cinq autres qui sont aussi utilisées par le module de *runtime scripting*. La FIGURE 34 montre le diagramme UML de ces classes.

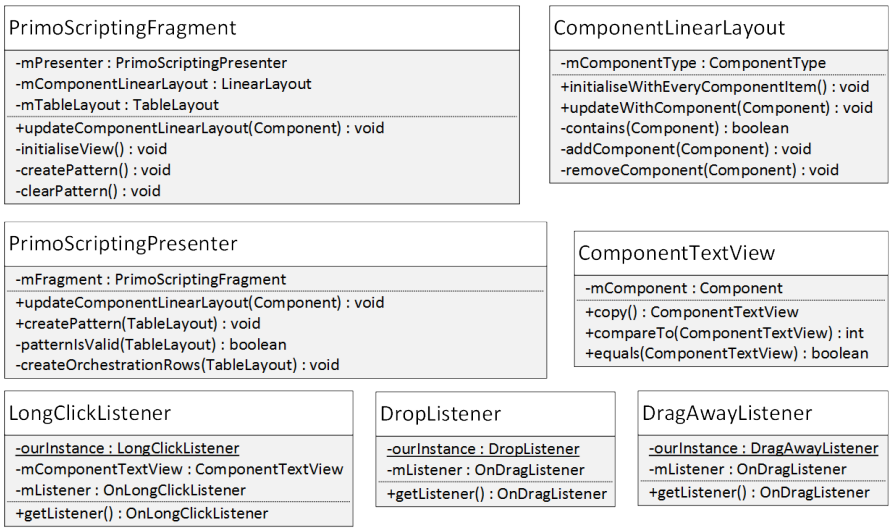


FIGURE 34 – Diagramme UML de classes décrivant l'implémentation du module de *primo-scripting*.

Les deux classes spécifiques au module de *primo-scripting* sont :

— Le `PrimoScriptingFragment`, qui correspond à la vue de ce module. Cette classe est chargée d'initialiser l'interface utilisateur avec en particulier l'affichage des listes des composants impliqués dans le scénario (composants de la zone 1 de la FIGURE 33), et d'initier le processus de validation du patron et de création de lignes de scénario.

— Le `PrimoScriptingPresenter`, qui correspond à la présentation de ce module. Cette classe est chargée de transmettre les composants impliqués dans le scénario du `ScriptManager` vers la vue, et de valider le contenu d'un patron de scénario avant de créer les lignes de scénario correspondantes.

Les cinq classes dont l'utilisation est partagée entre ce module et celui de *runtime scripting* sont :

— La classe `ComponentLinearLayout`, qui représente un conteneur dans lequel sont affichés les composants disponibles pour un type donné. La zone 1 de la FIGURE 33 présente trois objets de cette classe.

— La classe `ComponentTextView`, qui représente un composant de scénario et qui peut être déplacé par des actions de glisser-déposer. Par exemple, « *History* » est un `ComponentTextView`.

— La classe `LongClickListener`, qui identifie le début d'une action de glisser-déposer.

— La classe `DropListener`, qui identifie l'action de déposer un composant dans la zone de création de patrons.
— La classe `DragAwayListener`, qui identifie l'action de déposer un composant en dehors de la zone de création de patrons (pour le retirer de cette zone).

4.5.6 *Module de runtime scripting*

4.5.6.1 *Principes de l'interface utilisateur*

Le modèle de table d'orchestration, spécifiée en SECTION 4.3.2, est repris ici pour représenter l'interface de *runtime scripting*. Cette interface est composée de deux parties (voir FIGURE 35). Dans la zone 1, nous retrouvons les listes des composants impliqués dans la construction d'un scénario pédagogique. Dans la zone 2, la table d'orchestration affiche les lignes de scénario créées à partir du patron de lignes de scénario construit en FIGURE 33.

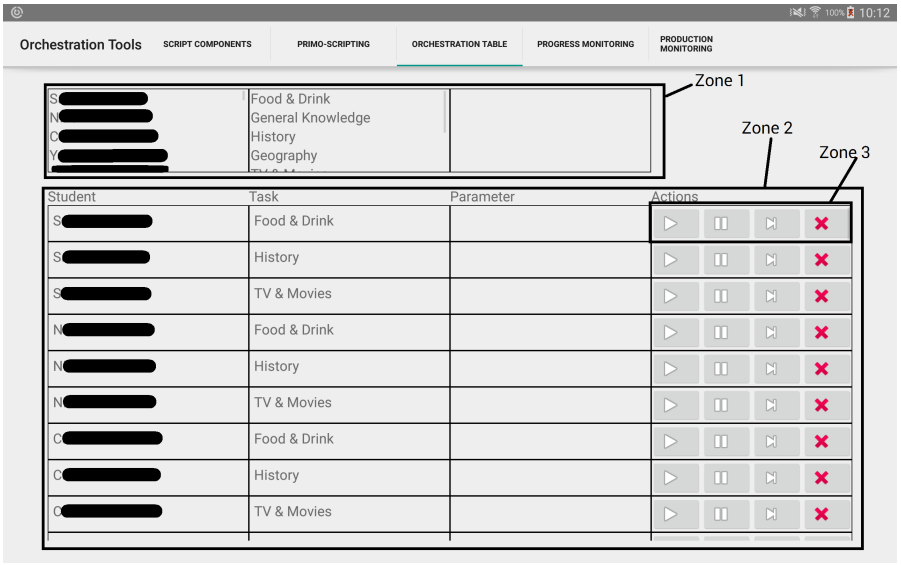


FIGURE 35 – Vue du module de *runtime scripting* pour CHAO-TOPEKA.

Dans le cadre du framework CHAO, cette interface présente l'intérêt d'offrir le même mécanisme de manipulation des composants que celui utilisé pour le module de *primo-scripting*, et d'afficher des lignes de scénario avec une représentation générique qui peut être adaptée à différents types de scénarios pédagogiques. Cette interface fournit aussi un espace réservé aux boutons d'actions (zone 3 de la figure). Par défaut, le framework propose quatre boutons sur chaque ligne de scénario. Seul le dernier est implémenté pour supprimer la ligne de scénario de la table d'orchestration.

Ce dernier bouton a plusieurs intérêts. Un premier intérêt est que l'enseignant peut purger l'affichage des lignes de scénario en supprimant les lignes qui ne sont plus utilisées par la suite. Un second

intérêt est que cela permet de réduire, en cours de session, le nombre de lignes de scénario total à afficher. En effet, l'écran en FIGURE 35 n'affiche simultanément que neuf lignes de scénario. Bien que l'écran permette de faire défiler les informations affichées, il peut être utile de pouvoir supprimer des lignes qui n'ont plus d'utilité dans la suite de la session.

4.5.6.2 *Implémentation technique*

En plus des cinq classes mentionnées lors de la description de l'implémentation du module de *primo-scripting*, le module de *runtime scripting* a nécessité l'implémentation de quatre classes spécifiques présentées en FIGURE 36 :

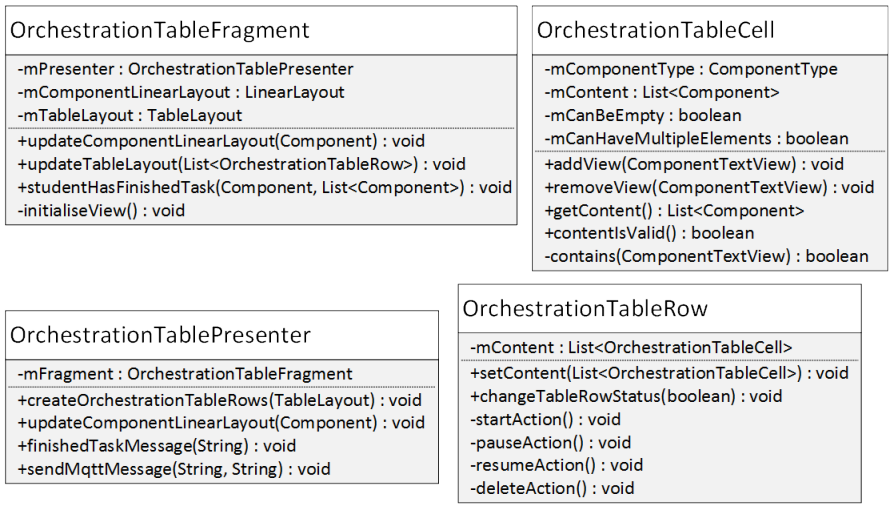


FIGURE 36 – Diagramme UML de classes décrivant l'implémentation du module de *runtime scripting*.

— La classe `OrchestrationTableFragment` est la vue de ce module. Elle affiche les `ComponentLinearLayout` qui contiennent les composants impliqués (zone 1 de la FIGURE 35) et la table d'orchestration (zone 2). Pour faciliter la lecture de cette table, la méthode `studentHasFinishedTask()` peut être implémentée pour, par exemple, changer la couleur de fond d'une ligne de scénario.

— La classe `OrchestrationTableRow` est utilisée pour représenter une ligne de scénario. La méthode `setContent()` est appelée lors de la création d'une ligne de scénario. Elle est utilisée pour initialiser le contenu d'une ligne afin de refléter le scénario construit avec le module de *primo-scripting*. Les quatre méthodes d'actions sont aussi déclarées, et seule `deleteAction()` est implémentée. Le comportement des trois boutons restants est à déterminer au moment de l'instanciation du framework.

— La classe `OrchestrationTableCell` est utilisée pour représenter une cellule d'une ligne de scénario. Les méthodes implémentées permettent de traiter l'ajout ou le retrait d'un objet de type `ComponentTextView` et de récupérer le contenu d'une cellule. Les attributs déclarés peuvent être modifiés lors de l'instanciation du framework pour spécifier si une cellule peut être vide ou si elle peut accueillir plusieurs composants.
— La classe `OrchestrationTablePresenter` est la présentation du module de *runtime scripting*. Au moment de la création de lignes de scénario à partir de patrons de lignes, la méthode `createOrchestrationTableRows()` est appelée et les lignes de scénario créées sont ensuite passées à la vue pour affichage. Lorsqu'un des trois premiers boutons d'actions est actionné, cette classe est chargée d'initier la création de messages MQTT en faisant appel à la classe utilitaire `JsonHelper`. Puis, ces messages sont envoyés par la classe `MQTT`.

4.5.7 *Module de monitoring*

4.5.7.1 *Principes de l'interface utilisateur*

Le modèle d'interface de *monitoring* spécifié en SECTION 4.3.3 propose d'afficher une table dont les colonnes sont : « Acteur », « Informations de *monitoring* », et « Tâche en cours ». Une telle structure permet d'afficher, sur un seul écran, la tâche en cours de chaque élève et les informations de progression ou de production correspondantes.

Avec les informations de progression, l'objectif est de proposer une visualisation simple et facile à lire pour qu'un enseignant puisse en déduire rapidement la progression de l'ensemble des élèves impliqués dans le scénario. Avec les informations de production, l'objectif est de fournir une représentation fidèle de la production de chaque élève compte-tenu de la taille d'écran d'une tablette.

La copie d'écran illustrée en FIGURE 37 montre l'interface de *monitoring* des productions des élèves de CHAO-Topeka. Dans cet exemple, nous avons choisi de représenter ces informations sous la forme d'une barre de progression où chaque cellule correspond à une question. La réponse de l'élève est inscrite à l'intérieur de chaque cellule, et la couleur de fond indique si cette réponse est correcte ou incorrecte.

La représentation de ces informations de *monitoring* est à déterminer au moment de l'instanciation du framework CHAO. En effet, les barres de progression sont des visualisations simples et bien adaptées pour des scénarios de dictée négociée ou Topeka. Dans d'autres cas, comme pour SimBûchettes, il est nécessaire de construire des représentations plus complexes.

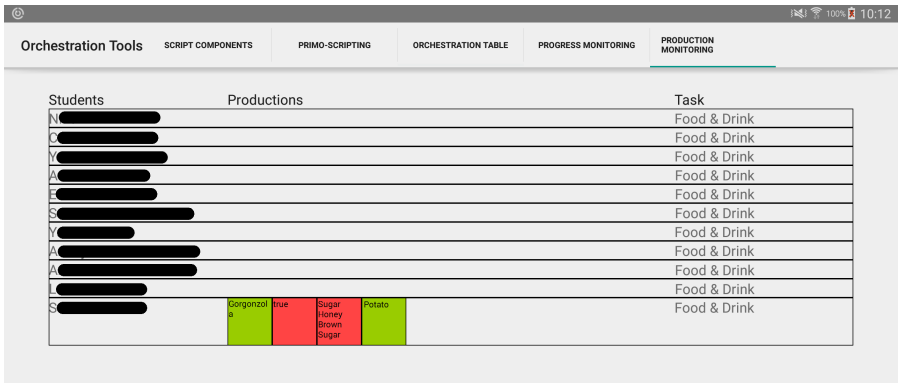


FIGURE 37 – Vue du module de suivi des productions pour CHAO-TOPEKA.

4.5.7.2 *Implémentation technique*

La FIGURE 38 présente les neuf classes créées pour implémenter le module de *monitoring*. Parmi ces neuf classes, trois sont des classes abstraites qui ont toutes été étendues à deux reprises (une fois pour le suivi de la progression et une autre fois pour le suivi de la production). Nous allons donc concentrer notre description sur ces trois classes abstraites :

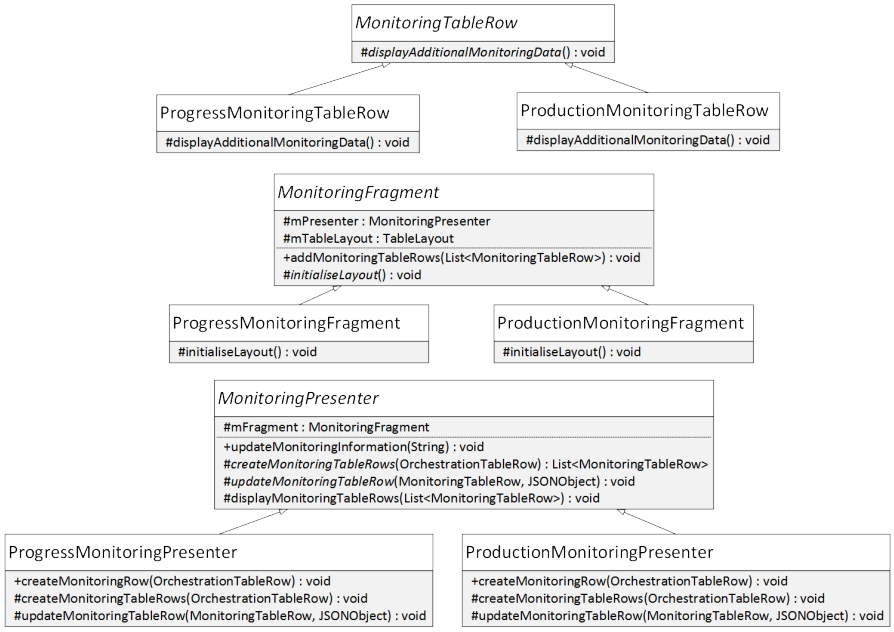


FIGURE 38 – Diagramme UML de classes décrivant l'implémentation du module de *monitoring*.

— La classe abstraite `MonitoringTableRow` est utilisée pour représenter les informations de *monitoring*. Chaque ligne présente dans la FIGURE 37 est donc un objet de cette classe. Lors de l'instanciation du framework, des modifications apportées aux classes concrètes permettent de définir la représentation

des informations de progression et de production, ainsi que le contenu des données additionnelles.
— La classe abstraite `MonitoringFragment` représente la vue de ce module. Cette classe déclare une méthode publique pour afficher de nouvelles informations de *monitoring* et une méthode abstraite pour initialiser l'interface de ce module. L'implémentation des classes concrètes revient à afficher des informations de *monitoring* reçues par une instance du framework.
— La classe abstraite `MonitoringPresenter` représente la présentation de ce module. Cette classe est chargée de créer des instances de `MonitoringTableRow` (lorsqu'un message MQTT demandant à un élève de faire une nouvelle tâche est envoyé) et de mettre à jour leurs contenus (lorsqu'un message MQTT contenant des informations de *monitoring* est reçu). L'implémentation des classes concrètes dépend alors du format des messages MQTT et de la représentation choisie pour afficher les données de *monitoring*.

4.5.8 *Le framework* CHAO *en quelques chiffres*

Pour conclure cette section décrivant l'implémentation technique du framework CHAO, nous proposons de fournir quelques données quantitatives détaillant son développement. Ces données sont reprises en SECTION 5.4 lors de l'évaluation du travail d'instanciation du framework.

La TABLE 4 propose un aperçu des objets qui composent le framework. Nous notons la déclaration de 37 classes (dont la répartition entre les couches du modèle MVP est donnée par la TABLE 5), qui donnent lieu à l'implémentation de 157 méthodes. Le nombre minimum de lignes pour une classe est atteint deux fois (par la classe `ServerHelper` et l'interface `IFragment`). Ce nombre minimum correspond à une classe dont l'implémentation n'a nécessité qu'une ligne de code. Le nombre maximum de lignes de code pour une classe est atteint pour la classe `OrchestrationTableRow`, qui est utilisée pour représenter les lignes de scénario et pour définir le comportement des boutons d'actions en bout de chaque ligne dans l'interface de *runtime scripting* (revoir la SECTION 4.5.6). Au total, le framework CHAO est constitué de 1688 lignes de code.

La TABLE 5 propose de fournir une description plus détaillée du développement du framework CHAO en distinguant quatre catégories d'éléments : les éléments métiers de la couche Modèle, les éléments utilitaires de la couche Modèle, les éléments de la couche Vue, et les éléments de la couche Présentation. Nous pouvons observer que la couche Vue représente la majorité du travail de développement du framework. Cela rend aussi compte des différents éléments graphiques qui ont dû être développés pour construire les interfaces utilisateurs

Type d'objet	Quantité
Classe	37
Méthode	157
Propriété, Attribut	109
Total de lignes de code	1688
Moyenne de lignes de code pour une classe	45.62
Lignes de code minimum pour une classe	4
Lignes de code maximum pour une classe	138

TABLE 4 – Données quantitatives relatives au développement du framework CHAO.

à partir des modèles d'interface présentés en SECTION 4.3. Nous notons par ailleurs la faible quantité de classes métiers appartement à la couche Modèle. Ceci s'explique par le fait que le framework ne fournit qu'une classe abstraite pour représenter les types de composant d'un scénario.

Éléments du modèle MVP	Nb de classes	Lignes de code	Moyenne de lignes de code	Lignes de code / Total
Modèle métier	4	154	38.50	9.12%
Modèle utilitaire	5	233	46.60	13.80%
Vue	20	1049	42.45	62.14%
Présentation	8	252	31.50	14.93%

TABLE 5 – Répartition des lignes de code en fonction des couches du modèle MVP.

4.6 PROCESSUS D'INSTANCIATION DU FRAMEWORK

Dans cette section, nous présentons le processus d'instanciation du framework.

Supposons qu'une application-élève soit identifiée pour l'intérêt des scénarios pédagogiques qu'elle permet de mettre en œuvre. L'instanciation du framework CHAO n'est possible que si l'informaticien peut adapter le code source de cette application-élève pour permettre les communications inter-tablettes et pour construire les messages envoyés et traiter les messages reçus. Dans ce cas, le travail d'instanciation du framework peut démarrer.

L'ordre dans lequel nous décrivons les étapes d'instanciation du framework peut constituer une ligne directrice dans le travail de l'informaticien. Cependant, le développement et l'adaptation d'un logiciel informatique n'est pas linéaire, et certaines démarches peuvent se faire en parallèle.

La suite de cette section est organisée de la façon suivante :
— SECTION 4.6.1 : Modélisation et instanciation des classes métiers liées au scénario pédagogique ;
— SECTION 4.6.2 : Modélisation et instanciation des classes métiers liées aux informations de *monitoring* ;
— SECTION 4.6.3 : Création d'une base de données et d'un service web ;
— SECTION 4.6.4 : Adaptation des classes utilitaires ;
— SECTION 4.6.5 : Modifications à apporter à l'application-élève.

4.6.1 *Modélisation et instanciation des classes métiers liées au scénario pédagogique*

La première étape consiste à modéliser le scénario pédagogique mis en œuvre par l'application-élève (comme l'illustre le diagramme de cas d'utilisation en FIGURE 9). Les résultats de cette étape doivent être :
— La liste des types de composant utilisés pour structurer le scénario pédagogique ;
— Pour chaque type de composant, la liste des composants disponibles.

À partir de ces résultats, le travail d'instanciation du framework consiste à :

1. Ajouter autant d'éléments à l'énumération `ComponentType` qu'il y a de types de composant ;
2. Déclarer et implémenter autant de classes héritant de `Component` qu'il y a de types de composant ;
3. Mettre à jour le fonctionnement de la fabrique `ComponentFactory` pour pouvoir instancier ces nouveaux objets.

4.6.2 *Modélisation et instanciation des classes métiers liées aux informations de monitoring*

Les informations de *monitoring* diffèrent en fonction des exercices réalisés par les élèves. L'instanciation du framework CHAO passe donc par la spécification du contenu et de l'affichage des informations de progression, de production, et additionnelles.

À partir de ces spécifications, l'informaticien a trois tâches liées à l'implémentation des classes concrètes décrites en SECTION 4.5.7.2 :

1. L'implémentation des classes `ProgressMonitoringTableRow` et `ProductionMonitoringTableRow`. Ces classes sont utilisées pour structurer les informations de *monitoring* en trois colonnes (« Acteur », « Informations de *monitoring* », et « Tâche »), et l'informaticien doit implémenter la représentation des informations de *monitoring* ;

2. L'implémentation des classes `ProgressMonitoringPresenter` et `ProductionMonitoringPresenter`. Ces classes sont utilisées pour créer des `MonitoringTableRow`, pour les envoyer aux vues du module de *monitoring*, et pour mettre à jour les informations de *monitoring* après la réception de nouveaux messages MQTT.
3. L'implémentation des classes `ProgressMonitoringFragment` et `ProductionMonitoringFragment`. Ces classes sont les vues du module de *monitoring* et sont utilisées pour afficher les `MonitoringTableRow` et les informations additionnelles dans une fenêtre pop-up.

4.6.3 *Création d'une base de données et d'un service web*

Une fois les objets en relation avec les classes métiers identifiés, l'informaticien peut schématiser et construire la base de données utilisée pour stocker les composants disponibles, les traces d'activités, et les lignes de scénario créées (comme mentionné en SECTION 4.4.1.3). Des fonctions et procédures stockées peuvent être développées pour effectuer des tâches récurrentes, comme sélectionner tous les éléments d'une table ou insérer de nouvelles traces d'activités.

Le développement du service web est nécessaire pour que le framework CHAO puisse communiquer avec la base de données. Pour cela, trois tâches sont nécessaires :

1. La configuration du service web pour communiquer avec la base de données ;
2. La création de pages web et donc de requêtes HTTP ;
3. La mise en forme des données récupérées de la base, en utilisant le format standard JSON.

4.6.4 *Adaptation des classes utilitaires*

En SECTION 4.5.3, nous avons identifié cinq classes utilitaires : `ServerHelper`, `MQTT`, `JsonParser`, `ComponentQuery`, et `JsonHelper`. Nous décrivons ci-dessous le travail d'adaptation de ces cinq classes :

1. Pour la classe `ServerHelper`, il s'agit de compléter l'attribut statique `IP_ADDRESS` en renseignant l'adresse IP de la machine hébergeant les serveurs HTTP, MQTT, et MySQL. Cet attribut est réutilisé par les autres classes utilitaires.
2. Pour la classe `MQTT`, il s'agit de spécifier les thèmes auxquels s'abonne l'enseignant et d'implémenter des méthodes de gestion et de traitements des messages reçus.
3. Pour la classe `JsonParser`, les seules modifications requises sont celles pour spécifier l'URL de l'API.

4. Pour la classe `ComponentQuery`, aucune modification n'est requise.
5. Pour la classe `JsonHelper`, il s'agit d'implémenter toutes les méthodes de traitement et de conversion d'objets Java en objets JSON, et vice-versa. Pour cela, la structure des objets JSON doit être définie par avance afin de faciliter le développement de ces méthodes.

4.6.5 *Modifications à apporter à l'application-élève*

Les modifications à apporter à l'application-élève sont de deux natures différentes.

Les modifications pour permettre la communication avec le framework correspondent à l'ajout d'un module de communication MQTT. Cela consiste à déclarer et implémenter des classes `MQTT` et `JsonHelper` similaires dans leurs rôles aux classes utilitaires implémentées dans le framework CHAO et portant les mêmes noms.

Les modifications liées aux communications correspondent à la création des messages à envoyer et au traitement des messages reçus. Dans le premier cas, il s'agit de synthétiser et d'envoyer des informations de progression et de production à partir du travail d'un élève utilisant l'application-élève. Dans le second cas, il s'agit de rendre l'application-élève suffisamment flexible pour, par exemple, pouvoir changer la tâche d'un élève en temps réel, modifier un paramètre de rétroaction, ou encore mettre à disposition des aides supplémentaires.

5 ÉVALUATION DES TRAVAUX

Contenu du chapitre

5.1 INTRODUCTION

Le travail de cette thèse comporte différents aspects. Le premier aspect concerne la conception d'un framework servant de base pour le développement de technologies d'orchestration. Le second aspect concerne la conception de modules et d'interfaces utilisateurs pour le support de l'orchestration d'un scénario pédagogique.

Dans ce chapitre, nous présentons les travaux que nous avons mené pour évaluer cette thèse sur ces deux aspects. Pour cela, nous allons d'abord détailler l'évaluation des modules de *monitoring* puis de *runtime scripting* en décrivant les instances du framework utilisées au cours de ces travaux. Ces descriptions facilitent ensuite la compréhension de l'évaluation du framework.

Deux modules du framework n'ont pas fait l'objet d'évaluation dans cette thèse : le module de gestion des composants et le module de *primo-scripting*.

En ce qui concerne le module de gestion des composants, l'interface utilisateur n'a pas fait l'objet d'un travail approfondi de conceptualisation et de modélisation. Nous justifions cette décision par le

fait que cette interface propose de réaliser une action simple et ne donnant pas lieu à une problématique de recherche.

En ce qui concerne le module de *primo-scripting*, nous rappelons ici que nous nous sommes appuyés sur les résultats de Sobreira concernant l'éditeur ediT2 (Sobreira, 2014), montrant qu'une représentation en table est facile à utiliser et permet de concevoir un grand nombre de scénarios différents. Par conséquent, nous n'avons pas mené d'expérimentation complémentaire à ce sujet.

5.2 ÉVALUATION DU MODULE DE MONITORING

5.2.1 *Description des interfaces de monitoring de* CHAO-*Dictée négociée*

Nous avons évalué les outils de *monitoring* en utilisant l'instance CHAO-Dictée négociée. Ces outils sont offerts à travers des interfaces utilisateurs que nous avons instanciées à partir du modèle d'interface de *monitoring* présenté en SECTION 4.3.3. Pour rappel, ce modèle d'interface de *monitoring* présente les informations de *monitoring* sous forme d'un tableau à trois colonnes : « Acteur », « Informations de *monitoring* », et « Tâche en cours ». Nous avons donc identifié le contenu et la structure des informations de *monitoring* durant les phases individuelle et collective de la dictée négociée. C'est-à-dire que pour chacune de ces phases, nous avons défini des informations permettant de rendre compte de la progression et de la production de chacun des acteurs (élèves ou groupes d'élèves), et des informations additionnelles décrivant le travail des acteurs. Ces informations sont répertoriées dans la TABLE 6.

Informations de *monitoring*	Phase individuelle	Phase collective
Informations de progression	Barres de progression représentant le rapport entre le nombre de caractères saisis et le nombre total de caractères dans la dictée.	Tableau de progression représentant les mots déjà négociés, en cours de négociation, et à négocier.
Informations de production	Textes des élèves saisis et affichés en temps réel.	Orthographes des mots négociés ou à négocier.
Informations additionnelles	Durée de la phase individuelle ; Nombre de corrections ; Nombre d'écoutes par piste.	Justifications des mots négociés.

TABLE 6 – Synthèse des informations de *monitoring* utilisées lors des expérimentations avec CHAO-Dictée négociée.

Pour illustrer ces informations, les figures ci-après présentent les interfaces de *monitoring* de l'instance CHAO-Dictée négociée lors de la

phase individuelle et de la phase collective. Comme expliqué précédemment, la conception du framework s'est appuyé sur l'application-élève de dictée négociée. C'est pourquoi les copies d'écran présentent une structure similaire à celle du modèle d'interface de *monitoring* mais avec une terminologie ayant évolué par la suite.

Les FIGURES 39 et 40 illustrent les interfaces de *monitoring* de la phase individuelle. Les élèves réalisent la tâche de dictée individuelle, et les informations de progression sont synthétisées par des barres de progression et les informations de production affichent le texte de la dictée en cours d'écriture pour chaque élève.

Les FIGURES 41 et 42 illustrent les interfaces de *monitoring* de la phase collective. Les groupes d'élèves réalisent la tâche de négociation et de justification des mots à négocier. Ces mots à négocier sont présentés dans les zones 1 des deux figures. En ce qui concerne les informations de progression, une cellule rouge représente un mot pas encore travaillé, une cellule orange représente un mot dont les orthographes ont été votées, et une cellule verte représente un mot dont les orthographes ont été justifiées. En ce qui concerne les informations de production, un mot en rouge représente l'orthographe du mot écrit tel quel dans une dictée d'un élève, un mot en orange représente l'orthographe d'un mot voté, et un mot en vert représente l'orthographe d'un mot justifié.

Les FIGURES 43 et 44 présentent les informations additionnelles pour les phases individuelle et collective. En ce qui concerne les données additionnelles de la phase individuelle, l'instance CHAO-Dictée négociée propose de dénombrer la réalisation de certaines actions d'un élève et de présenter des informations sur la durée de la phase individuelle. En ce qui concerne les données additionnelles de la phase collective, les justifications des orthographes des mots à négocier sont affichées.

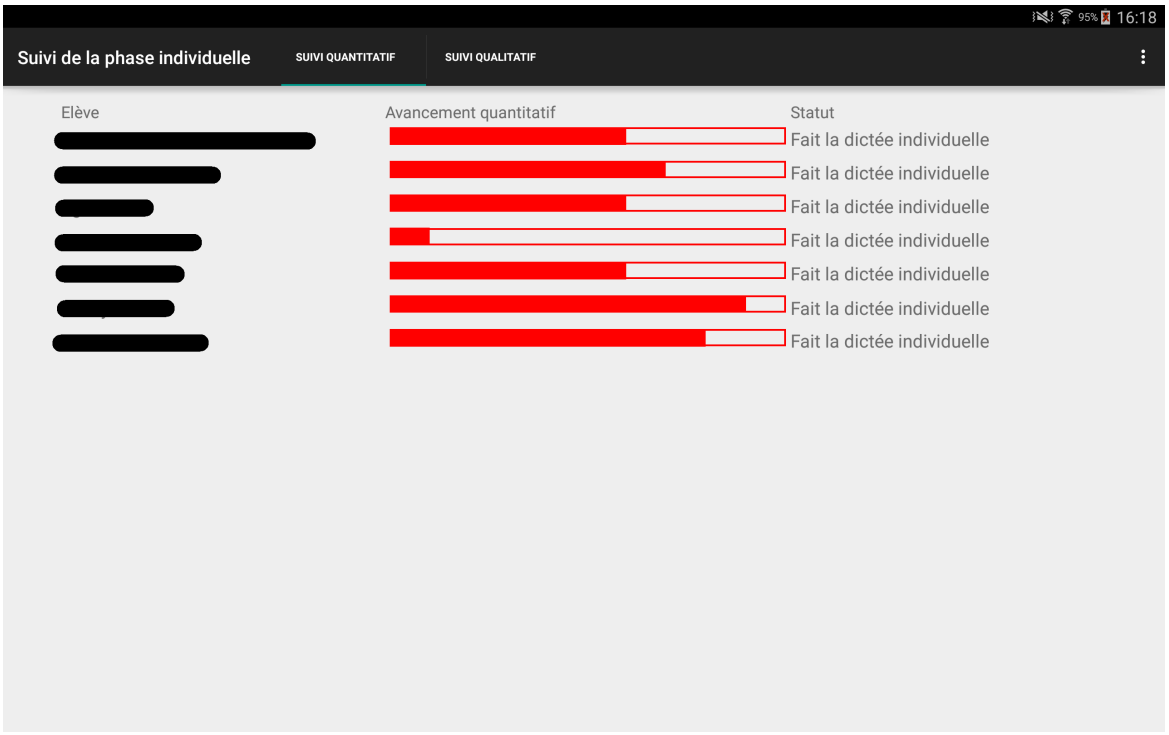


FIGURE 39 – Informations de progression de la phase individuelle présentées par CHAO-Dictée négociée.

FIGURE 40 – Informations de production de la phase individuelle présentées par CHAO-Dictée négociée.

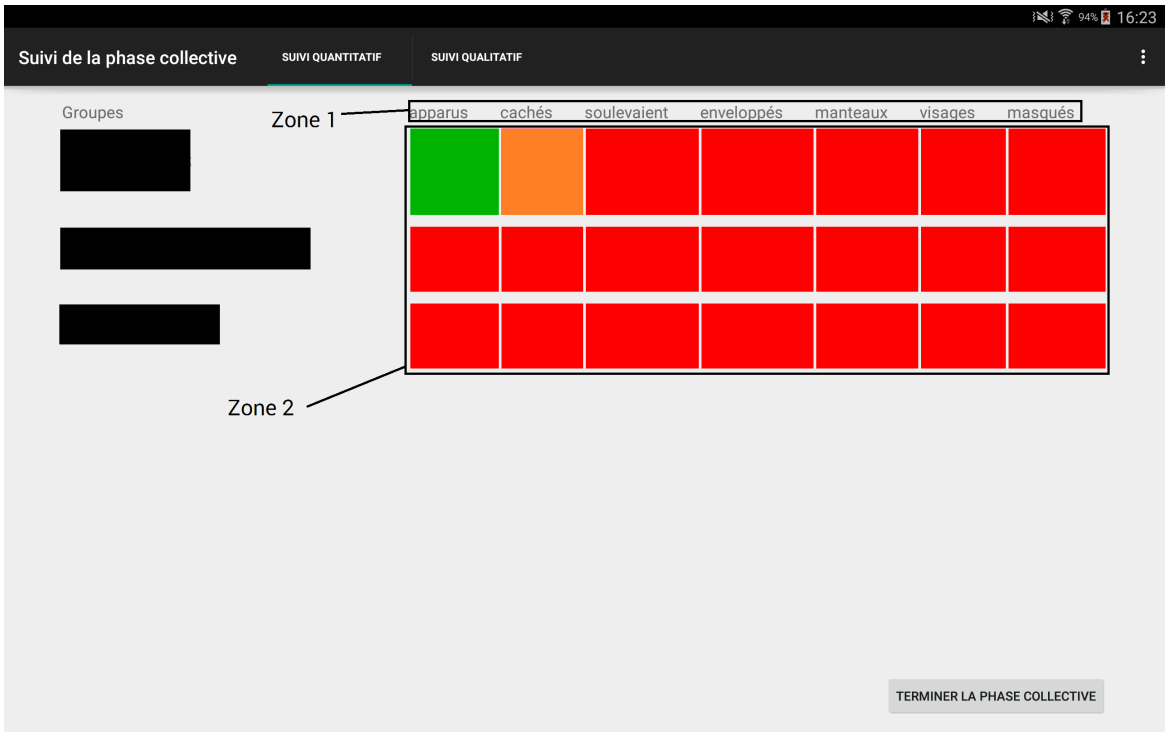


FIGURE 41 – Informations de progression de la phase collective présentées par CHAO-Dictée négociée.

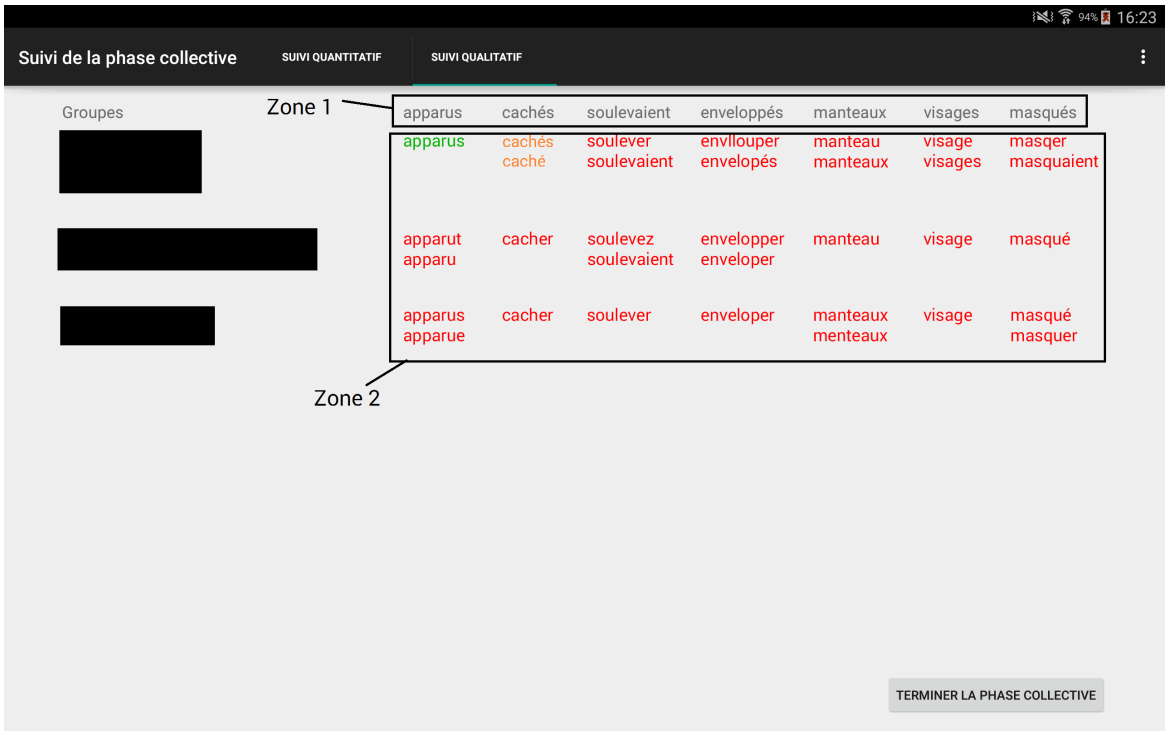


FIGURE 42 – Informations de production de la phase collective présentées par CHAO-Dictée négociée.

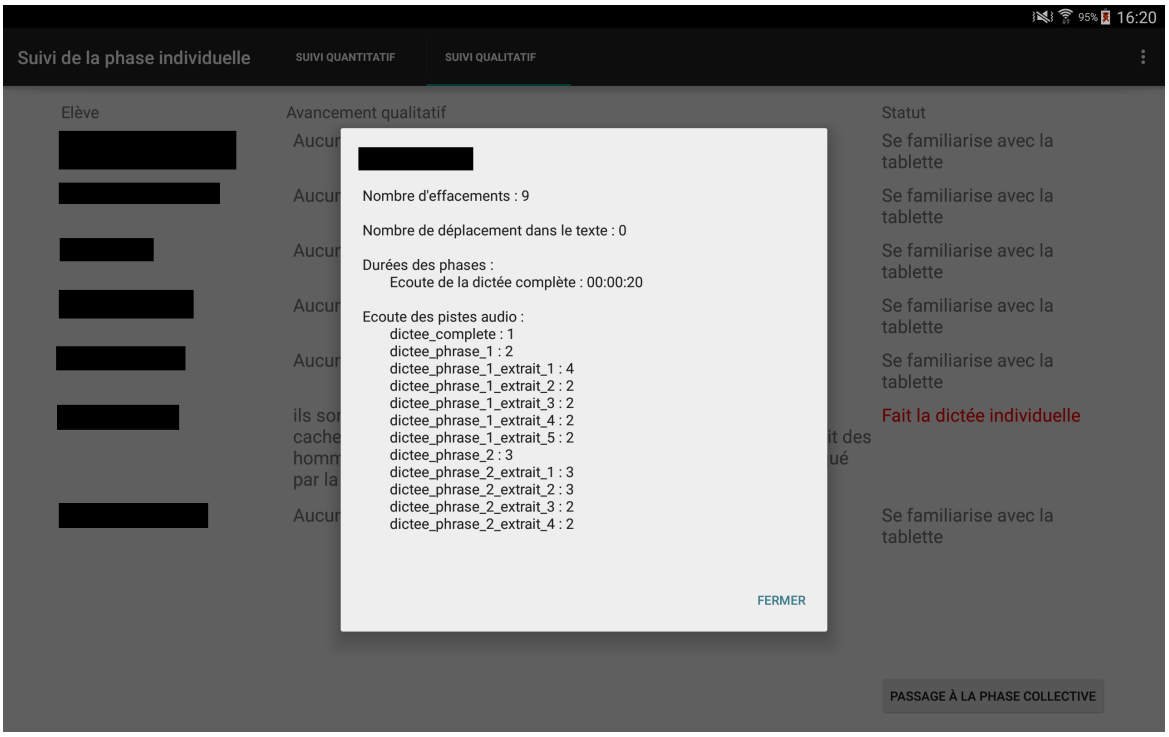


FIGURE 43 – Informations additionnelles de la phase individuelle présentées par CHAO-Dictée négociée.

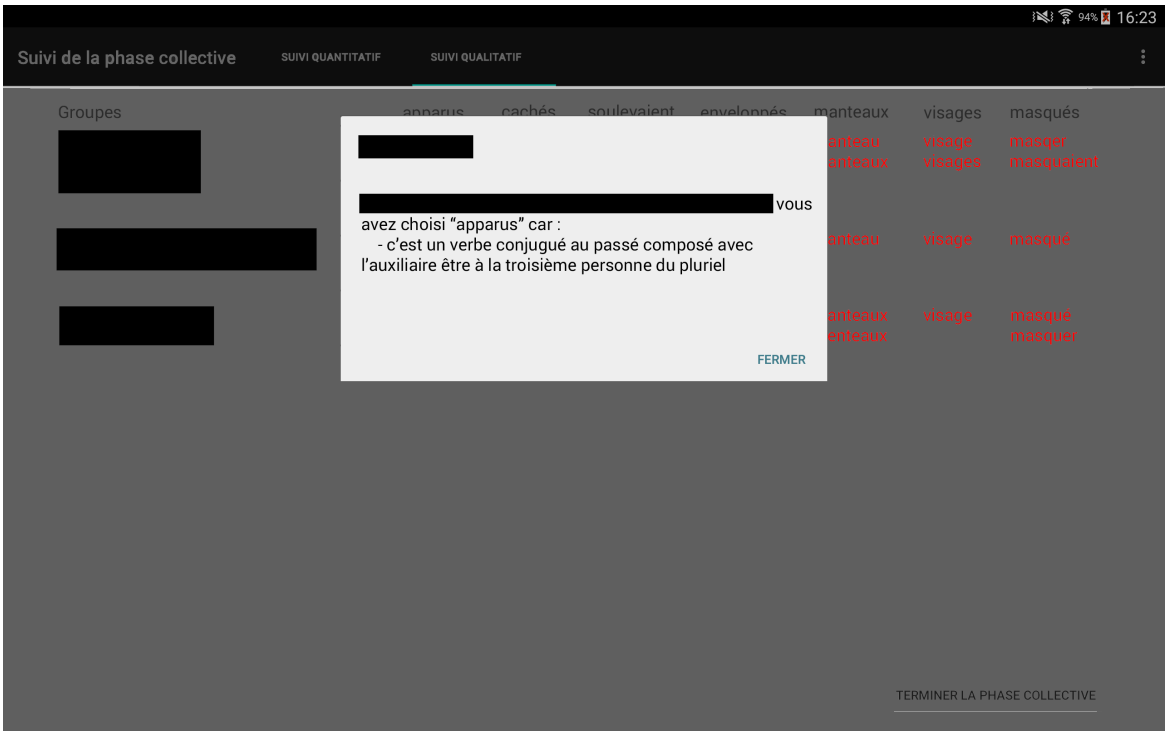


FIGURE 44 – Informations additionnelles de la phase collective présentées par CHAO-Dictée négociée.

5.2.2 *Méthodologie*

Les évaluations que nous avons menées ont cherché à déterminer l'utilité des informations de *monitoring* présentées à un enseignant et l'efficacité des interfaces pour supporter le suivi d'un scénario mis en œuvre par l'application-élève de dictée négociée. À ce titre, nous avions émis les deux hypothèses expérimentales suivantes :

(H1) Les informations de progression, de production, et additionnelles sont utiles pour le suivi de la dictée négociée.

(H2) Les interfaces de *monitoring*, en fournissant ces informations, supportent efficacement un enseignant dans le suivi de la dictée négociée.

Pour vérifier ces hypothèses, nous avons mené quatre expérimentations dans deux classes de CM1 (avec respectivement 27 et 29 élèves) et deux classes de CM2 (avec respectivement 20 et 24 élèves), et avec donc quatre enseignantes de primaire. Nous avons demandé à ces quatre enseignantes d'orchestrer une dictée négociée (le texte de la dictée est resté le même lors de chaque expérimentation), en se concentrant sur le suivi de cette activité. Le protocole expérimental est fourni en Table 7.

Nous avons conduit deux sessions lors de chaque expérimentation, en divisant la classe en deux : une moitié de classe travaille sur la dictée négociée quand l'autre travaille de façon autonome, et vice-versa. Nous avons ainsi laissé l'opportunité aux enseignantes de se servir de l'application Chao-Dictée négociée lors de deux séances successives.

5.2.3 *Résultats*

5.2.3.1 *Utilité des informations de monitoring*

La première évaluation que nous avons menée concerne l'utilité des informations de *monitoring*. Ces informations peuvent être retrouvées dans la Table 6.

Dans le questionnaire que nous faisons passer aux enseignantes en fin d'expérimentation, une partie portait sur l'utilité de chaque type d'informations de *monitoring*. L'échelle de Likert utilisée pour ce questionnaire comportait quatre réponses possibles : « Pas du tout d'accord », « Pas d'accord », « D'accord », et « Tout à fait d'accord ». La synthèse des réponses est fournie par la Figure 45 ci-après.

En analysant ces quatre diagrammes radars, nous constatons que les quatre enseignantes ont jugé que les informations de production sont utiles aussi bien en phase individuelle qu'en phase collective. Les appréciations par rapport aux informations de progression sont à la fois moins favorables et moins homogènes. Nous expliquons ces variations par des stratégies de suivi différentes utilisées par les en-

Phase	Actions
Préparation de la dictée négociée	Une fois le matériel installé, nous expliquons à l'enseignant et aux élèves le déroulement de l'expérimentation et le fonctionnement général des tablettes. Nous assistons l'enseignant dans la création du scénario pédagogique qui consiste essentiellement à sélectionner les élèves participant à la dictée négociée, cette étape ne faisant pas l'objet principal de l'expérimentation.
Phase individuelle	L'enseignant se focalise sur le suivi de ses élèves en utilisant les outils et informations conçus pour la phase individuelle. Il intervient auprès des élèves en allant directement les voir lorsque nécessaire (pour par exemple demander à un élève de bien relire sa dictée). Il décide aussi de quand se termine cette phase (que ce soit lorsque tous les élèves ont fini d'écrire la dictée, ou lorsqu'il estime que cette phase a duré suffisamment longtemps).
Préparation de la phase collective	Cette étape consiste à créer des groupes pour la phase collective. De la même façon que durant la préparation de l'activité, nous aidons l'enseignant à former les groupes pour nous assurer du bon déroulement des manipulations réalisées.
Phase collective	L'enseignant se focalise sur le suivi des groupes en utilisant les outils et informations conçus pour la phase collective. Il intervient auprès des groupes en allant directement les voir lorsque nécessaire (pour par exemple inciter les élèves d'un groupe à construire des explications basées sur leurs connaissances des règles de grammaire et de conjugaison). Il décide aussi de quand se termine cette phase (que ce soit lorsque tous les groupes ont fini de négocier tous les mots, ou lorsqu'il estime que cette phase a duré suffisamment longtemps).
Fin de la dictée négociée	L'enseignant remplit un questionnaire (utilisant une échelle de Likert) dont les questions peuvent être regroupées en trois catégories : une première catégorie, d'ordre général, porte sur l'utilisation de tablettes par l'enseignant et l'intérêt de la dictée négociée ; une seconde catégorie s'intéresse à l'utilité des informations de progression, de production, et additionnelles de *monitoring* pour le suivi de la phase de dictée négociée ; une dernière catégorie s'intéresse à l'efficacité des interfaces de *monitoring* pour le suivi de la dictée négociée. Enfin, un entretien de fin de séance avec l'enseignant nous permet de recueillir ses impressions à chaud.

TABLE 7 – Description du protocole expérimental mis en œuvre pour évaluer le module de *monitoring*.

seignantes. En effet, l'enseignante 3 a préféré se focaliser sur les productions, tandis que l'enseignante 1 a profité de l'ensemble des informations proposées pour suivre l'activité de ses élèves.

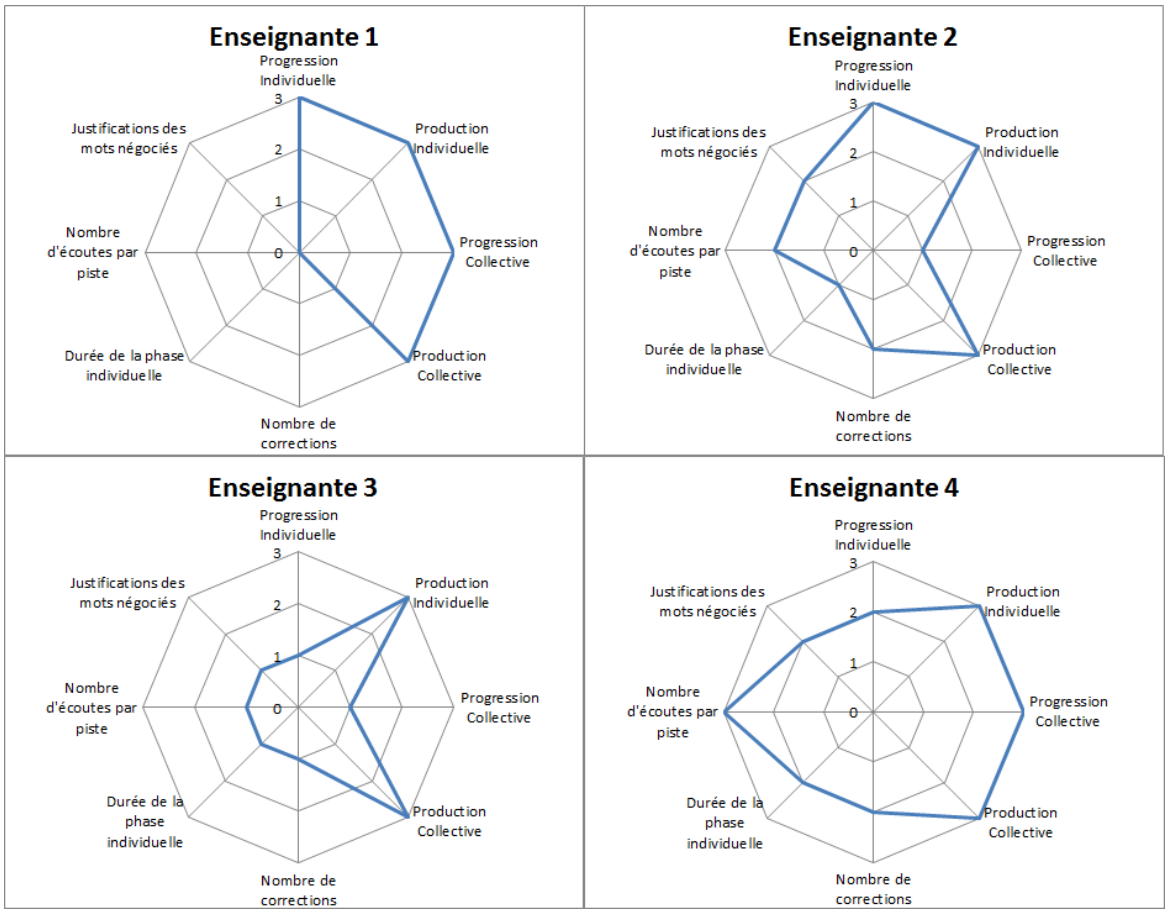


FIGURE 45 – Appréciations de l'utilité des informations de *monitoring*. La valeur 0 correspond à la réponse « Pas du tout d'accord », 3 à « Tout à fait d'accord ».

Les réponses concernant les données additionnelles sont quant à elles beaucoup plus hétérogènes. L'enseignante 1 n'a par exemple jamais consulté ces informations additionnelles, ce qui s'est traduit par des avis très défavorables par rapport à ces informations. L'enseignante 3 présente un profil similaire en adjugeant une note de 1/3 pour chaque type d'information additionnelle. Enfin, les enseignantes 2 et 4 semblent trouver utiles le nombre d'écoutes par piste et le nombre de correction pour la phase individuelle, et les justifications produites pour la phase collective. L'hétérogénéité de ces réponses laisse supposer que les informations additionnelles sont plus ou moins utiles, et que les enseignants peuvent s'en passer. Ce résultat ne nous surprend pas puisqu'il reflète la volonté des enseignants d'observer en priorité le suivi de la production et de la progression de la dictée négociée.

5.2.3.2 *Efficacité des interfaces de monitoring*

Pour évaluer l'efficacité des interfaces de *monitoring*, nous avons demandé aux enseignantes de répondre au questionnaire décrit en TABLE 8. Pour pallier le fait de n'avoir que quatre enseignantes impliquées dans les expérimentations, nous avons construit ce questionnaire de façon à ce que chaque enseignante fournisse un jugement quant au suivi de la progression (respectivement de la production) de chaque élève et groupe. Pour l'ensemble des quatre classes, cela représente un total de 100 élèves et de 36 groupes (les groupes étant composés de deux ou trois élèves).

Question	Pas du tout d'accord	Pas d'accord	D'accord	Tout à fait d'accord
Q1 : Les outils vous ont permis de suivre de façon satisfaisante la progression de [élève].				
Q2 : Les outils vous ont permis de suivre de façon satisfaisante la production de [élève].				
Q3 : Les outils vous ont permis de suivre de façon satisfaisante la progression de [groupe].				
Q4 : Les outils vous ont permis de suivre de façon satisfaisante la production de [groupe].				

TABLE 8 – Questionnaire d'évaluation de l'efficacité des interfaces de *monitoring*. L'enseignante doit répondre à ces questions pour chaque élève et chaque groupe de sa classe.

Dans la TABLE 9, nous présentons les réponses à ce questionnaire en reprenant les codes utilisés pour identifier les quatre questions différentes, et en précisant le nombre de réponses (qui reflète aussi un nombre d'élèves ou de groupes d'élèves) pour chaque valeur de l'échelle de Likert.

Les résultats de ce questionnaire présentent un avis global positif en faveur des interfaces de *monitoring*. De ces résultats, nous en tirons deux conclusions. La première est que les interfaces de *monitoring* ont supporté efficacement les enseignantes dans leurs tâches de suivi de la dictée négociée. En effet, les réponses suggèrent que le suivi des progressions et des productions de chaque élève et groupe a été satisfaisant. La deuxième conclusion est que présenter deux interfaces séparées ne semble pas gêner les enseignantes dans la réalisation du suivi de la dictée négociée.

5.2.4 *Discussion*

Les résultats de ces expérimentations montrent que, malgré des pratiques différentes, les quatre enseignantes ont estimé que les informations de progression et de production étaient utiles et que les

Question	# Enseignante	Pas du tout d'accord	Pas d'accord	D'accord	Tout à fait d'accord
Q1	Enseignante 1				29
	Enseignante 2				27
	Enseignante 3		1		19
	Enseignante 4				24
Q2	Enseignante 1				29
	Enseignante 2				27
	Enseignante 3				20
	Enseignante 4				24
Q3	Enseignante 1			10	
	Enseignante 2				10
	Enseignante 3				8
	Enseignante 4				8
Q4	Enseignante 1				10
	Enseignante 2				10
	Enseignante 3				8
	Enseignante 4				8

TABLE 9 – Réponses au questionnaire d'évaluation portant sur l'efficacité des interfaces de *monitoring*.

interfaces de *monitoring* proposées les supportaient efficacement dans leur activité de suivi de la dictée négociée. Cela suggère que les hypothèses énoncées en SECTION 5.2.2 sont valides, avec un bémol par rapport à l'utilité des informations additionnelles.

Quatre enseignantes seulement ont participé à ces expérimentations. Il est donc difficile de conclure définitivement quant à l'efficacité des interfaces de *monitoring* proposées. La multiplication des questions quant à la satisfaction du suivi de la progression et de la production de chaque acteur a été pensée pour pallier ce défaut, mais l'échantillon d'utilisateurs reste faible. Malgré cela, les réponses montrent que les enseignantes étaient toutes satisfaites des outils de suivis proposés.

Durant ces expérimentations, nous n'évaluons les interfaces de *monitoring* que dans le cadre d'une instance : CHAO-Dictée négociée. Les conclusions tirées quant à l'utilité des différentes informations de *monitoring* sont donc relatives à ce type d'exercice. L'instanciation du framework pour supporter l'orchestration d'un autre type de scénario peut alors fournir des compléments de réponse quant à l'utilité des informations de *monitoring* et quant à l'efficacité des interfaces proposées. Sur ce point, et bien que cela n'était pas le sujet des évaluations, les expérimentations menées avec CHAO-SimBûchettes montrent que

l'enseignante a été en mesure de suivre de façon satisfaisante le travail de ses élèves.

5.3 ÉVALUATION DU MODULE DE RUNTIME SCRIPTING

5.3.1 *Description de l'interface de runtime scripting de* CHAO-*SimBûchettes*

Les outils de *runtime scripting* ont été évalués en utilisant l'instance CHAO-SimBûchettes.

L'application SimBûchettes met en œuvre des scénarios pédagogiques dans lesquels des élèves font des exercices mettant en jeu leurs connaissances du principe décimal et du principe de position.

Les actions de *runtime scripting* d'un enseignant peuvent prendre deux formes : (1) après avoir construit un scénario pédagogique, un enseignant peut entreprendre d'effectuer des modifications sur ce scénario pendant son déroulement en changeant l'enchaînement des exercices pour un élève ; (2) un enseignant peut décider de modifier les paramètres de rétroaction d'un exercice pour un élève. Ces paramètres de rétroaction sont au nombre de six :

- Groupement contraint : l'application n'autorise pas les regroupements ne comportant pas exactement dix objets identiques (par exemple, il n'est pas possible de créer un groupement de dix bûchettes – représentant une dizaine – avec seulement neuf bûchettes).
- Groupement contraint et message : même chose que précédemment, et affiche un message d'information à l'élève.
- Principe décimal contraint : l'application n'autorise pas de déposer plus de neuf objets dans une boîte (par exemple, déposer 10 bûchettes dans la boîte des unités).
- Principe décimal contraint et message : même chose que précédemment, et affiche un message d'information à l'élève.
- Principe de position contraint : l'application n'autorise pas de déposer un objet dans une boîte ne lui correspondant pas (par exemple, déposer une bûchette dans la boîte des dizaines).
- Principe de position contraint et message : même chose que précédemment, et affiche un message d'information à l'élève.

Le modèle d'orchestration que nous avons conçu et présenté en SECTION 4.2.3 spécifie qu'un enseignant peut réaliser des actions de *runtime scripting* en modifiant des lignes de scénario. Ainsi, et en instanciant le modèle d'interface de *runtime scripting* détaillé en SECTION 4.3.2, l'application CHAO-SimBûchettes offre à l'enseignant l'interface de table d'orchestration illustrée en FIGURE 46.

En utilisant cette table d'orchestration, un enseignant peut ajouter ou retirer un composant d'une ligne de scénario. Pour rendre ces changements effectifs, il suffit ensuite de cliquer sur le bouton « Démarrer », qui correspond au premier bouton d'action. Les autres bou-

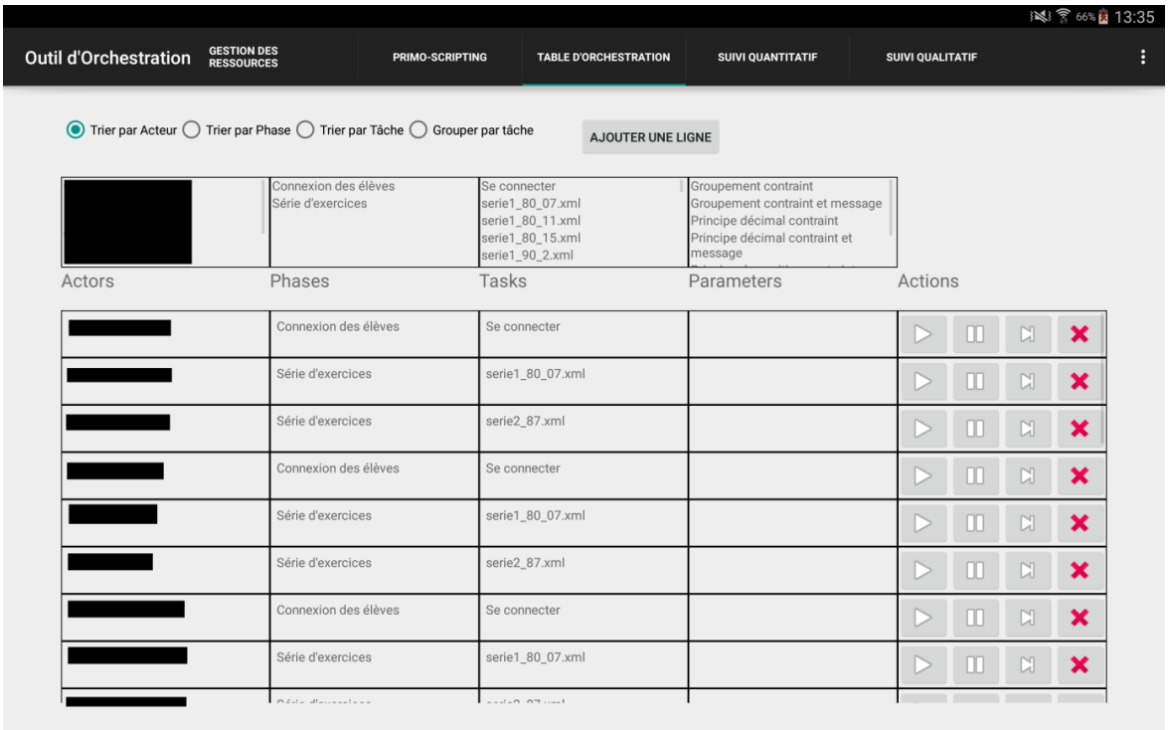


FIGURE 46 – La table d'orchestration permet de modifier le scénario en changeant l'exercice en cours ou en modifiant les paramètres de rétroaction des exercices.

tons d'actions (de la gauche vers la droite) ont pour effet de mettre en pause la tablette de l'acteur concerné, de remettre en marche la tablette de l'acteur concerné, et de supprimer définitivement la ligne de scénario de l'écran.

Pour cette instance, nous avons aussi implémenté des fonctionnalités de tri de table et de création de lignes de scénario vierges. Ces fonctionnalités ne sont pas fournies par le framework et peuvent être considérées comme des fonctionnalités supplémentaires constituant une piste d'évolution pour des travaux futurs. Nous détaillons cet aspect en SECTION 6.3.

5.3.2 *Méthodologie*

Nous avons mené une expérimentation visant à évaluer l'efficacité du module de *runtime scripting*, c'est-à-dire, sa faculté à permettre à un enseignant de réaliser des actions de *runtime scripting*. Ces actions concernent l'ajout ou le retrait de composants dans des lignes de scénario et la création de lignes de scénario intermédiaires. Pour cela, nous avons émis l'hypothèse expérimentale suivante :

(H1) La table d'orchestration permet à un enseignant de modifier efficacement un scénario pédagogique pendant son déroulement.

Pour valider cette hypothèse, nous avons conduit, sur deux jours, deux expérimentations. Ces expérimentations ont impliqué une enseignante n'ayant auparavant jamais utilisé l'instance CHAO-SimBûchettes (ni CHAO-Dictée négociée d'ailleurs) et sa classe de CE1 composée de 27 élèves. Le protocole expérimental utilisé lors de ces expérimentations est détaillé par la TABLE 10.

Lors de la première expérimentation, la classe était divisée en quatre groupes, et chaque groupe a tour à tour utilisé l'application SimBûchettes pendant environ 25 minutes. Pour chacun des élèves, une succession de huit exercices avait été définie au préalable, avec un niveau de difficulté croissant après chaque exercice.

Lors de la seconde expérimentation, seulement cinq élèves ont utilisé l'application SimBûchettes. Cette expérimentation a été menée lors d'une séance d'activité pédagogique complémentaire, et avait pour objectif de retravailler certains aspects de la numération décimale avec des élèves en difficulté. Pour cette séance, chaque élève s'est vu attribué une succession de six exercices dont un dernier exercice de contrôle.

Phase	Actions
Introduction	Nous présentons l'application CHAO-SimBûchettes à l'enseignante, en lui expliquant le fonctionnement des outils d'orchestration proposés.
Construction du scénario pédagogique	En suivant les consignes de l'enseignante, nous construisons le scénario pédagogique. Dans cette expérimentation, il s'agit de spécifier des enchaînements d'exercices pour chaque élève.
Mise en œuvre et gestion du scénario	L'enseignante démarre les activités pour les élèves, et gère le déroulement du scénario en utilisant les outils de *monitoring* et de *runtime scripting*. Nous nous intéressons particulièrement à l'utilisation des outils de *runtime scripting*, qui permettent à l'enseignante de (1) modifier l'enchaînement des exercices pour un élève, et (2) modifier les règles de rétroaction d'un exercice pour un élève.

TABLE 10 – Description du protocole expérimental mis en œuvre pour évaluer le module de *runtime scripting*.

Afin d'identifier les actions de *runtime scripting* durant les expérimentations, nous avons recensé toutes les fois que l'enseignante a cliqué sur le premier bouton (c'est-à-dire sur le bouton « Démarrer »). L'actionnement de ce bouton provoque la génération d'un message MQTT envoyé aux élèves qui entraîne soit le démarrage d'une tâche spécifiée par le scénario, soit une modification du scénario (qui caractérise les actions de *runtime scripting*). Les FIGURES 47 et 48 montrent des exemples de messages MQTT envoyés par l'instance CHAO-SimBûchettes.

Les raisons du message de connexion sont simples : il s'agit de demander aux élèves de s'identifier pour apparier un élève avec sa tablette. Les intentions du message de démarrage peuvent être mul-

```
{
    "contentType": "CONNEXION",
    "eleves": [
        {
            "idEleve" : "1",
            "notionName" : "Prenom et Nom"
        },
        {
            "idEleve" : "2",
            "notionName" : "Prenom et Nom"
        },
        {
            "idEleve" : "3",
            "notionName" : "Prenom et Nom"
        }
    ]
}
```

FIGURE 47 – Un exemple de message MQTT de connexion.

```
{
    "contentType": "START_DEFAULT",
    "idEleve": "11",
    "code_exercice": "serie1_90_2.xml",
    "contraintes": [
           "Principe décimal contraint"
       ]
}
```

FIGURE 48 – Un exemple de message MQTT de démarrage d'un exercice.

tiples et dépendent du contexte et du déroulement du scénario. En effet, ce même message peut avoir quatre significations différentes :

- Le démarrage de l'exercice suivant, comme spécifié par le scénario pédagogique. Cela revient à démarrer les tâches des lignes de scénario en reprenant l'ordre dans lequel ces lignes ont été créées.
- Le démarrage d'un exercice spécifié par le scénario pédagogique mais qui *saute* un exercice qui aurait dû être démarré avant. Cela peut être fait en démarrant une tâche d'une ligne de scénario alors que la tâche de la ligne précédente n'a pas été démarrée.
- La modification d'un exercice spécifié par le scénario pédagogique. Cela est fait en modifiant les règles de rétroaction d'un exercice ou en proposant un exercice similaire (c'est-à-dire qui met en jeu le même type de connaissances des principes de position et décimal) mais avec des données d'énoncé différentes.

— Le démarrage d'un exercice qui n'est pas spécifié par le scénario pédagogique. Cela revient à créer, en cours de session, une nouvelle ligne de scénario et de démarrer la tâche décrite par cette ligne.

Nous avons alors cinq catégories de messages MQTT envoyés par un enseignant, dont les trois dernières témoignent de la réalisation d'action de *runtime scripting* :

1. Les messages de demande de connexion ;
2. Les messages de démarrage d'un exercice du scénario ;
3. Les messages contribuant à ignorer un exercice du scénario ;
4. Les messages modifiant un exercice du scénario ;
5. Les messages impliquant l'ajout d'un exercice au scénario.

5.3.3 *Résultats*

5.3.3.1 *Efficacité des outils de runtime scripting*

Dans cette évaluation, nous avons classé l'ensemble des messages MQTT envoyés par l'enseignante dans les cinq catégories de messages mentionnées auparavant. Ce travail est synthétisé dans la TABLE 11. Puis, nous avons analysé ces données afin de juger de l'efficacité des outils de *runtime scripting* pour modifier un scénario pédagogique en cours de session.

Type de message	Première séance	Deuxième séance	Total
1. Messages de demande de connexion des élèves	6	5	11
2. Messages de démarrage d'un exercice du scénario	63	30	93
3. Messages contribuant à ignorer un exercice spécifié par le scénario	36	0	36
4. Messages modifiant un exercice du scénario	40	0	40
5. Messages impliquant l'ajout d'un exercice au scénario	39	0	39
Total	184	35	219

TABLE 11 – Détail des messages envoyés par l'enseignante au cours des expérimentations.

Les messages de type 3, 4, et 5 témoignent d'actions de *runtime scripting*. C'est au cours de la première expérimentation que l'enseignante a eu à modifier le scénario pédagogique simultanément à son déroulement. C'est pourquoi nous allons nous focaliser uniquement sur celle-ci.

Pour bien comprendre les résultats de ces données, nous allons décrire succinctement le déroulement de la première expérimentation. L'enseignante avait préparé des *parcours* qui correspondent à des enchaînements de huit exercices. En fonction de l'aisance de l'élève par

rapport à la numération décimale, ce-dernier se voyait attribué un parcours plus ou moins difficile. L'enseignante, en affectant un parcours à chaque élève, a ainsi conçu un scénario pédagogique en amont de l'expérimentation. L'expérimentation s'est déroulée sur quatre sessions durant lesquelles un quart de la classe était tour à tour impliqué. Lors de la première session, l'enseignante s'est aperçue que les exercices préparés étaient trop simples pour ses élèves, et ces-derniers les finissaient trop rapidement. Les actions de *runtime scripting* menées par l'enseignante au cours des quatre sessions ont donc concerné des modifications importantes du scénario pour proposer des exercices plus complexes à ses élèves.

Les données font état de 115 modifications de scénario, soit 62.5% du total des instructions pour la première expérimentation. Les modifications ont porté sur la modification de l'enchaînement des exercices et sur la modification des données de l'énoncé des exercices. Nous avons cependant constaté que l'enseignante n'a jamais modifié les règles de rétroaction des exercices. Ceci s'explique, à notre sens, par le fait que l'enseignante a préféré se déplacer pour voir et aider ses élèves en leur rappelant les principes de la numération décimale.

L'enseignante a ainsi pu attribuer aux élèves des exercices qui n'étaient *a priori* pas spécifiés par le scénario pédagogique. Cet aspect semble montrer qu'elle a été en mesure d'adapter le scénario pédagogique autant de fois que nécessaire. Pour ces raisons, nous estimons que les outils de *runtime scripting* ont permis de modifier de façon efficace un scénario pédagogique.

5.3.3.2 *Utilisabilité de l'interface de table d'orchestration*

Nous avons aussi cherché à obtenir l'appréciation de l'enseignante quant à l'utilisabilité de la table d'orchestration pour effectuer ses actions de *runtime scripting*. Pour cela, nous lui avons proposé de remplir un questionnaire après la première expérimentation (c'est-à-dire, celle durant laquelle tous les élèves ont participé). Ce questionnaire utilise une échelle de Likert avec sept options [1] et la TABLE 12 liste les questions posées et les réponses de l'enseignante.

Les quatre premières questions portent sur l'utilité des trois premiers boutons d'actions et des messages MQTT envoyés en les actionnant. Sur ces points, les réponses montrent que l'enseignante est globalement satisfaite. Elle a pourtant indiqué son intérêt pour avoir d'autres types de messages à envoyer. En l'interrogeant sur cette réponse, il nous est apparu qu'elle désirait en fait avoir un fonctionnement différent d'envoi de messages. En particulier, elle aurait souhaité que les envois de messages de démarrage de nouvel exercice soient automatisés lorsqu'un élève a fini correctement un exercice du

1. Les options étant (1 : Pas du tout d'accord, 2 : Pas d'accord, 3 : Plutôt pas d'accord, 4 : Sans avis, 5 : Plutôt d'accord, 6 : D'accord, 7 : Tout à fait d'accord).

Question	Réponses
1. Avoir plusieurs types de messages à envoyer aux élèves est utile.	D'accord
2. Les différents messages sont utiles.	Tout à fait d'accord
3. J'aurais aimé avoir d'autres messages à envoyer aux élèves.	Plutôt d'accord
4. Les modifications des exercices correspondent aux modifications attendues.	Sans avis
5. Avoir des fonctions pour trier la table est utile.	D'accord
6. Les fonctions de tri de la table sont utiles.	Plutôt d'accord
7. J'aurais aimé avoir d'autres fonctions de tri de table.	Plutôt d'accord
8. La table est facile à lire.	Plutôt d'accord
9. Le déplacement des éléments dans et en dehors de la table est facile.	Sans avis
10. La création d'une nouvelle ligne dans la table d'orchestration est facile.	D'accord
11. La suppression d'une ligne dans la table d'orchestration est facile.	Sans avis

TABLE 12 – Questionnaire et réponses de l'enseignante concernant l'utilisabilité de la table d'orchestration.

scénario (c'est-à-dire un message de type 2). En effet, cette fonctionnalité aurait pu simplifier le travail d'orchestration de l'enseignante. Cette remarque conduit à la question pertinente de l'identification de certaines actions d'orchestration qu'il serait possible d'automatiser tout en respectant le principe de conception de contrôle.

Les questions 5 à 8 portent sur la lisibilité de la table d'orchestration et sur l'utilité des fonctions de tri. Bien que la table soit facile à lire, l'enseignante a apprécié la présence de fonctionnalités pour trier et regrouper les lignes de scénario. Ce résultat peut se comprendre par la grande quantité de lignes de scénario affichées lors de chaque séance de la première expérimentation. En effet, une séance impliquait un quart de la classe (soit six ou sept élèves) et chaque élève se voyait attribuer une série de huit exercices. En y ajoutant les lignes de scénario pour la connexion des élèves, cela représente un total de 54 ou 63 lignes de scénario affichées simultanément à l'écran.

Les questions 9 à 11 portent sur l'interaction avec la table d'orchestration pour créer, modifier, ou supprimer une ligne de scénario. Sur cette thématique, l'enseignante a trouvé facile la création de nouvelles lignes de scénario. Ce point montre l'intérêt de pouvoir créer des lignes de scénario aussi bien grâce à l'interface de *primo-scripting* que grâce à la table d'orchestration, ce qui laisse la liberté à l'enseignante d'utiliser l'instance CHAO-SimBûchettes en fonction de ses propres préférences. Enfin, l'enseignante n'a pas d'avis concernant la fonctionnalité de suppression d'une ligne de scénario. Ce résultat peut s'expliquer par le besoin qu'a évoqué l'enseignante d'avoir les lignes de scénario automatiquement supprimées lorsqu'un élève a fini un exercice correctement. Cette fonctionnalité peut paraître inté-

ressante, mais un risque est que l'enseignante oublie ce qui a déjà été réalisé par ses élèves. Pour répondre à ce besoin, nous avons, entre les deux expérimentations, implémenté un code couleur pour différencier les états de chaque ligne de scénario : blanc par défaut, rouge pour une tâche en cours, orange pour une tâche validée mais incorrecte, et verte pour une tâche validée et correcte.

5.3.4 *Discussion*

Une seule enseignante a été impliquée dans l'évaluation des outils de *runtime scripting*. Avec un effectif aussi faible, il est donc difficile de conclure quant à l'efficacité et l'utilisabilité de l'interface de table d'orchestration.

Cependant, la première expérimentation a requis un grand nombre d'actions de *runtime scripting* et l'enseignante a été en mesure de modifier le scénario en temps réel. En rappelant le fait que l'enseignante n'avait jamais utilisé l'application Chao-SimBûchettes auparavant, cette première expérimentation nous a fourni des éléments de réponse quant à la facilité de prise en main de l'interface de table d'orchestration. De plus, les changements se sont répercutés correctement sur le comportement des applications SimBûchettes des élèves ce qui suggère que les outils de *runtime scripting* sont efficaces pour modifier un scénario pédagogique.

Lors de la seconde expérimentation, les exercices ont été construits pour être plus difficiles dès le départ. Et alors que des situations auraient pu mener à la modification de règles de rétroaction, l'enseignante a préféré aller voir les élèves en difficulté pour leur prodiguer des conseils. Cette observation n'est pas pour nous déplaire, et vient confirmer l'idée que l'utilisation d'un outil technologique ne doit pas se faire aux dépens des pratiques des enseignants. En effet, l'enseignante n'avait pas l'habitude d'utiliser une tablette en classe, comme l'atteste les nombreuses fois où elle est allée voir ses élèves en oubliant sa tablette sur son bureau.

Il est difficile d'affirmer que la table d'orchestration est utilisable pour gérer et modifier un scénario en temps réel. Nous pouvons néanmoins nous baser sur les travaux de Sobreira pour dire que la représentation en forme de table est facile à utiliser pour éditer un scénario pédagogique (Sobreira, 2014). La table d'orchestration ne fait que compléter cette interface en proposant des outils de tri et de regroupement de lignes de scénario, et en fournissant des boutons d'actions pour permettre de rendre effectifs des modifications du scénario.

De façon générale, ces expérimentations nous ont permis de voir que l'enseignante a pu se servir des outils de *runtime scripting* et de la table d'orchestration pour modifier le scénario pédagogique en temps réel, et que ces modifications se sont répercutées, comme prévu, sur le comportement des applications-élève.

5.4 QUANTIFICATION DU TRAVAIL D'INSTANCIATION DU FRAMEWORK

5.4.1 *Méthodologie*

Au cours de cette thèse, nous avons instancié le framework CHAO à trois reprises avec trois applications distinctes : la dictée négociée, SimBûchettes, et Topeka. L'objectif du travail d'évaluation présenté dans cette section est de quantifier et de décrire la tâche nécessaire d'instanciation du framework. Puisque nous avons élaboré le framework en nous basant sur la dictée négociée, nous avons laissé de côté l'instance CHAO-Dictée négociée pour cette analyse.

Nous avons donc effectué dans un premier temps ce travail d'instanciation du framework pour l'application SimBûchettes. Le protocole utilisé pour cette étude est décrit en TABLE 13. Deux personnes ont été impliquées dans ce travail de programmation : l'auteur de la thèse a été en charge de l'instanciation du framework et la stagiaire de Master qui a développé SimBûchettes a été en charge de l'adaptation de l'application-élève pour permettre son orchestration.

Ce même protocole a été repris dans le cadre de l'instanciation du framework CHAO pour l'application Topeka à une exception près : la totalité du travail d'instanciation et de modification a été réalisée par l'auteur de cette thèse.

Pour quantifier ce travail d'instanciation du framework, nous avons calculé les différences en nombres de lignes de code source [2] entre le framework et sa version instanciée, et entre l'application-élève et sa version adaptée pour l'orchestration. Pour l'instanciation du framework, nous avons poussé l'analyse en classant les modifications apportées en cinq catégories :

- Les importations de librairies et de *packages* ;
- Les déclarations de classes et implémentations des constructeurs ;
- Les déclarations d'attributs de classe et implémentations des *getters* et des *setters* ;
- Les implémentations ou les modifications de méthodes déclarées par le framework mais vides (par exemple, les méthodes abstraites déclarées dans la classe `MonitoringTableRow`) ;
- Les déclarations et implémentations de nouvelles méthodes.

Nous avons ensuite caractérisé ces cinq catégories en modifications *faciles* et *difficiles*. Les modifications faciles sont les importations de librairies, les déclarations de classes, et les déclarations d'attributs ; les modifications potentiellement difficiles sont les implémentations de méthodes abstraites et les déclarations de nouvelles méthodes.

2. Les commentaires ne sont pas comptabilisés. Par contre, les lignes ne contenant qu'une accolade ouvrante ou fermante le sont.

En ce qui concerne les données liées à l'adaptation de l'application-élève, nous nous sommes contentés d'observer l'augmentation relative en nombre total de lignes de code et d'identifier les raisons de ces modifications.

Étapes	Description
État initial	Les objets utilisés sont le framework CHAO dans son état non-instancié et l'application-élève cible.
Développement informatique	L'informaticien instancie le framework et modifie aussi l'application-élève pour permettre les communications entre tablettes.
État final	Une instance du framework CHAO pour l'orchestration de situations mises en œuvre grâce à l'utilisation de l'application-élève.
Analyse des différences	À partir de la différence en nombres de lignes de code (sans compter les commentaires ni les lignes vides), nous avons construit des indicateurs dans l'objectif de fournir des éléments de réponse quant au travail à fournir par un informaticien pour l'instanciation du framework.

TABLE 13 – Protocole utilisé pour quantifier le travail d'instanciation du framework.

5.4.2 *Résultats*

5.4.2.1 *Développement de* CHAO-*SimBûchettes*

La TABLE 14 présente les données recueillies après le développement de CHAO-SimBûchettes.

Type d'objet	Ancienne valeur	Nouvelle valeur	Augmentation relative
Classe	37	48	+29.73%
Méthode	157	224	+42.68%
Propriété, Attribut	109	196	+79.82%
Total de lignes de code	1688	2756	+63.27%
Moyenne de lignes de code pour une classe	45.62	57.33	N/A
Lignes de code minimum pour une classe	4	3	N/A
Lignes de code maximum pour une classe	138	192	N/A

TABLE 14 – Données quantitatives relatives au développement de CHAO-SimBûchettes.

Dans un premier temps, nous observons la création de 11 nouvelles classes :

— Trois classes, héritant de la classe `Component`, utilisées pour définir les types de composant d'un scénario mis en œuvre par SimBûchettes (`Student`, `Task`, et `Parameter`) ;
— Une interface utilisée pour déclarer une méthode abstraite de récupération des données additionnelles pour un acteur ;

— Trois classes pour spécifier le contenu des informations de *monitoring* (`MonitoringData`, `ProductionMonitoringData`, et `ProgressMonitoringData`) ;
— Deux classes pour faire le lien entre les messages MQTT et le contenu des informations de *monitoring* :
  — `ActionType` qui énumère les différentes actions que peut effectuer un élève et qui ont été détaillées en SECTION 4.3.3 ;
  — `Container` qui correspond aux différentes boîtes dans lesquelles un élève peut déposer des bûchettes ;
— Deux classes pour afficher les informations de *monitoring* à l'écran avec la représentation particulière de tableau comme illustrée en FIGURES 19 et 20 (`ProductionMonitoringTableLayout`, et `ProgressMonitoringTableLayout`).

Dans un second temps, nous observons l'ajout de 67 nouvelles méthodes, dont 12 correspondent à des méthodes de traitement situées dans la classe `JsonHelper` et 50 correspondent à des méthodes liées à la structuration et l'affichage des informations de *monitoring*. Les nouvelles méthodes restantes correspondent à la création et la réception de messages MQTT. Ce nombre total de nouvelles méthodes créées peut paraître important, mais il reflète notre volonté de suivre la bonne pratique expliquant qu'une « méthode ne doit faire qu'une chose » (Martin, 2008, p. 35).

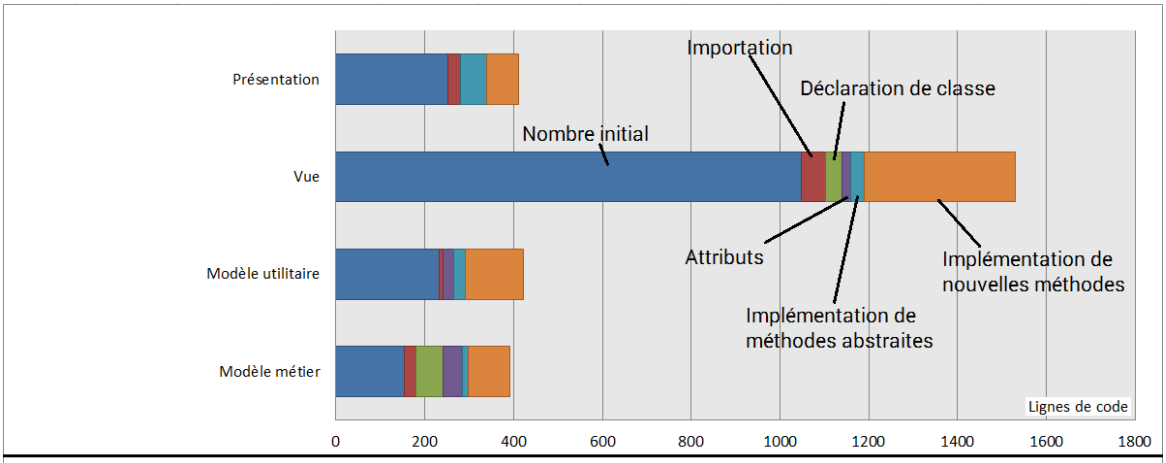
Enfin, nous pouvons observer que l'instanciation du framework a nécessité l'ajout de 1068 lignes de code, soit une augmentation relative de 63.27%.

Pour mieux comprendre ce que représentent ces 1068 lignes de code supplémentaires, nous avons analysé les modifications apportées pour chaque couche du modèle MVP. La TABLE 15 présente un résumé des modifications apportées au framework.

Éléments du modèle MVP	Nb de classe	Classes créées	Lignes de code	Augmentation relative	Lignes de code / Total	Part dans les modifications
Modèle métier	13	9	392	+154.55%	14.22%	22.28%
Modèle utilitaire	5	0	423	+81.55%	15.35%	17.79%
Vue	22	2	1530	+45.85%	55.52%	45.03%
Présentation	8	0	411	+63.10%	14.91%	14.89%

TABLE 15 – Description détaillées des modifications apportées à chaque couche du modèle MVP pour le développement de CHAO-SimBûchettes.

En reprenant les cinq catégories de modifications précisées en SECTION 5.4.1, nous avons pu compter le nombre de lignes de code ajoutées pour chaque couche du modèle MVP. Cette analyse est synthétisée par la FIGURE 49.



Lignes de code	Modèle métier	Modèle utilitaire	Vue	Présentation
Nombre initial	154	233	1049	252
Importation	27	8	54	28
Déclaration de classe et Constructeur	60	0	37	0
Attribut et Getter/Setter	43	25	20	0
Implémentation de méthodes abstraites	14	25	31	60
Déclaration et implémentation de nouvelles méthodes	94	132	339	71

FIGURE 49 – Synthèse des modifications apportées pour l'instance CHAO-SimBûchettes, classées par couche du modèle MVP et par catégories de modification de code.

Nous pouvons constater dans un premier temps que, pour chaque couche sauf celle Modèle métier, les lignes de code fournies initialement par le framework non-instancié (en bleu) constitue la plus grande part des lignes de code. Puis, pour chaque couche, les déclarations de nouvelles méthodes représentent la plus grosse part des modifications (en orange). D'ailleurs, ce type de modification compte pour 23% du total de lignes de code qui composent le framework instancié CHAO-SimBûchettes (comme illustré dans la FIGURE 50).

Dans un second temps, la TABLE 15 nous informe que 45.03% des modifications concernent des ajouts dans la couche Vue. Une première conclusion que nous en tirons est que la plus grosse partie du travail d'instanciation est liée à l'affichage d'éléments graphiques. Les données de la couche Vue présentées par la FIGURE 49 nous permettent ensuite de conclure que ce travail concerne essentiellement la création de nouvelles méthodes. Or nous avons vu précédemment que les nouvelles méthodes créées avaient pour la plupart pour objectif la structuration et l'affichage des informations de *monitoring*.

Nous en concluons que la quantité de travail à fournir de la part de l'informaticien dépend principalement de la complexité des informations de *monitoring*.

En étudiant plus en détail le diagramme en FIGURE 50, nous pouvons tirer les résultats suivants :

— 61% du total de lignes de code est fourni par le framework non-instancié ;

— 11% du total de lignes de code est constitué de modifications *faciles* à réaliser pour l'informaticien (c'est-à-dire, les importations de librairies, les déclarations de nouvelles classes et les implémentations de constructeurs, et les déclarations d'attributs de classe ainsi que les implémentations des *getters* et *setters*) ;
— 28% du total de lignes de code est constitué de modifications *difficiles* à réaliser pour l'informaticien (les déclarations et implémentations de nouvelles méthodes, et les implémentations de méthodes abstraites déclarées par le framework).

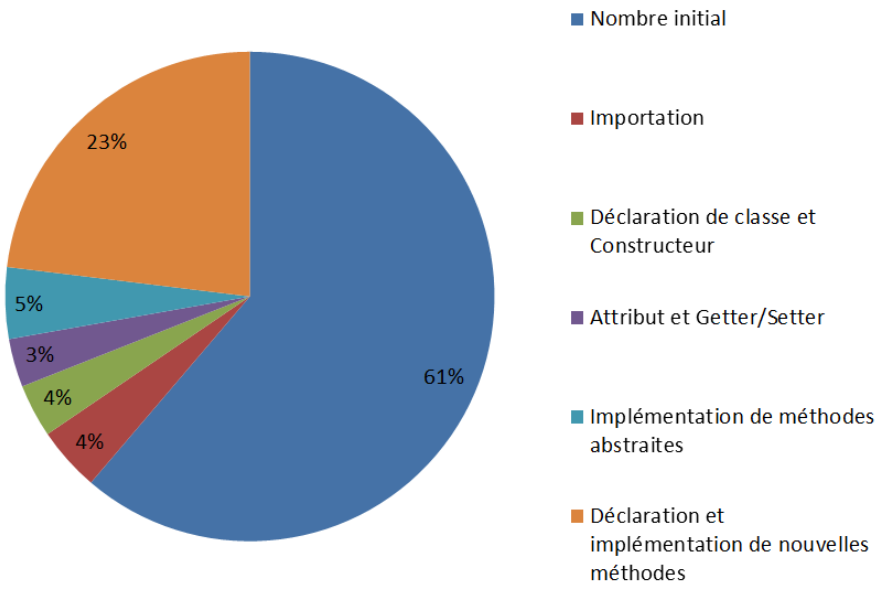


FIGURE 50 – Part de nombres de lignes de code initial et de chaque catégorie de modification pour CHAO-SimBûchettes.

5.4.2.2 *Développement de* CHAO-*Topeka*

La TABLE 16 présente les données quantitatives du développement de CHAO-Topeka.

Ce travail a nécessité la déclaration et l'implémentation de seulement quatre classes : `Student`, `Phase`, `Task`, et `Parameter`. Ce sont ces classes qui définissent les types de composant utilisés pour structurer un scénario pédagogique mis en œuvre par l'application Topeka.

Nous n'avons pas eu besoin de créer de classe pour spécifier une structure complexe d'affichage des informations de *monitoring*. En effet, les informations de *monitoring* sont représentées par des barres de progression (voir la FIGURE 37), et sont donc plus simples que dans le cas de CHAO-SimBûchettes.

Nous notons aussi la déclaration et l'implémentation de 31 nouvelles méthodes (soit presque deux fois moins que pour CHAO-SimBûchettes).

Type d'objet	Ancienne valeur	Nouvelle valeur	Augmentation relative
Classe	37	41	+10.81%
Méthode	157	188	+19.75%
Propriété, Attribut	109	137	+25.69%
Total de lignes de code	1688	2178	+29.03%
Moyenne de lignes de code pour une classe	45.62	53.12	N/A
Lignes de code minimum pour une classe	4	3	N/A
Lignes de code maximum pour une classe	138	161	N/A

TABLE 16 – Données quantitatives relatives au développement de CHAO-Topeka.

Parmi ces 31 nouvelles méthodes, dix sont ajoutées à la classe `JsonHelper` utilisée pour faire le lien entre les messages JSON et les objets Java (contre 12 dans le cadre de CHAO-SimBûchettes), et 16 sont liées à la structuration et l'affichage des informations de *monitoring* (contre 50 dans le cadre de CHAO-SimBûchettes). Cet élément montre de façon nette que l'implémentation de barres de progression pour l'affichage d'informations de *monitoring* est beaucoup plus directe.

Enfin, nous observons une augmentation relative du nombre de lignes de code de 29.03% (soit 490 nouvelles lignes de code). Ce total est nettement inférieur aux 63.27% d'augmentation relative (ou 1068 lignes ajoutées) dans le cadre de l'instance CHAO-SimBûchettes.

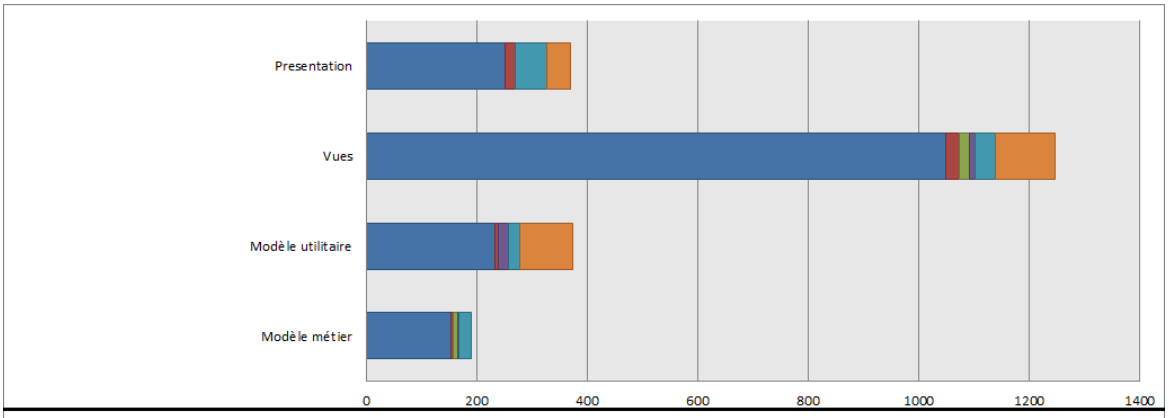
Nous avons aussi mené le même travail de catégorisation des ajouts de lignes de code au cours du développement de CHAO-Topeka. Une synthèse est fournie par la TABLE 17.

Éléments du modèle MVP	Nb de classe	Classes créées	Lignes de code	Augmentation relative	Lignes de code / Total	Part dans les modifications
Modèle métier	8	4	189	+22.73%	8.68%	7.14%
Modèle utilitaire	5	0	372	+59.66%	17.08%	28.37%
Vue	20	0	1248	+18.97%	57.30%	40.61%
Présentation	8	0	369	+46.43%	16.94%	23.88%

TABLE 17 – Description détaillées des modifications apportées à chaque couche du modèle MVP pour le développement de CHAO-Topeka.

Nous observons la même tendance montrant que les éléments de la couche Vue comptent pour 40.61% des modifications totales (contre 45.03% dans le cadre de CHAO-SimBûchettes). Cependant, et parce que CHAO-Topeka n'a pas nécessité la déclaration de classe pour structurer les informations de *monitoring*, nous constatons que la part de modifications des éléments métiers du Modèle a diminué au profit des deux autres couches restantes.

La FIGURE 51 montre aussi que la part de lignes de code définie initialement par le framework reste majoritaire dans chacune des couches du modèle MVP. Et, de la même façon que pour CHAO-SimBûchettes, les déclarations de nouvelles méthodes représentent la principale tâche d'instanciation de l'informaticien et comptent pour 11% du total de lignes de code utilisées pour développer CHAO-Topeka (voir FIGURE 52).



Nb de lignes	Modèle métier	Modèle utilitaire	Vues	Presentation
Nombre initial	154	233	1049	252
Importation	4	6	24	18
Déclaration de classe et Constructeur	8	0	18	0
Attribut et Getter/Setter	2	19	11	0
Implémentation de méthodes abstraites	21	19	36	57
Déclaration et implémentation de nouvelles méthodes	0	97	108	42

FIGURE 51 – Synthèse des modifications apportées pour l'instance CHAO-Topeka, classées par couche du modèle MVP et par catégories de modification de code.

En catégorisant ces ajouts en modifications *faciles* et modifications *difficiles*, la FIGURE 52 nous apporte les informations suivantes :

— 78% du total de lignes de code est fourni par le framework non-instancié ;
— 5% du total de lignes de code est constitué des modifications *faciles* à réaliser par un informaticien ;
— 17% du total de lignes de code est constitué des modifications *difficiles* à réaliser par un informaticien.

Ces résultats, mis en relation avec ceux obtenus lors du développement de CHAO-SimBûchettes, renforcent l'hypothèse que nous avions formulé quant à la corrélation entre la complexité des informations de *monitoring* et la complexité du travail d'instanciation.

5.4.2.3 *Modifications des applications-élève*

Nous avons analysé les modifications apportées aux applications-élève pour déterminer quelles fonctionnalités ont été adaptées pour rendre ces applications orchestrables. En SECTION 4.6.5, nous avons identifié deux types de modifications : celles pour permettre la communication avec le framework, et celles pour synthétiser les données des messages à envoyer et pour traiter les messages reçus.

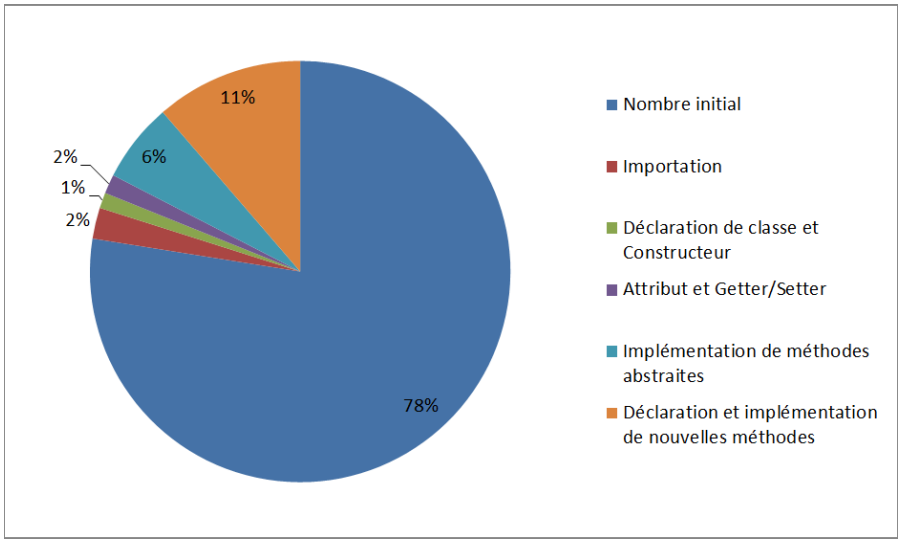


FIGURE 52 – Part de nombres de lignes de code initial et de chaque catégorie de modification pour CHAO-Topeka.

La TABLE 18 présente une comparaison des données de modifications des applications SimBûchettes et Topeka.

Type de modification	SimBûchettes	Topeka
Nombre de lignes de code ajoutées	1094	381
Augmentation relative	21.41%	9.79%

Classes créées	`Eleve`
	`EleveTextView`
	`MQTT`
	`NomEleveCommunication`
		`IFragment`
		`MQTT`
		`ServerHelper`

Classe présentant le plus de modifications	`MQTT`	`MQTT`
Lignes de codes modifiées pour cette classe	688	158
Part de cette classe dans les modifications	62.89%	48.29%

TABLE 18 – Comparaison entre les données de modifications de SimBûchettes et celles de Topeka.

Nous remarquons une différence quant au nombre de lignes de code ajoutées. Cette différence résulte de la qualité de la conception logicielle des deux applications-élève[3]. En effet, l'architecture logicielle de l'application-élève SimBûchettes a été mal conçue. Cela a entraîné un fort taux de redondance dans certaines classes de traitement, et en particulier dans la classe `MQTT`. Améliorer la qualité de

3. Nous rappelons que SimBûchettes a été développée par une étudiante de Master 1 quand Topeka l'a été par des ingénieurs de Google.

l'architecture de SimBûchettes ne s'inscrit pas dans le cadre de cette thèse, et nous l'avons donc laissé en l'état.

En ce qui concerne les modifications pour permettre les échanges inter-tablettes, nous remarquons dans les deux cas que l'implémentation de la classe `MQTT` constitue la classe ayant nécessité le plus de travail. En effet, la classe `MQTT` présente 688 lignes de code supplémentaires dans le cadre de SimBûchettes contre 158 pour Topeka.

Nous voyons deux explications à une telle différence entre ces chiffres. Premièrement, cette différence résulte du fait que les méthodes de traitement ont été implémentées directement dans la classe `MQTT` de SimBûchettes, alors que nous les avons déclarées dans la classe `JsonHelper` (qui existait déjà) de Topeka. En comparant la classe `MQTT` de SimBûchettes avec les classes `MQTT` et `JsonHelper` de Topeka, nous obtenons respectivement des parts de modifications de 62.89% et 65.61%. Deuxièmement, cette différence provient aussi de la forte redondance évoquée dans le paragraphe précédent.

En ce qui concerne les modifications pour synthétiser les informations des messages à envoyer et pour traiter les messages reçus, elles constituent 35.92% des modifications apportées à SimBûchettes et 36.22% pour Topeka.

Finalement, ces données montrent que le travail d'adaptation d'une application-élève repose bien sur l'implémentation d'un module de communication et de fonctionnalités permettant de synthétiser les informations de *monitoring* à envoyer ou de modifier les paramètres d'un exercice en fonction des messages reçus.

5.4.3 *Discussion*

Du point de vue de la conception d'un framework, nous constatons que le framework Chao fournit initialement 61% et 78% des lignes de code des instances Chao-SimBûchettes et Chao-Topeka. En y ajoutant les modifications *faciles*, ces taux montent respectivement à 72% et 83%. Ces résultats suggèrent que Chao, en tant que framework, est une base logicielle supportant grandement le travail de développement d'une technologie d'orchestration.

En analysant les modifications dans chaque couche du modèle MVP, nous observons que les éléments de la couche Vue ont nécessité le plus de modifications (45.03% pour Chao-SimBûchettes et 40.61% pour Chao-Topeka). En outre, les éléments de la couche Modèle (métiers et utilitaires) comptent respectivement pour 40.07% et 35.51% des modifications d'instanciation. Ces deux résultats confirment nos propos de la Section 4.6 précisant que l'instanciation du framework Chao passe par l'instanciation des classes métiers et utilitaires et par l'instanciation et l'affichage des informations de *monitoring*.

Au regard de ces informations quant à l'instanciation du framework Chao, nous en tirons deux conclusions. La première conclusion

est que la quantité de travail d'instanciation dépend de la complexité de la structure du scénario pédagogique et des informations de *monitoring*. La seconde conclusion est que le travail *difficile* d'instanciation constitue une part minoritaire. Cela suggère que le framework CHAO simplifie le travail de l'informaticien en implémentant de base la plupart des fonctionnalités proposées ou en facilitant leurs adaptations.

En ce qui concerne le travail d'adaptation d'une application-élève pour la rendre orchestrable, les données montrent que ce travail correspond essentiellement à l'implémentation de la classe `MQTT` permettant les échanges avec le framework. Les méthodes à implémenter sont donc celles chargées de se connecter au serveur, de créer et d'envoyer des messages MQTT, et de recevoir et traiter les messages entrants. Nous en concluons que ce travail d'adaptation dépend en premier lieu de la complexité de la structure des messages MQTT. Plus cette structure est complexe, plus un message MQTT sera difficile à créer et à traiter.

6

CONCLUSION, DISCUSSION, PERSPECTIVES

Contenu du chapitre

6.1 CONCLUSION

6.1.1 *Synthèse des travaux menés*

Les travaux menés au cours de cette thèse ont conduit à la construction d'un modèle d'orchestration et à la conception et au développement du framework CHAO. Ce framework propose des interfaces utilisateurs basées sur les modèles d'interfaces conçus en SECTION 4.3. Ces contributions viennent répondre à la problématique énoncée en SECTION 3.2 et que nous rappelons ici : « Est-il possible de concevoir un framework logiciel qui soit facile à instancier et qui, une fois instancié, supporte un enseignant dans l'orchestration d'un scénario en classe utilisant des tablettes ? »

Nous avons évalué ces contributions sur deux aspects :

1. L'utilisabilité et l'efficacité d'une instance du framework CHAO dans le support d'un enseignant au cours de son activité d'orchestration ;
2. La complexité du travail d'instanciation du framework CHAO par un informaticien.

Par rapport au premier aspect, les résultats montrent que les interfaces utilisateurs fournies par les instances du framework sont utilisables et semblent supporter efficacement un enseignant dans l'orchestration d'un scénario pédagogique en classe. Cela vient renforcer l'idée qu'une structure de table offre des caractéristiques d'utilisabilité et de flexibilité. De plus, les avis des enseignantes quant à l'utilité des informations de *monitoring* valident notre démarche de faire une distinction entre informations de progression et de production.

Par rapport au second aspect, les résultats obtenus suggèrent que le framework CHAO constitue une base logicielle facilitant le travail de développement d'une technologie orchestrable. De plus, nous avons identifié une relation entre la quantité de lignes de code à produire et la complexité de la structure du scénario pédagogique et des informations de *monitoring*.

6.1.2 *Spectre d'application*

Nous avons construit le modèle d'orchestration et développé le framework CHAO avec le soucis de proposer une base conceptuelle et technique pour des travaux futurs dans le domaine de l'orchestration d'EIAH. Dans cette section, nous détaillons le spectre d'application de ces deux contributions. Pour le modèle d'orchestration, nous allons préciser son intérêt pour la spécification d'un scénario pédagogique et des informations de *monitoring* associées. Pour le développement du framework CHAO, nous allons commenter nos choix à propos des interfaces utilisateurs et des technologies tierces utilisées.

6.1.2.1 *Spectre d'application du modèle d'orchestration*

Le modèle d'orchestration est constitué de deux parties : les actions d'orchestration que mènent un enseignant pour créer et modifier un scénario pédagogique, et les données de *monitoring* l'informant du déroulement de ce scénario en temps réel.

En ce qui concerne les actions d'orchestration, le modèle se repose sur la notion centrale de ligne de scénario définissant un acteur, une tâche, et une production. Nous avons conçu ce modèle avec l'idée qu'un scénario pédagogique peut être défini comme étant une succession de lignes de scénario. Néanmoins, ces éléments du modèle ne sont pas spécifiques à une telle représentation. Il convient ici de différencier le fond de la forme : quelle que soit le type représentation d'un scénario utilisé, les notions d'acteur, de tâche, et de production peuvent spécifier un grand nombre de scénarios différents.

En ce qui concerne les données de *monitoring*, nous proposons de fournir des informations de progression pour compléter celles illustrant la production des acteurs. Ces informations de progression sont pertinentes s'il est possible de déterminer une valeur de référence correspondant au minimum d'actions requises pour terminer un exercice. Nous obtenons ces informations de progression en comparant cette valeur de référence avec le nombre d'actions d'un acteur.

6.1.2.2 *Spectre d'application du framework*

Le framework CHAO propose des interfaces utilisateurs conçues pour permettre à un enseignant de réaliser des actions d'orchestration. Ces interfaces, qui ont été évaluées quant à leur utilisabilité et

leur efficacité, font partie intégrante du framework. En effet, si un informaticien souhaite modifier une de ces interfaces, ces changements risquent de se répercuter dans l'ensemble des couches du modèle MVP et d'entraîner un travail conséquent.

Nous nous sommes aussi appuyés sur des technologies tierces pour développer le framework : le protocole MQTT, une base de données MySQL, et un service web. L'utilisation de ces technologies est nécessaire pour le fonctionnement du framework CHAO tel que nous l'avons conçu. Cependant, il est possible d'utiliser des technologies équivalentes mais cela nécessiterait des modifications du framework. Ces modifications auront pour conséquences aussi de modifier les composants matériels de l'infrastructure informatique utilisée dans le cadre de cette thèse. Nous rappelons cependant que nous avons conçu cette infrastructure pour pouvoir mener des expérimentations dans des établissements scolaires différents. Dans le cadre d'une utilisation continue dans une classe, le framework peut être adapté pour fonctionner avec l'infrastructure informatique et réseau pré-installée sur place.

Enfin, l'utilisation d'une instance du framework CHAO requiert aussi l'utilisation d'une application-élève. Afin de permettre des échanges de données avec le framework CHAO, cette application nécessite d'être adaptée. Pour cela, il est impératif que l'informaticien soit en mesure d'accéder et de modifier son code source. Ce dernier critère constitue une limitation forte du spectre d'application du framework CHAO.

6.2 DISCUSSION

En reprenant le modèle d'orchestration $5 + 3$ étudié en SECTION 2.1, les travaux que nous avons menés durant cette thèse reprennent les cinq éléments caractérisant l'orchestration. En effet, l'enseignant reste l'acteur principal de l'orchestration. Dans ce contexte, le framework CHAO offre une aide pour la planification d'un scénario pédagogique et la régulation et la visualisation du travail des élèves. Le framework reprend aussi le principe de flexibilité en proposant des outils pouvant modifier le scénario pédagogique prévu.

Nous avons aussi suivi les trois points présentant des pistes méthodologiques pour mener des travaux de recherche en orchestration. Nous avons ainsi collaboré avec des enseignants de primaire pour concevoir une première version du framework CHAO et avons mené des expérimentations en classe durant des séances traditionnelles de cours (ce qui fait référence au principe de pragmatisme). Nous avons aussi suivi le principe de synergie, puisque notre volonté depuis le départ a été de proposer un support aux enseignants et non une technologie réalisant tout le travail d'orchestration à leur place. Enfin, nous proposons aussi un modèle qui spécifie les actions d'orches-

tration d'un scénario pédagogique et les informations de *monitoring* utiles pour suivre le déroulement de ce scénario.

En analysant notre travail et ceux mentionnés dans la SECTION 2.5, nous proposons deux axes de discussion : (1) la possibilité de réutiliser ces technologies pour des situations d'apprentissage en classe mettant en œuvre des scénarios pédagogiques divers, et (2) la conception des outils et des interfaces d'orchestration.

En ce qui concerne la réutilisabilité de la technologie, notre objectif était de concevoir un framework logiciel pouvant servir de base pour le développement de technologies d'orchestration. Un avantage de cette approche est que le framework CHAO peut être réutilisable dans différentes situations comme le suggère les trois instances que nous avons développées au cours de cette thèse. En contrepartie, ce framework n'est pas directement utilisable par un enseignant et nécessite de passer par une étape préliminaire d'instanciation réalisée par un informaticien.

Les travaux de recherche analysés en SECTION 2.5 optent pour une approche différente en proposant des technologies directement prêtes à l'emploi. Cependant, ces technologies ont aussi été conçues dans le cadre de situations particulières d'apprentissage en classe, ce qui limite leur réutilisation. Par exemple, les auteurs de CK3 précisent que cette technologie est indépendante du contenu pédagogique mis en œuvre (Fong, Cober, Moher, et al., 2015) mais il n'est pas possible de réutiliser CK3 pour des contextes autres que des situations d'apprentissage par démarche d'investigation.

En ce qui concerne les outils d'orchestration proposés, nous avons décidé de distinguer trois types d'actions conformément au modèle PRM de Tchounikine : les actions de *primo-scripting*, les actions de *runtime scripting*, et les actions de *monitoring*. Pour chacune de ces actions, notre approche a été de proposer des outils simples à prendre en main. Cela se traduit par des outils moins sophistiqués ou des informations moins complètes en comparaison avec d'autres travaux.

En ce qui concerne l'outil de *primo-scripting*, notre démarche a été de reprendre des travaux de l'équipe montrant qu'une structure de table était facile à utiliser pour un enseignant et permettait d'éditer une grande diversité de scénarios pédagogiques (Sobreira & Tchounikine, 2015). Cependant, de nombreux travaux se sont penchés sur la question de la conception d'un scénario pédagogique, et des langages de modélisation comme IMS-LD (IMS Global Learning Consortium, 2003) ou encore COML (Niramitranon, 2009) ont été construits à partir du langage XML. La sémantique offerte par ces langages permet de mieux détailler un scénario pédagogique qu'avec notre interface de *primo-scripting*, mais induit aussi une difficulté supplémentaire de prise en main pour un enseignant peu familier avec ce type de langage de modélisation.

En ce qui concerne l'outil de *runtime scripting*, nous avons construit la table d'orchestration en gardant la volonté de fournir une interface simple à utiliser. L'outil se rapprochant le plus du nôtre sur cette question du *runtime scripting* est MTDashboard, qui propose à l'enseignant des actions comme passer à la phase suivante, mettre en pause ou reprendre le travail de tous les élèves, et envoyer un message à un groupe d'élèves (Martinez Maldonado et al., 2012).

En ce qui concerne l'outil de *monitoring*, la représentation de la progression des acteurs est une pratique fréquente. En reprenant la classification des outils de suivi élaborée par Soller et collègues (Soller et al., 2005), notre contribution se situe au niveau des « outils miroirs » qui ne font qu'afficher des informations de suivi sans proposer de traitement complexe.

Nous avons aussi fait le choix de fournir des informations quant à la progression des acteurs par rapport à une tâche. L'utilisation de barres de progression pour représenter ce type d'information est courante. Néanmoins, l'information convoyée varie en fonction du degré de précision recherché. Par exemple, des barres de progression peuvent représenter la réalisation de différentes étapes au sein d'un seul exercice (Guéraud & Cagnat, 2006), ou bien la progression d'un acteur dans une succession d'exercices (Roschelle, Rafanan, et al., 2010), ou encore la progression d'un acteur dans l'ensemble du scénario pédagogique (Martinez-Maldonado, Clayphan & Kay, 2015).

De la même façon, l'affichage de la production des élèves pour en informer l'enseignant est une pratique fréquente. Ces informations peuvent être présentées de façon brute comme nous le faisons, ou peuvent être traitées et analysées pour illustrer des traces d'activités. C'est par exemple ce qui est implémenté par MTDashboard puisque les données affichées informent l'enseignant des actions de chaque élève et de leur degré de participation (Martinez Maldonado et al., 2012; Martinez-Maldonado, Dimitriadis, et al., 2013). Parfois, l'affichage des productions a aussi pour but d'informer les élèves sur le travail de leurs camarades avec l'utilisation d'un tableau blanc interactif (Fong, Cober, Moher, et al., 2015; Kreitmayer et al., 2013; Do-Lenh et al., 2012).

Au sujet des informations de *monitoring*, nos travaux se distinguent de ceux mentionnés ci-dessus en proposant simultanément ces deux types d'informations afin d'aider les enseignants à se construire une représentation du déroulement des activités du point de vue de chaque individu (avec les données concernant les productions des acteurs) et de la classe entière (en synthétisant les données relatives à la progression de chaque acteur).

6.3 PERSPECTIVES

Les travaux menés au cours de cette thèse génèrent des questions auxquelles il pourrait être intéressant de répondre dans le futur. Ces questions sont en lien avec les trois acteurs identifiés dans la SECTION 4.1.3 : l'informaticien, l'ingénieur pédagogique, et l'enseignant.

6.3.1 *Perspectives pour l'informaticien*

Les perspectives pour l'informaticien peuvent concerner deux aspects : l'approfondissement de l'évaluation du travail d'instanciation du framework CHAO et l'ajout de fonctionnalités.

Le travail exploratoire que nous avons mené pour quantifier les modifications liées à l'instanciation du framework CHAO a été mené par l'auteur de cette thèse. Une étude plus complète impliquant la participation d'informaticiens externes au projet fournira des informations complémentaires quant à la facilité de prise en main du code source du framework CHAO, quant à la durée nécessaire pour l'instancier pour un scénario spécifique, ou encore quant aux difficultés rencontrées lors de cette instanciation.

Au cours des descriptions de différentes instances du framework CHAO, nous avons présenté des fonctionnalités qui ne sont pas proposées par défaut. Par exemple, CHAO-SimBûchettes propose une fonctionnalité de tri de table et d'ajout de ligne de scénario vide dans le module de *runtime scripting*. Les enseignantes ont parfois exprimé des idées d'amélioration du framework, comme l'envoi automatique de messages de démarrage d'exercices. L'ajout des fonctionnalités lève des questions sur l'éventuelle existence d'actions d'orchestration pouvant compléter les outils déjà proposés par le framework.

Nous avons placé ce projet de recherche dans un contexte précis : l'utilisation en classe de tablettes tactiles par les élèves et l'orchestration de cette situation par un enseignant. Or la mobilité introduite par l'utilisation de tablettes offre de nouvelles situations d'apprentissage en dehors de la classe. Par exemple, dans le cadre d'une sortie au musée (voir les travaux de Proctor et Burton (Proctor & Burton, 2004)), quels éléments du framework et de l'infrastructure nécessiteraient d'être adaptés ?

6.3.2 *Perspectives pour l'ingénieur pédagogique*

Pour l'ingénieur pédagogique, une perspective peut être d'étudier les critères d'intégration d'une instance du framework CHAO dans des contextes scolaires. Il s'agit d'analyser une application-élève déjà utilisée en classe et s'appuyer sur les modèles et le framework pour développer une instance venant en aide aux enseignants pour orchestrer ces situations.

Une autre perspective peut être de reprendre les applications-élève que nous avons utilisée (la dictée négociée, SimBûchettes, et Topeka) et d'étudier comment elles peuvent être adaptées pour d'autres technologies d'orchestration. Ce travail peut mettre en avant des compléments de réponse quant à la réutilisation du modèle d'orchestration et des informations de *monitoring* construits. Ce travail peut aussi permettre de comparer les instances du framework CHAO avec d'autres technologies d'orchestration.

6.3.3 *Perspectives pour l'enseignant*

Les expérimentations ont toutes été menées avec des enseignantes découvrant les outils proposés par le framework. Il peut être intéressant d'étudier l'utilisation de ces outils dans la durée. Dans cette optique, et en identifiant les outils les plus souvent utilisés, la question de l'automatisation de certaines actions d'orchestration peut être envisagée. Cependant, ceci entrerait potentiellement en conflit avec le principe de conception mentionnant le contrôle de l'enseignant sur la situation.

De plus, ces expérimentations ont aussi toutes été conduites dans des écoles primaires. Ce choix peut se justifier par l'importance du rôle d'un enseignant et de ses actions d'orchestration dans ce contexte. Nous avons cependant mentionné des travaux d'orchestration avec des étudiants d'université dans la SECTION 2.2. Une question intéressante est de savoir si des enseignants du secondaire ou du supérieur possèdent les mêmes besoins qu'un enseignant du primaire, et si des instances du framework CHAO peuvent répondre à ces besoins.

Enfin, en revenant vers le contexte d'étude de la thèse, un travail peut être mené sur l'utilisation répétée d'une instance du framework par un enseignant du primaire. Les résultats de cette étude peut procurer des informations quant aux éventuelles limites des outils proposés.

RÉFÉRENCES

Alvarez, C., Alarcon, R. & Nussbaum, M. (2011). « Implementing collaborative learning activities in the classroom supported by one-to-one mobile computing : A design-based process. » In: *The Journal of Systems and Software* 84.11, pp. 1961–1976.

Boticki, I., Looi, C.-K. & Wong, L.-H. (2011). « Supporting Mobile Collaborative Activities through Scaffolded Flexible Grouping. » In: *Journal of Educational Technology & Society* 14.3, pp. 190–202.

Cellier, M. (2004). « Dire l'orthographe : quelques dispositifs. » In: *Langues et études de la langue : Approches linguistiques et didactiques*. Ed. by C. Vargas, J.-F. Halté, B. Combettes & C. Touratier. Publications de l'université de Provence, pp. 311–321.

Chamberlain, M. K., Williams, C. B., Cowan, F. S. & Mistree, F. (2001). « Orchestrating learning in a graduate engineering design course. » In: *13th International Conference on Design Theory and Methodology*.

Chan, T.-W. (2013). « Sharing sentiment and wearing a pair of 'field spectacles' to view classroom orchestration. » In: *Computers & Education* 69, pp. 514–516.

Chan, T.-W. et al. (2006). « One-to-one technology-enhanced learning : an opportunity for global research collaboration. » In: *Research and Practice in Technology Enhanced Learning* 1(1), pp. 3–29.

Cheong, C., Bruno, V. & Cheong, F. (2012). « Designing a Mobile-app-based Collaborative Learning System. » In: *Journal of Information Technology Education : Innovations in Practice* 11.1, pp. 94–119.

Cher, C. (2005). « Comment faire de la dictée un outil d'apprentissage ? » MA thesis. Site de Nîmes: IUFM de Montpellier.

Cuendet, S., Bonnard, Q., Do-Lenh, S. & Dillenbourg, P. (2013). « Designing augmented reality for the classroom. » In: *Computers & Education* 68, pp. 557–569.

Dillenbourg, P. (2013). « Design for classroom orchestration. » In: *Computers & Education* 69, pp. 485–492.

Dillenbourg, P., Järvelä, S. & Fischer, F. (2009). « The Evolution of Research on Computer-Supported Collaborative Learning. » In: *Technology-Enhanced Learning*. Ed. by D. N. Balacheff, D. S. Ludvigsen, D. T. d. Jong, D. A. Lazonder & D. S. Barnes. Springer Netherlands, pp. 3–19.

Dillenbourg, P. & Jermann, P. (2010). « Technology for Classroom Orchestration. » en. In: *New Science of Learning*. Ed. by M. S. Khine & I. M. Saleh. DOI : 10.1007/978-1-4419-5716-0_26. Springer New York, pp. 525–552.

Dillenbourg, P. & Tchounikine, P. (2007). « Flexibility in macro-scripts for computer-supported collaborative learning. » en. In: *Journal of Computer Assisted Learning* 23.1, pp. 1–13.

Dillenbourg, P., Zufferey, G., Alavi, H., Jermann, P., Do-Lenh, S., Bonnard, Q., Cuendet, S. & Kaplan, F. (2011). « Classroom orchestration : The third circle of usability. » In: *To See the World and a Grain of Sand : Learning Across Levels of Space, Time, and Scale : CSCL 2013 Conference Proceedings*. 10th International Conference on Computer-Supported Collaborative Learning. Vol. 1, pp. 510–517.

Echeverria, A., Nussbaum, M., Calderon, J. F., Bravo, C., Infante, C. & Vasquez, A. (2011). « Face-to-face collaborative learning supported by mobile phones. » In: *Interactive Learning Environments* 19.4, pp. 351–363.

Fong, C., Cober, R. M., Madeira, C. A., Messina, R., Murray, J., Peebles, B. & Slotta, J. D. (2013). « Common Knowledge : Orchestrating Synchronously Blended F2F Discourse in the Elementary Classroom. » In: *To See the World and a Grain of Sand : Learning Across Levels of Space, Time, and Scale : CSCL 2013 Conference Proceedings*. 10th International Conference on Computer-Supported Collaborative Learning. Vol. 2, pp. 26–29.

Fong, C., Cober, R. M., Moher, T. & Slotta, J. D. (2015). « The 3R Orchestration Cycle : Fostering Multi-Modal Inquiry Discourse in a Scaffolded Inquiry Environment. » In: *Exploring the Material Conditions of Learning*. 11th International Conference on Computer-Supported Collaborative Learning.

Framework — Wikipédia (2016). URL: https://fr.wikipedia.org/wiki/Framework (visited on 03/04/2016).

Guéraud, V. & Cagnat, J.-M. (2006). « Automatic Semantic Activity Monitoring of Distance Learners Guided by Pedagogical Scenarios. » In: *Innovative Approaches for Learning and Knowledge Sharing : First European Conference on Technology Enhanced Learning, EC-TEL 2006 Crete, Greece, October 1-4, 2006 Proceedings*. Ed. by W. Nejdl & K. Tochtermann. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 476–481.

Ifenthaler, D. & Schweinbenz, V. (2013). « The acceptance of Tablet-PCs in classroom instruction : The teachers' perspectives. » In: *Computers in Human Behavior* 29.3, pp. 525–534.

IMS Global Learning Consortium (2003). *Learning Design Specification*. URL: http://www.imsglobal.org/learningdesign/index.html (visited on 05/05/2016).

Jahnke, I., Cerratto-Pargman, T., Furberg, A., Järvelä, S. & Wasson, B. (2015). « Changing Teaching and Learning Practices in Schools with Tablet-Mediated Collaborative Learning (#TMCL15) : Nordic, European and International Views. » In: *Exploring the Material*

Conditions of Learning. 11th International Conference on Computer-Supported Collaborative Learning. Vol. 2, pp. 888–893.

Kharrufa, A., Martinez-Maldonado, R., Kay, J. & Olivier, P. (2013). « Extending Tabletop Application Design to the Classroom. » In: *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*. ITS '13. St. Andrews, Scotland, United Kingdom: ACM, pp. 115–124.

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P. & Fischer, F. (2007). « Specifying computer-supported collaboration scripts. » en. In: *International Journal of Computer-Supported Collaborative Learning* 2.2-3, pp. 211–224.

Kollar, I. & Fischer, F. (2013). « Orchestration is nothing without conducting – But arranging ties the two together ! : A response to Dillenbourg (2011). » In: *Computers & Education* 69, pp. 507–509.

Kreitmayer, S., Rogers, Y., Laney, R. & Peake, S. (2013). « UniPad : Orchestrating Collaborative Activities Through Shared Tablets and an Integrated Wall Display. » In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp '13. New York, NY, USA: ACM, pp. 801–810.

Lachaud-Beauchene, M.-H. (2006). « Dans quelle mesure la dictée négociée permet-elle aux élèves de faire de l'ORL ? » PhD thesis. IUFM de Bourgogne. 41 pp.

Do-Lenh, S., Jermann, P., Legge, A., Zufferey, G. & Dillenbourg, P. (2012). « TinkerLamp 2.0 : Designing and Evaluating Orchestration Technologies for the Classroom. » In: *21st Century Learning for 21st Century Skills : 7th European Conference of Technology Enhanced Learning, EC-TEL 2012, Saarbrücken, Germany, September 18-21, 2012. Proceedings*. Ed. by A. Ravenscroft, S. Lindstaedt, C. D. Kloos & D. Hernández-Leo. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 65–78.

Martin, R. C. (2008). *Clean Code : A Handbook of Agile Software Craftsmanship*. Prentice Hall.

Martinez Maldonado, R., Dimitriadis, Y., Kay, J., Yacef, K. & Edbauer, M.-T. (2012). « Orchestrating a Multi-tabletop Classroom : From Activity Design to Enactment and Reflection. » In: *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces*. ITS '12. New York, NY, USA: ACM, pp. 119–128.

Martinez-Maldonado, R., Clayphan, A. & Kay, J. (2015). « Deploying and Visualising Teacher's Scripts of Small Group Activities in a Multi-surface Classroom Ecology : a Study in-the-wild. » In: *Computer Supported Cooperative Work (CSCW)* 24.2, pp. 177–221.

Martinez-Maldonado, R., Dimitriadis, Y., Kay, J., Yacef, K. & Edbauer, M.-T. (2013). « MTClassroom and MTDashboard : supporting analysis of teacher attention in an orchestrated multi-tabletop classroom. » In: *To See the World and a Grain of Sand : Learning Across Levels of Space, Time, and Scale : CSCL 2013 Conference Proceedings*.

10th International Conference on Computer-Supported Collaborative Learning. Vol. 1, pp. 320–327.

Martinez-Maldonado, R., Kay, J. & Yacef, K. (2010). « Collaborative concept mapping at the tabletop. » In: *ACM International Conference on Interactive Tabletops and Surfaces*. ACM, pp. 207–210.

MQTT (2014). URL: http://mqtt.org/ (visited on 02/19/2016).

Niramitranon, J. (2009). « Orchestration Learning in a one-to-one Technology Classroom. » PhD thesis. Learning Sciences Research Institute: University of Notthingham.

Niramitranon, J., Sharples, M., Greenhalgh, C. & Lin, C.-P. (2010). « Orchestrating Learning in a One-to-One Technology Classroom. » In: *2010 6th IEEE International Conference on Wireless, Mobile and Ubiquitous Technologies in Education (WMUTE)*, pp. 96–103.

Onrubia, J. & Engel, A. (2011). « The role of teacher assistance on the effects of a macro-script in collaborative writing tasks. » In: *International Journal of Computer-Supported Collaborative Learning* 7.1, pp. 161–186.

Prieto, L. P. (2015). « Classroom Technology Design Guidelines : A Focused Survey. » In: *Proceedings of the Orchestrated Collaborative Classroom Workshop 2015*. co-located with the 11th International Conference on Computer-Supported Collaborative Learning, pp. 10–14.

Prieto, L. P., Dimitriadis, Y., Asensio-Pérez, J. & Looi, C.-K. (2015). « Orchestration in learning technology research : evaluation of a conceptual framework. » In: *Research in Learning Technology* 23.

Prieto, L. P., Holenko Dlab, M., Gutiérrez, I., Abdulwahed, M. & Balid, W. (2011). « Orchestrating technology enhanced learning : a literature review and a conceptual framework. » In: *International Journal of Technology Enhanced Learning* 3.6, pp. 583–598.

Prieto, L. P., Sharma, K. & Dillenbourg, P. (2015). « Studying Teacher Orchestration Load in Technology-Enhanced Classrooms. » In: *Design for Teaching and Learning in a Networked World*. Ed. by G. Conole, T. Klobučar, C. Rensing, J. Konert & É. Lavoué. Springer, pp. 268–281.

Prieto, L. P., Sharma, K., Wen, Y. & Dillenbourg, P. (2015). « The burden of facilitating collaboration : towards estimation of teacher orchestration load using eye-tracking measures. » In: *Proceedings of the 11th International Conference on Computer-Supported Collaborative Learning (CSCL 2015)*. Gothenburg Sweden, pp. 212–219.

Proctor, N. & Burton, J. (2004). « Tate modern multimedia tour pilots 2002-2003. » In: *Learning with Mobile Devices : Research and Development.'London : Learning and Skills Development Agency*, pp. 127–30.

Roschelle, J., Dimitriadis, Y. & Hoppe, U. (2013). « Classroom orchestration : Synthesis. » In: *Computers & Education* 69, pp. 523–526.

Roschelle, J., Rafanan, K., Estrella, G., Nussbaum, M. & Claro, S. (2010). « From handheld collaborative tool to effective classroom module : Embedding CSCL in a broader design framework. » In: *Computers & Education* 55.3, pp. 1018–1026.

Roschelle, J., Tatar, D., Chaudhury, S. R., Dimitriadis, Y., Patton, C. & DiGiano, C. (2007). « Ink, improvisation, and interactive engagement : Learning with tablets. » In: *Computer* 9, pp. 42–48.

Sobreira, P. (2014). « T2/ediT2 : a flexible and easy-to-use model/system for editing and operationalizing learning scenarios. » PhD thesis. Université de Grenoble.

Sobreira, P. & Tchounikine, P. (2015). « Table-based representations can be used to offer easy-to-use, flexible, and adaptable learning scenario editors. » In: *Computers & Education* 80, pp. 15–27.

Soller, A., Martínez, A., Jermann, P. & Muehlenbrock, M. (2005). « From Mirroring to Guiding : A Review of State of the Art Technology for Supporting Collaborative Learning. » In: *Int. J. Artif. Intell. Ed.* 15.4, pp. 261–290.

Stein, M. K., Engle, R. A., Smith, M. S. & Hughes, E. K. (2008). « Orchestrating productive mathematical discussions : Five practices for helping teachers move beyond show and tell. » In: *Mathematical thinking and learning* 10.4, pp. 313–340.

Tchounikine, P. (2008). « Operationalizing macro-scripts in CSCL technological settings. » en. In: *International Journal of Computer-Supported Collaborative Learning* 3.2, pp. 193–233.

Tchounikine, P. (2009). *Précis de recherche en ingénierie des EIAH*.

Tchounikine, P. (2013). « Clarifying design for orchestration : Orchestration and orchestrable technology, scripting and conducting. » In: *Computers & Education* 69, pp. 500–503.

Tempier, F. (2016). « New perspectives for didactical engineering : an example for the development of a resource for teaching decimal number system. » In: *Journal of Mathematics Teacher Education* 19.2, pp. 261–276.

Weinberger, A., Clark, D., Dillenbourg, P., Diziol, D., Sampson, V., Stegmann, K., Rummel, N., Hong, F., Spada, H., McLaren, B., et al. (2007). « Orchestrating learning activities on the social and the cognitive level to foster CSCL. » In: *Proceedings of the 8th iternational conference on Computer supported collaborative learning*. International Society of the Learning Sciences, pp. 38–47.

Wong, L.-H. & Looi, C.-K. (2011). « What seams do we remove in mobile-assisted seamless learning ? A critical review of the literature. » In: *Computers & Education* 57.4, pp. 2364–2381.

Zurita, G. & Nussbaum, M. (2004). « Computer supported collaborative learning using wirelessly interconnected handheld computers. » In: *Computers & Education* 42.3. 42, pp. 289–314.

ANNEXES

A

ANALYSE DES TRAVAUX

Nous proposons d'analyser notre travail par rapport à la grille d'analyse construite par Tchounikine (Tchounikine, 2009). Le résultat de ce travail d'analyse est fourni dans la TABLE 19 ci-après.

Axe d'analyse	Analyse des travaux
Nature des travaux	Projet de recherche visant à modéliser l'orchestration et à construire un framework pour supporter le développement de technologies d'orchestration.
Cadre théorique de référence	Référence à une modélisation d'un scénario pédagogique (Kobbe et al., 2007), à une représentation en table pour l'édition de scénario pédagogique (Sobreira, 2014), et à une modélisation de l'orchestration (Tchounikine, 2013) pour identifier les éléments de notre modèle d'orchestration et les propriétés du framework Chao.
Type de résultat recherché	Conception et développement d'un framework logiciel supportant le développement logiciel de technologies d'orchestration sur tablettes.
Façon dont est considérée l'EIAH au sein de la SPI	Support à l'enseignant lors de l'orchestration d'un scénario pédagogique en classe.
Finalités de l'EIAH	Fournir une base logicielle facilitant le travail de développement d'une technologie d'orchestration sur tablette.
Approche de conception	Approche participative dans un premier temps pour concevoir les interfaces et outils de *monitoring*, puis approche linéaire pour la conception des autres composants du framework.
Acteurs impliqués	Implication de chercheurs, d'enseignants, et d'élèves dans la construction du framework et des applications-élève.
Contexte et historique du projet	Ce travail s'appuie sur des travaux antérieurs concernant l'édition de scénario pédagogique, et propose de reprendre ces résultats dans le cadre du développement de notre framework d'orchestration.
Niveau d'analyse des propriétés du logiciel	Le logiciel est étudié au niveau des éléments logiciels structurants qu'il propose au développeur chargé de l'instanciation ; au niveau des fonctionnalités qu'il propose et qui sont conçues pour aider les enseignants dans leurs tâches d'orchestration ; et au niveau des interfaces qui sont conçues pour être faciles à utiliser.
Nature des traitements informatiques	Les traitements informatiques concernent la création de messages MQTT envoyés aux élèves, le traitement des messages MQTT reçus de la part des élèves, le traitement des informations récupérées de la base de données, et la visualisation des composants du scénario, des lignes de scénario créées, et des informations de *monitoring*.
Niveau de réalisation du logiciel créé	Le logiciel est un prototype de recherche.

Table 19 – Analyse des travaux par rapport à la grille d'analyse proposée par Tchounikine (Tchounikine, 2009).

B

ACRONYMES UTILISÉS

API Application Programming Interface
CHAO orCHestration of Activities in classrOoms
EIAH Environnement Informatique pour l'Apprentissage Humain
HTTP HyperText Transfer Protocol
JSON JavaScript Object Notation
MQTT Message Queuing Telemetry Transport
MVP Model-View-Presenter
REST REpresentational State Transfer
SDK Software Development Kit
TEL Technology-Enhanced Learning
UML Unified Modeling Language
XML eXtensive Markup Language