



HAL
open science

Chao: a framework for the development of orchestration technologies for technology-enhanced learning activities using tablets in classrooms

Patrick Wang, Pierre Tchounikine, Matthieu Quignard

► To cite this version:

Patrick Wang, Pierre Tchounikine, Matthieu Quignard. Chao: a framework for the development of orchestration technologies for technology-enhanced learning activities using tablets in classrooms. *International journal of technology enhanced learning*, 2018, 10 (1/2), pp.1-21. 10.1504/IJ-TEL.2018.10008583 . hal-01655616

HAL Id: hal-01655616

<https://hal.science/hal-01655616>

Submitted on 10 Jan 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Chao: a framework for the development of orchestration technologies for Technology-Enhanced Learning activities using tablets in classrooms

Patrick Wang

Laboratoire d'Informatique de Grenoble
Université Grenoble Alpes
700 Avenue Centrale, 38401 Saint-Martin d'Hères, France
Email: Patrick.Wang@imag.fr

Pierre Tchounikine

Laboratoire d'Informatique de Grenoble
Université Grenoble Alpes
700 Avenue Centrale, 38401 Saint-Martin d'Hères, France
Email: Pierre.Tchounikine@imag.fr

Matthieu Quignard

ICAR, CNRS
École Normale Supérieure
15 Parvis René Descartes, 69342 Lyon Cedex 07, France
Email: Matthieu.Quignard@ens-lyon.com

Abstract: In Technology-Enhanced Learning, orchestration technologies refer to computer systems which support teachers in the orchestration of learning applications. Due to the specificity and diversity of each learning application, the use of these orchestration technologies is often not adequate in situations they were not designed for in the first place. In this article, we tackle this issue and present the software framework Chao. This framework has been designed to provide a set of classes, methods, and user interfaces to facilitate the development of orchestration technologies for tablets. The evaluation of this framework concerns its design, the usability of its user interfaces, and its ability to be adapted for various learning applications. The results suggest that teachers found the instances of the framework useful in assisting them during their orchestration tasks, and that little work is required to instantiate the framework.

Keywords: framework; orchestration technologies; technology-enhanced learning; classroom orchestration; tablets.

Biographical notes: Patrick Wang holds a PhD in Computer Science from Université Grenoble-Alpes, France. His research interests are related to Technology-Enhanced Learning and to Human-Computer Interactions. He is currently a Research Associate at the Knowledge Media institute, The Open University.

Pierre Tchounikine (<http://membres-liglab.imag.fr/tchounikine>) is a computer scientist with an interest in Technology Enhanced Learning and, more specifically, orchestration of learning activities and CSCL; he is a Professor at the

P. Wang, P. Tchounikine, and M. Quignard

University of Grenoble-Alpes (France).

Matthieu Quignard is senior researcher in cognitive science and computational linguistics at the French national research agency (CNRS). Currently working at ICAR laboratory in Lyon, his major research interests are related to cognitive and linguistics aspects of argumentation and collaboration.

1 Introduction

In Technology-Enhanced Learning (TEL), orchestration can be defined as “how a teacher manages in real time multi-layered activities in a multi-constraints context” (Dillenbourg, 2013). Orchestration originates from the difficulties a teacher might face while designing and managing a learning situation. In particular, Prieto et al. (2015) identified the occurrence of complex orchestration tasks performed by teachers in precise situations. These “high-load episodes” happened while allocating new tasks for students, assessing the student work, and providing explanations.

Formal learning settings, and especially primary school classrooms, are eager to provide students with mobile devices. This phenomenon also led to new orchestration issues that teachers might have to face. Firstly, it can be difficult to design or customise learning activities if the learning applications used by their students do not allow for it. Secondly, teachers often have few or no technological means of their own to monitor and control the flow of the activities.

One way to tackle these issues is to design systems for both teachers and students. By working in a synergistic fashion, these systems can provide teachers with support during their orchestration tasks and offer students a dedicated virtual learning environment. One example is SceDer (Niramitranon et al., 2010). It allows teachers to design learning scenarios, requires students to use GroupScribbles (Roschelle et al., 2007) to perform their parts of the scenario, and supports teachers in monitoring the student work.

This approach can prove itself efficient when designing systems for a specific setting. There is, however, a potential flaw: such systems are often hardly reusable for the orchestration of situations they were not primarily designed for. Another possible approach could be to adopt a *one-size-fits-all* design methodology, but the diversity of both learning situations and teacher practices makes it difficult to satisfy all requirements (Tchounikine, 2011).

In this project, we took on the task of following an alternative approach. It resulted in a contribution which is threefold. The main part of our contribution is the software framework Chao. The objective of the framework is to offer computer scientists the software foundations for developing tablet applications which can support teachers in the orchestration of a given learning application. This framework relies on the results of two conceptual works we conducted beforehand. The first result is a model of orchestration which has two purposes: to specify the actions teachers can undertake while orchestrating their classrooms, and to characterise the information useful to teachers for monitoring the setting. The second result is related to the design of abstract user interfaces. These abstract user interfaces organise the display of the orchestration tools proposed by the framework. The goal was to design user interfaces that are both easy to use and efficient in supporting teachers in their orchestration actions.

This paper is organised as follows. Section 2 sets the conceptual grounds of this project by providing definitions, analysing orchestration technologies found in the literature, and highlighting principles for the design of such technologies. In Section 3, we describe the model of orchestration that we built. This model supports the design of our software contribution. In Section 4, we present the software framework Chao by detailing the implementation of the orchestration tools. In Section 5, we present the results of two experimental works we conducted. In the first experiment, we studied the usability and efficiency of the orchestration tools and user interfaces with primary school teachers in their classrooms. In the second experiment, we quantified and characterised the task of instantiating the framework. We conclude this paper with a discussion on our approach and potential leads for future work.

2 Literature review

2.1 Classroom orchestration

Originating from the comparison between a teacher leading her students and a conductor leading her orchestra, the metaphor of “orchestration” has known slightly different interpretations amongst TEL researchers.

Some definitions emphasize the management of learning activities. For example, orchestration can be defined as “the process of productively coordinating supportive interventions across multiple learning activities occurring at multiple social levels” (Dillenbourg et al., 2009). This definition was later refined to consider the constraints inherent to the context in which classroom orchestration takes place (Dillenbourg, 2013). These can relate to time, institutional, or logistic constraints.

Other researchers suggested that the design of learning activities should be taken into account in addition to their management. For instance, Kollar and Fischer (2013) propose the following definition: “Orchestrating TEL means the process of creating, adapting and enacting a technology-enhanced learning scenario under complex classroom conditions”.

These small differences demonstrated the need for a unifying conceptual model. In such an attempt, Prieto et al. (2011) proposed the “5+3 framework”. This framework identifies five aspects which characterise orchestration and three more which offer insights on how to conduct research projects on orchestration. This model is articulated around the actors of orchestration (i.e., teachers) and their tasks (i.e., the design, management, assessment, and adaptation of the classroom activities). The methodological aspects accentuate the necessity to pursue the modelling and theoretical work on orchestration and to involve teachers in the research projects so as to design technologies that can be used in live conditions.

2.2 Orchestration and orchestrable technologies

In the context of TEL and classroom orchestration, the term “orchestration technology” has been used to represent technological settings designed to support the orchestration of a classroom. However, it is unclear as to what these orchestration technologies really refer to. Tchounikine (2013) proposed to make the distinction between *orchestration* and *orchestrable* technologies.

Orchestration technologies are designed for teachers so that their use can facilitate the task of orchestrating a classroom. For example, SceDer (Niramitranon et al., 2010) and MTDashboard (Martinez-Maldonado et al., 2013) are two orchestration technologies that allow teachers to create or edit a learning scenario, to control the flow of the classroom activities (e.g., by starting or stopping a task for a particular student), and to visualise the work of their students.

Orchestrable technologies are learning applications (i.e., applications allowing learners to play a learning scenario) which can be adapted on-the-go (e.g., by modifying parameters for an exercise). Two examples are GroupScribbles (Roschelle et al., 2007) and Cmate (Martinez-Maldonado et al., 2010). These technologies can be qualified as orchestrable because they operationalise a learning application, send information regarding the current state of the scenario to the teacher's computer device, and allow the change of some of their settings (e.g., the composition of groups during collective activities).

In the literature, we can also find systems which propose features from both categories. For instance, CK3 (Fong et al., 2013; Fong et al., 2015) is an application which enables students to write notes on their tablets. These notes are then displayed to the teacher and the whole class on an interactive whiteboard. The teacher can therefore see what the students are doing, and manipulate these notes to create clusters of ideas and foster discussions in the classroom. TinkerLamp is another example of such systems (Dillenbourg et al., 2013; Do-Lenh et al., 2012). Students work on their TinkerSheets, and again, a shared display shows the students' work to the teacher. TinkerKeys can also be used by teachers to stop or resume a student activity, or to test a student's work by launching a demo session.

These examples put to the fore the synergy between orchestration and orchestrable technologies: an orchestration technology is used by teachers to tune some parameters of an orchestrable technology used by students. This synergy is strong. It can be explained by the orchestration technologies being tailored to support the orchestration of a specific learning application operationalised by the orchestrable technology. A side-effect of this synergy is that these orchestration technologies cannot be easily reused in situations they were not primarily intended for.

In this paper, we wish to address this issue by designing a technological framework which could be used as a basis for the development of orchestration technologies. We focused our approach on situations in classrooms where students use tablets and designed the framework accordingly. This approach can leverage the previous works related to the use of such devices in classrooms (see for examples, Boticki et al., 2013; Dillenbourg and Evans, 2011; Jahnke et al., 2015; Looi et al., 2011; Roschelle et al., 2010; Zurita and Nussbaum, 2004).

2.3 Principles for the design of orchestration technologies

The design of our technological framework was guided by principles found in the literature related to the design of technologies for classroom orchestration.

A first design principle relates to the teacher's leadership (Dillenbourg and Jermann, 2010). This aspect suggests that teachers are responsible for conducting the classroom activities and for helping students if needed. Dillenbourg and Jermann also mention control, which is closely related to leadership. Control refers to the teacher's ability to override any decision made by the technology. This aspect reinforces the role of teachers and draws a clear separation with orchestration performed by intelligent tutoring systems.

Dillenbourg (2013) also stresses the physical dimension of orchestration. For example, the type of devices used in classrooms can influence the orchestration tasks. In one-to-one TEL situations (Chan et al., 2006), the use of tablets seems wise. Indeed, in comparison with laptops, the mobility of tablets allows for easier collaborative learning situations (Alvarez et al., 2011). It is also easier for teachers to see their students and visualise what they are doing since tablets can be laid horizontally on the students' desk.

Regarding visualisation, Kharrufa et al. (2013) are in favour of providing teachers with a private space, in opposition to using a shared display only. This private space can be a personal device such as a tablet, which would enable teachers to move around in the classroom while simultaneously having access to the orchestration technology.

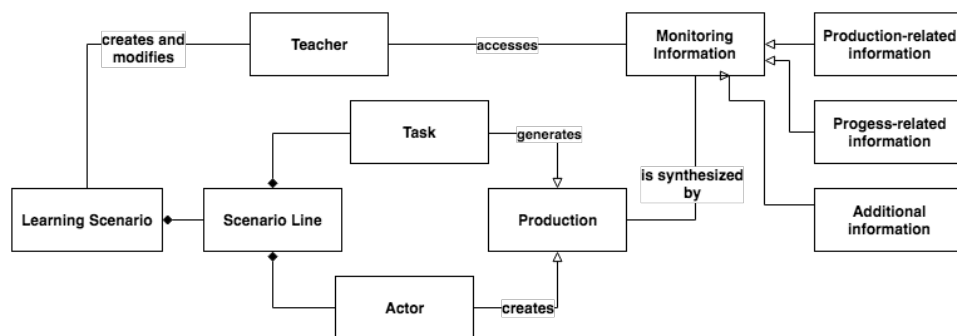
Flexibility is a recurrent design principle in the literature (Dillenbourg and Jermann, 2010; Dillenbourg, 2013; Kharrufa et al., 2013; Prieto et al., 2011). Learning scenarios must be flexible so that teachers can adapt them during the course of the activity (Dillenbourg and Tchounikine, 2007). In the same way, the learning applications that are in use must allow for this flexibility (Tchounikine, 2008; 2016).

Finally, Dillenbourg (2013) pleads for technologies that are minimalistic. This principle is in favour of reducing the teachers' orchestration load, as technologies that are too complex can deter teachers from reusing them (Roschelle et al., 2013).

3 Model of orchestration

We designed our model of orchestration (illustrated in Figure 1) to fulfil two objectives. The first objective is to define the actions teachers might have to undertake while orchestrating their classrooms. This specification will be used when designing the orchestration tools provided by the framework Chao. The second objective is to identify the types of information teachers rely on when performing these particular orchestration actions. This identification will provide insight on what data the learning applications should send and how to efficiently display this data to teachers.

Figure 1 The model of orchestration which describes primo-scripting, runtime-scripting, and monitoring actions.

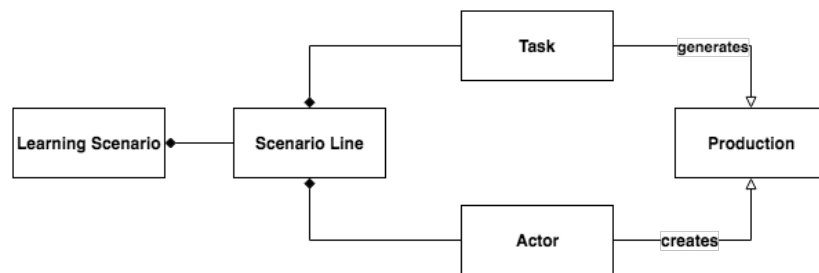


To achieve the first objective, we based our work on the identification of three types of orchestration actions: *primo-scripting* actions, *runtime-scripting* actions, and *monitoring* actions (Tchounikine, 2013). *Primo-scripting* actions concern the planning of a learning situation and the design of the corresponding learning scenario. *Runtime-scripting*

actions concern the modification and adaptation of the initial learning scenario while it is being played. Monitoring actions concern the assessment of the student work and the management of the classroom (e.g., answering questions or helping students). The combination of these three types of actions express our vision of classroom orchestration: orchestration is both digital and physical. It is digital because the teacher can make use of digital tools and information displayed on her tablet; it is physical because the teacher can move around in the classroom and directly interact with the students.

To better specify the primo-scripting and runtime-scripting actions, we also designed a model of learning scenarios. This model (Figure 2) is based on two preliminary results.

Figure 2 The model of learning scenarios which relies on a tabular structure.



The first result suggests to use tabular interfaces to represent and design learning scenarios (Sobreira and Tchounikine, 2012). With such a representation, each line of a table corresponds to an activity that students have to perform. The authors conducted two studies which showed that tabular interfaces are flexible enough to adapt to the changing conditions of a classroom activity, and have a high “pedagogical expressiveness” (Sobreira and Tchounikine, 2012; 2015). The possibility to represent a broad range of learning scenarios is an important aspect when considering the development of a software framework.

The second result concerns the modelling of Computer-Supported Collaborative Learning (CSCL) scenarios and the definition of scenario components (Kobbe et al., 2007), which we adapted to represent all sorts of learning scenarios, not only CSCL ones.

In Figure 3, we represented an example of a learning scenario designed using an instance of the framework Chao. With this tabular interface, one can perform primo-scripting actions by creating new lines, and filling them by dragging and dropping components in the table cells. Runtime-scripting actions correspond to changes made to the learning scenario designed with primo-scripting actions. These changes can correspond to the creation of new table rows or the modification of the contents of some table cells.

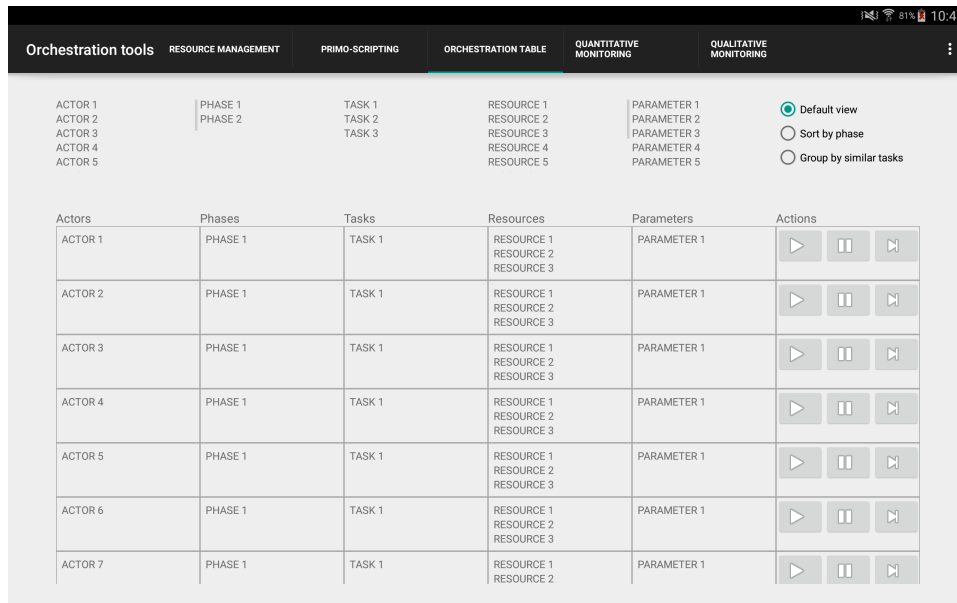
Based on this tabular structure and on the definition of a scenario component proposed by Kobbe et al., (2007), the model we propose relies on the following notions:

- A *learning scenario* is a specification of the activities that actors have to perform and of the resources and tools these actors might have at their disposal to support them in said activities.
- A *component type* is an abstract notion used to structure a learning scenario. For example, the learning scenario in Figure 3 is structured around four component types (Actor, Phase, Task, and Parameter).

Chao: a framework for the development of orchestration technologies

- A *component* is the result of the instantiation of a component type. For example, “Actor 1” is a component of type “Actor”.
- A *scenario line* is an association of components of several types, in order to specify an expected step in the learning scenario. In Figure 3, the screen shows seven scenario lines simultaneously.

Figure 3 A learning scenario displayed with a table interface. Components are listed at the top of the screen. Scenario lines are shown just below. Each line is terminated by action buttons that teachers can activate to validate runtime-scripting actions.






To specify the monitoring actions and to achieve the second objective, we dissociate “progress-related”, “production-related”, and “additional” monitoring data.

Progress-related data displays information to support teachers in their monitoring tasks at a macroscopic level (i.e., at a class-wide level). See Figure 4 (left part) for an example. Progress-related data inform about the students’ pace. It may be computed by calculating the ratio between the number of student interactions with the learning application when solving a particular task and the minimum number of interactions necessary to correctly complete this task.

Production-related data displays, in real-time, the production of each student related to their current task. By providing a direct access to the students’ productions, this type of data is designed to provide teachers with the necessary information to monitor their classrooms at a microscopic level (i.e., at an individual or group level). See Figure 4 (right part) for an example.

Finally, additional data is any data related to student work obtained from the student application which might be of particular interest for teachers. For example, additional data can correspond to information regarding the process followed by the students when creating their productions. Our framework provides components to display such data in pop-up windows.

Figure 4 Students are writing a dictation on their tablets. Progress-bars (left part) are used to display progress-related monitoring data and represent the ratio between the numbers of characters typed by each student and the total amount of characters in the dictation. The texts written by each student (right part) correspond to production-related monitoring data. Additional data can include information such as the time spent on the dictation or the amount of corrections made on a text by a student.

Student	Progress	Task / State	Student	Progress	Task / State
Jim		Writing the dictation	Jim	The scale is broken at the moment, but its used t	Writing the dictation
Mary		Writing the dictation	Mary	The scale is brocken at the moment, but it is u	Writing the dictation
Dan		Waiting for teacher validation	Dan	The scale is broken at the moment, but it's used to measure weight.	Waiting for teacher validation

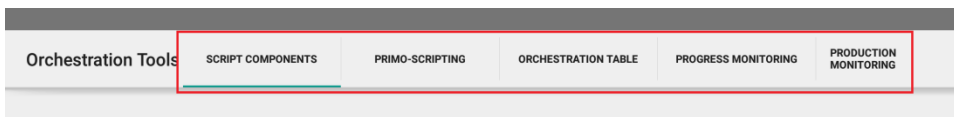
By specifying the orchestration actions and the nature of the monitoring information, the model of orchestration presented in this section proposes a specification for the abstract user interfaces and the components of the framework Chao, which we present in Section 4.

4 Implementation and user interfaces of the software framework Chao

By software framework, we understand the definition provided by Wikipedia: “A software framework is an abstraction in which software providing generic functionality can be selectively changed by additional user-written code thus providing application-specific software”. Such a definition implies that the framework Chao only implements generic user interfaces and orchestration tools. The purpose of the framework is to facilitate the implementation of orchestration technologies through the use and instantiation of its code and abstract user interfaces.

The framework Chao is designed so that, once instantiated, a teacher can have access to all the orchestration tools at all time on her tablet. The framework implements a navigation system based on tabs as depicted in Figure 5 (the active user interface is displayed below these tabs; for instance, Figure 3 illustrates the “orchestration table” view). By swiping left and right, the teacher can go from one interface to another (or from one orchestration tool to another). This possibility is particularly important as the orchestration actions are not time-sequenced but rather intertwined (Tchounikine, 2013).

Figure 5 The application presents five separate tabs (shown in the rectangular box), all of which are accessible by the teacher at all time.



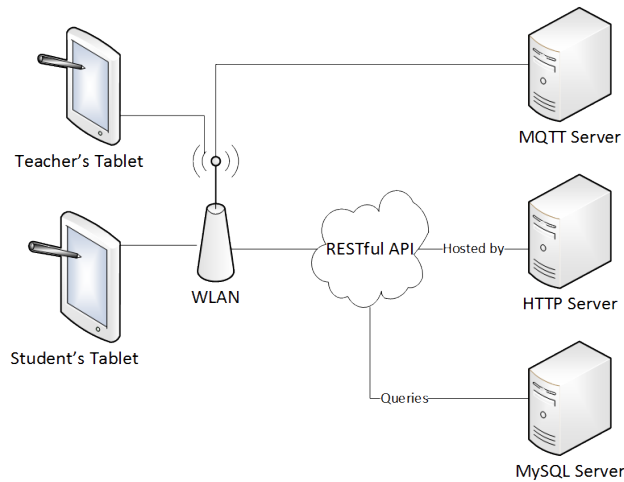
In this section, we will first present the network infrastructure designed to allow for the deployment of instances of the framework in classrooms. This infrastructure introduces third-party technologies used to assure the exchange of data between tablets and to store data. Then, we will present the abstract user interfaces we implemented for the primo-scripting, runtime-scripting, and monitoring features. Finally, we will conclude this section with a description of the steps one should follow when instantiating the

framework: the framework declares abstract classes and methods that require to be completed during the instantiation of the framework for a particular learning application.

4.1 Technical specifications

An important criterion for the acceptance of technologies in schools by teachers is the robustness of the IT infrastructure deployed in their classrooms (Ifenthaler and Schweinbenz, 2013). In order to be independent from the IT infrastructures installed in classrooms, we designed the setting depicted in Figure 6.

Figure 6 The IT infrastructure is independent from the network and machines present in the classrooms. All three servers can easily be installed on a single computer.



The system relies on a router which created a dedicated wireless local area network. The system also relies on three servers: an MQ Telemetry Transport (MQTT) server which is responsible for ensuring the exchange of data between tablets, a MySQL server which is responsible for storing in a database information related to the learning scenarios, and an HTTP server which hosts a RESTful API and allows the tablet applications to interact with the database.

Finally, tablets are handed to both teachers and students. On the teachers' tablets, an instantiated version of the framework is installed to support the orchestration of a specific learning scenario. On the students' tablets, a learning application enabling students to perform the learning scenario is installed.

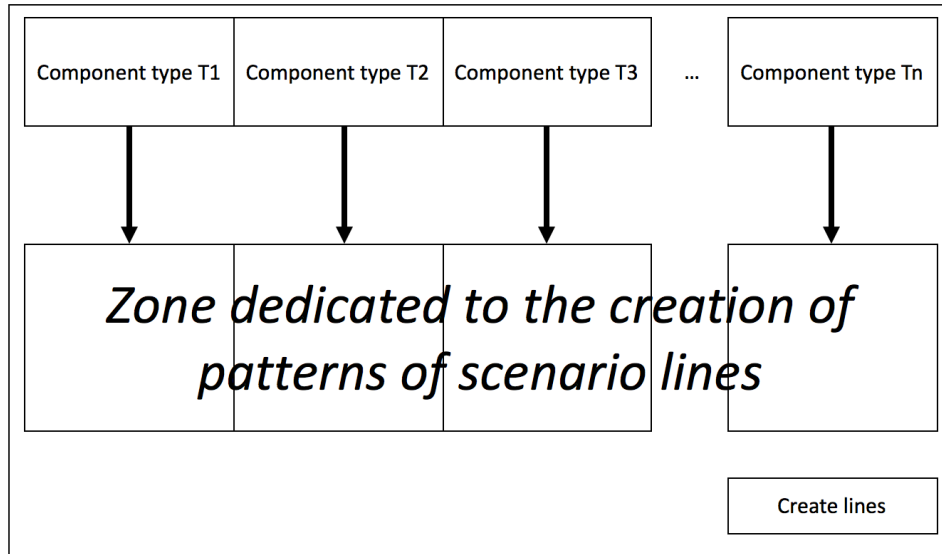
We developed the framework to function with this set-up. However, the technologies used to develop the server-side elements can be switched to other equivalent ones, provided that the framework is adapted accordingly.

4.2 Primo-scripting feature

In Figure 7, we represent the abstract user interface we designed for the primo-scripting module. This abstract user interface is based on the tabular representation for the edition

of learning scenarios (Sobreira and Tchounikine, 2012). It benefits of the flexibility of such displays: a variety of learning scenarios can be represented using this interface, as long as the component types are defined in advance. It is thus necessary to determine, before instantiating the framework, the number and names of the component types used to specify the learning scenario that will be orchestrated.

Figure 7 The abstract user interface for the primo-scripting feature.



The abstract user interface is divided between its upper and lower parts. The upper part is dedicated to displaying the components that can be used to design the learning scenario. The lower part is dedicated to the specification of scenario lines. With an instantiated version of this user interface, a teacher would be able to see all the components for each component type in the upper part of the interface. By dragging and dropping components from the top to the bottom of the screen, the teacher can create scenario lines and design the desired learning scenario. Finally, the “Create lines” button is responsible for the validation and sending of the scenario lines to the runtime-scripting user interface.

Using tabular interfaces, however, leads to a practical issue: learning scenarios can be composed of a large amount of lines, which also means a large amount of scenario lines to create. To lessen the number of scenario lines to create, we implemented a functionality to generate multiple scenario lines from a *pattern*. We define a *pattern of scenario line* as a single scenario line which contents can be equivalent to multiple distinct scenario lines. For example, if every student is to perform a task T at some point, it is possible to specify a pattern with the components *Every student* and *Task T*.

4.3 Runtime-scripting feature

Runtime-scripting actions concern the modification of the learning scenario at runtime. To support these actions, teachers must be able to see all the scenario lines created with the primo-scripting module and have the means to easily interact with these lines. We

took advantage of the usability of tabular interfaces to design the abstract user interface of the runtime-scripting module illustrated in Figure 8.

Figure 8 The abstract user interface for the runtime-scripting feature, also called the *orchestration table* interface.

Component type T1	Component type T2	Component type T3	...	Component type Tn	
Component type T1	Component type T2	Component type T3	...	Component type Tn	Action
			...		Action buttons
<i>Scenario lines created by the primo-scripting interface</i>					Action buttons
					Action buttons
					Action buttons
					Action buttons

The abstract user interface presents an organisation which is similar to the one for the primo-scripting module. The upper part lists all of the components that are available for the edition of the learning scenario. The lower part displays all the scenario lines generated from the patterns created with the primo-scripting interface. Action buttons are also appended to the end of each line to validate the modification made to the corresponding scenario line or to control the flow of the classroom. Figure 3 illustrates an instantiation of the framework with four action buttons to: (1) notify the start of a new task, (2) notify of a task being paused, (3) notify of a task being resumed, and (4) to delete a scenario line from the orchestration table. After modifying the content of a scenario line and activating the “Start” button, an MQTT message is generated and sent out by the teacher’s tablet to notify a change. The targeted student’s tablet receives the message, processes it, and alters its internal settings to comply with the runtime edition of the learning scenario.

4.4 Monitoring feature

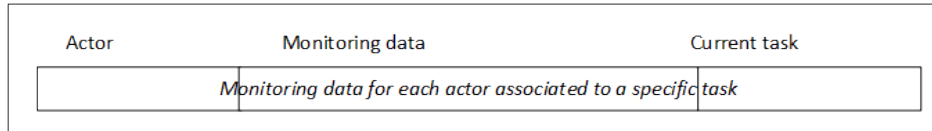
Monitoring actions concern the regulation of the learning scenario played in the classroom. These actions can range from an individual level (e.g., providing help to a student facing a particular issue) to a class-wide level (e.g., asking for the attention of the whole class before making a recap on a topic). Such actions are not innocent, and the teacher needs to be well-informed before considering taking adequate measures.

Even though we distinguish production-related monitoring data from progress-related monitoring data (a distinction that is also shown in the separate tabs in Figure 5), we designed a unique abstract user interface to display both types of monitoring information (see Figure 9). Once again, the abstract user interface is structured as a table: each row updates itself to display the latest monitoring data for each student working on their current task. By clicking on one of these rows, a pop-up window appears and displays the additional monitoring data related to an actor performing a task

This abstract user interface does not suggest a particular format for the representation of monitoring data. Depending on the learning scenario, the display of production-related

monitoring data could significantly vary. For progress-related monitoring data, however, progress bars could fit the purpose of displaying this particular type of data quite well. Figure 4 presents an illustration. The framework only requires a final decision to be made before its instantiation.

Figure 9 The abstract user interface for the monitoring feature, which displays monitoring data for each actor performing their current tasks.



4.5 Instantiation process

Chao is a framework that offers computer scientists software foundations for developing orchestration technologies. The prototypical case is as follows: it is decided to use a learning application that allows students to play a specific learning scenario on their tablets, and to instantiate Chao to support teachers in the orchestration of such a setting. The instantiation process requires the following steps:

- Specification of the learning scenario: All the components involved in the construction of the learning scenario must be identified. These components can then be separated amongst several component types (e.g., Actors and Tasks)
- Declaration of the classes representing the components: The framework facilitates this step by providing an abstract class *Component*. Using a factory design pattern, concrete classes that inherit from *Component* can easily be instantiated.
- Implementation of the classes representing the monitoring data: The representation of monitoring data is not fixed by the framework. A suitable visualisation for the monitoring data must be defined.
- Configuration of the database and web service: The database must be updated to store all the components as defined previously. The database can also be designed to save scenario lines or statuses of on-going activities. The web service is used as an application programming interface, and conveys data between the database and the instantiated version of the framework Chao.
- Implementation of utility and helper classes: The objective of such classes is to ensure that data originating from the web service can be used by the instantiated version of the framework, and vice-versa.
- Changes to the learning application: The source code of the learning application must be adapted before the application can send or receive MQTT messages.

5 Evaluation of the framework

Three criteria were studied for the evaluation of the framework. The first criterion is the possibility to instantiate the framework Chao for a wide spectrum of learning applications. The second criterion concerns the usability of the user interfaces and their efficien-

cy in supporting teachers in the orchestration of their classrooms. We focused on the evaluation of the monitoring tools. Another minor evaluation was conducted for the runtime-scripting interface. Indeed, the primo-scripting and runtime-scripting interfaces are similar, and we considered that the usability results of the former (Sobreira and Tchounikine, 2012; 2015) could be transferred to the latter. Finally, the third criterion relates to the technical difficulty of instantiating Chao.

5.1 Evaluation of the spectrum of application of the framework

To evaluate the possibility to instantiate Chao for a range of learning applications, we implemented three instantiations for three different learning applications.

The first learning application operationalises a learning scenario which is called a negotiated dictation. This scenario is traditionally played in its paper version, but a tablet-based learning application was analysed to pinpoint its specific orchestration issues (Wang et al., 2015). A negotiated dictation is composed of three phases:

1. An individual phase, in which students work individually on their dictations;
2. A collective phase, in which students form groups and collaboratively rewrite the dictation based on their own texts. In particular, they have to discuss and justify the spellings of key words by providing grammatical explanations;
3. An institutionalisation phase, in which the teacher analyses and corrects the groups' dictations.

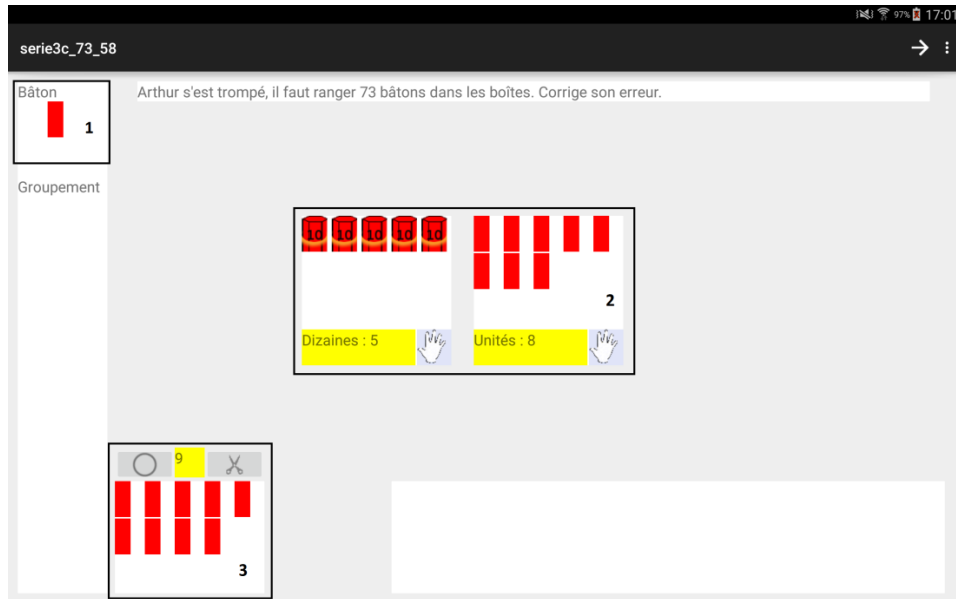
The second learning scenario is based on SimBûchettes (Brasset, 2016). This application asks K-12 students to represent numbers using sticks. In doing so, students apply their knowledge of the decimal and positional principles (Tempier, 2016). The specificity of SimBûchettes is that it allows teachers to define the behaviour of the application when a learner does not respect these principles. For example, if a student tries to place more than nine sticks to represent the unit digit, the application can warn the student of the decimal principle being violated.

The third learning scenario is based on Topeka¹, an application developed by Google. The application simply allows its user to answer thematic quizzes. The specificity of this application is that each question has several ways to display its possible answers

Negotiated dictations are collaborative learning activities, exercises with SimBûchettes are individual maths exercises, exercises with Topeka are individual assessment exercises. In these three cases, Chao could be used to easily implement orchestration tools such as the monitoring tool illustrated in Figure 4 for students working on the individual phase of a negotiated dictation. This suggests Chao presents generic features, as expected from a software framework.

Figure 10 Student interface of the SimBûchettes application. In this exercise, a student is asked to represent the number 73 from a predefined incorrect answer (6 tens and 13 units). To do so, the student can move sticks between zone 1 (which represents spare sticks), zone 2 (which represents the tens and units), and zone 3 (in which tens can be created from 10 units, and vice-versa).

¹ <https://github.com/googlesamples/android-topeka>



5.2 Evaluation of the monitoring and runtime-scripting tools

In order to evaluate the usability of the monitoring tools, we conducted experiments in four primary school classrooms with four different teachers. In these trials, we asked teachers to use an instance of the framework Chao to orchestrate negotiated dictations, and more particularly to monitor the work of their students during both the individual and collective phases. In each classroom, we divided the class in halves and ran the experiment twice, once with each half. As a result, teachers were able to use the monitoring tools during two successive sessions.

At the end of each session, teachers were asked to answer a questionnaire (using a Likert scale) regarding the usability of the monitoring user interfaces and the usefulness of each type of monitoring information provided. Overall, the four teachers found the monitoring user interfaces easy to use and were satisfied with how these interfaces helped them in assessing the work of their students. Regarding the usefulness of each type of monitoring information, Figure 11 shows the teachers' preference towards production-related monitoring data. The heterogeneity in the answers mirrors the heterogeneity in the teachers' own practices. Wang et al., (2015) provide a more complete description of the context and results of these experiments.

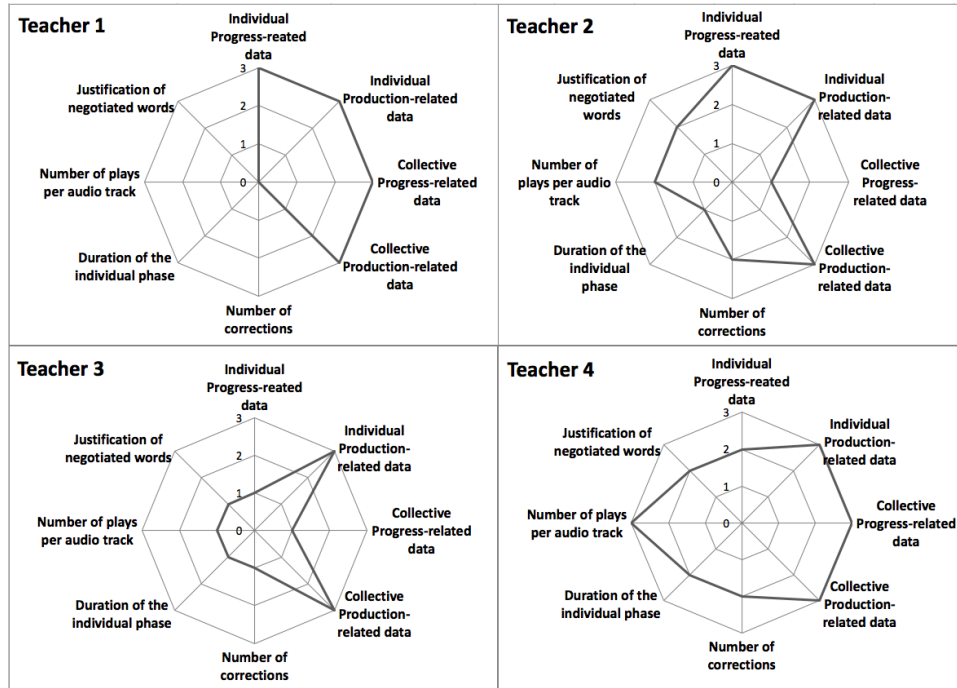
In order to evaluate the runtime-scripting orchestration tool, we analysed how a K-12 teacher used the SimBûchettes instance of the framework Chao. In this instance, runtime-scripting actions refer to the teacher attributing new exercises to students or modifying the current exercises at runtime.

During the experiment, we logged every MQTT message sent by the teacher's tablet to her students, and searched for messages that reflected runtime-scripting decisions. This work is summarised in Table 1. In total, 115 messages out of 219 are related to runtime-scripting actions, and a post-experiment interview with the teacher showed that she appreciated having the ability to modify the learning scenario during the session. This statement is even stronger knowing that she used to use the SimBûchettes learning appli-

Chao: a framework for the development of orchestration technologies

cation in her classroom without the orchestration tools offered by the framework Chao. She is thus in a position where she can reflect on and compare the two situations.

Figure 11 The results concerning the perceived usefulness of each type of monitoring information. “0” means “Not useful at all”, “3” means “Very useful”.



On another note, the results of this experiment also support our claim regarding the usability and efficiency of the monitoring tool. The relatively large amount of runtime-scripting actions is a sign that the teacher was able to quickly assess the work of her students and easily modify the learning scenario when needed.

Table 1 Number of actions that highlight runtime-scripting decisions taken by the teacher during the experiment.

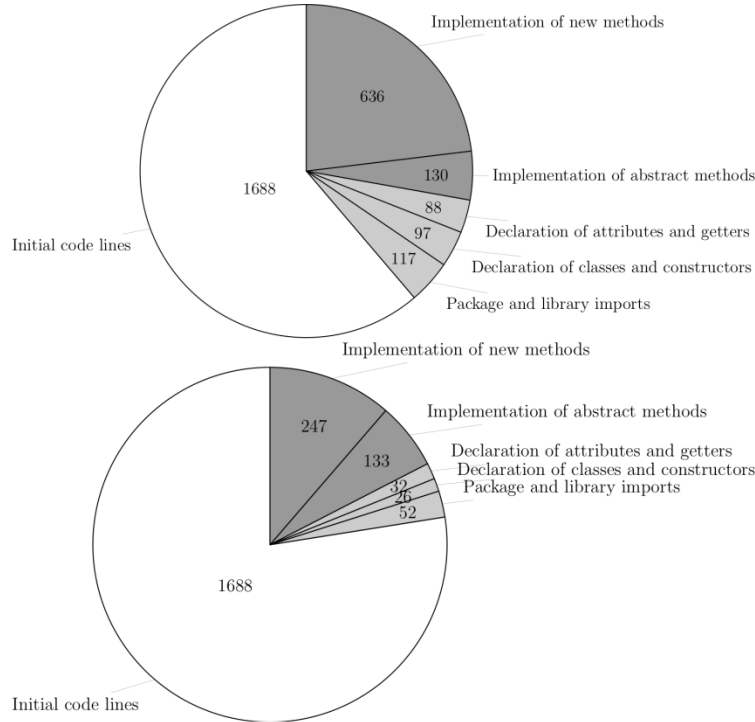
Type of message	Count
1. Start of a new task by skipping an intermediate scenario line.	36
2. Addition and/or removal of a scenario component at runtime.	40
3. Creation and addition of a scenario line at runtime.	39
4. Other type of message (not related to runtime-scripting actions).	104
Total	219

5.4 Evaluation of the instantiation task

In order to evaluate the technical difficulty of instantiating the framework, we conducted a quantitative analysis of this programming work by counting the number of code lines

that were modified or added to the framework for two instances (SimBûchettes and Topeka). A summary of this analysis is illustrated in Figure 12.

Figure 12 Top: Results for the instantiation of Chao for the orchestration of SimBûchettes-based learning scenarios. Bottom: Results for the instantiation of Chao for the orchestration of Topeka-based learning scenarios.



These code lines were put into five categories of modifications based on their roles: (1) the importation of packages and libraries; (2) the declaration of classes and implementation of their constructors; (3) the declaration of class attributes and implementation of their getters and setters; (4) the implementation of abstract methods or the modification of existing methods; and (5) the declaration and implementation of new methods.

Based on these categories, we also distinguished *straightforward* modifications (items 1, 2, and 3) from *complex* ones (items 4 and 5). We regarded some modifications as straightforward because a standard integrated development environment can automatically perform these sorts of changes. On the other hand, modifications that are considered complex require design decisions and implementation work from the developer.

The first conclusion we can draw from this figure is that the framework (in white) provides more than half of the code lines for both instances. This result comforts us in the design of the framework being well-founded, as it appears to be a good basis for the development of orchestration technologies.

If we consider the initial framework plus the straightforward modifications (in light grey), we can see that the sum of these code lines amount to around (respectively, more

than) 75% in the first case (respectively, in the second case). Thus, the complex programming work of the developer represents only a quarter of the total amount of code lines. Again, this result suggests that the framework presents a rather complete set of classes and methods which reduces the concrete amount of code lines a developer has to write when implementing orchestration technologies using Chao rather than from scratch.

If we consider the total amount of modifications, the complex modifications (in dark grey) represents 71.7% (respectively, 77.6%) of the changes for the instantiation of Chao for SimBûchettes (respectively, Chao for Topeka). When taking a closer look at these modifications, we can observe that they are mostly implemented to allow for the exchange of information with SimBûchettes or Topeka, and to process and display this information to the teacher. A conclusion we can draw from this analysis is that the developer's instantiation work is driven by the specificity of the learning application. Indeed, the majority of complex modifications can be found in classes used to exchange MQTT messages, and to format and display monitoring data to the teacher.

We draw mainly two conclusions from these analyses. The first conclusion is that the framework Chao seems to offer a complete set of classes and methods and only requires minimal adaptation work to make it compatible for the orchestration of a particular learning application. The second conclusion is that the difficult part of the instantiation work mostly concerns the processing and display of monitoring data. This suggests that the more complex the monitoring data is, the more code lines are necessary to instantiate the framework.

6 Discussion and conclusion

In this article, we presented the software framework Chao. This framework is the result of our approach towards the design of orchestration technologies: it is an alternative to ready-to-use orchestration tools specific to one learning application (which are difficult to develop), and to one-size-fits-all orchestration tools (which are often too generic). The objective of this framework is to serve as a software basis that can be instantiated at limited development cost to support the orchestration of a given learning application. The design of the framework was divided into two aspects: the construction of a model of orchestration, and the design of abstract user interfaces to facilitate the taking of orchestration actions. The goal of these contributions is to better describe these orchestration actions, and to provide user interfaces and a library of tools that can facilitate and guide the development of orchestration technologies for tablets.

The instantiation of the framework to support the orchestration of scenarios played on three different learning applications suggests that the model and the framework have a wide spectrum of use and a little instantiation cost. The evaluation of the table-based user-interfaces confirm they are easy to use by first-time users and are efficient in supporting teachers during the orchestration of a learning scenario.

Although the experiments in classrooms showed that teachers were satisfied with the instances of the framework, their use was only temporary. Our work could thus benefit from longer experiments which could also put into play other learning applications. The results of such experiments could prove useful in validating our model of orchestration and the abstract user interfaces, and also in emphasizing the importance of the minimalism, flexibility, and physicality design principles (Dillenbourg, 2013) that we applied while designing the framework.

Finally, we chose to describe the instantiation task of the framework quantitatively, using as a metric the number of code lines added or modified. Moreover, we developed all the instances of the framework Chao as presented in this article. A complementary study, which would involve external developers, could provide additional insight regarding the difficulty of this programming task.

Acknowledgements

This work was partially funded by a grant from the ARC6 action of the Rhône-Alpes region, France. The authors are grateful to the Aslan (ANR-10-LABX-0081) of Université de Lyon, for its financial support within the program “Investissements d’Avenir” (ANR-11-IDEX-0007) of the French government operated by the National Research Agency (ANR). The authors would like to thank the teachers that participated in the design of the framework Chao and in the evaluation of its instances.

References

- Alvarez, C., Brown, C. and Nussbaum, M. (2011) ‘Comparative study of netbooks and tablet PCs for fostering face-to-face collaborative learning’, *Computers in Human Behavior*, Vol. 27 No. 2, pp. 834-844.
- Boticki, I., Wong, L. H. and Looi, C. K. (2013) ‘Designing Technology for Content-Independent Collaborative Mobile Learning’, *IEEE Transactions on Learning Technologies*, Vol. 6 No. 1, pp. 14-24.
- Brasset, N. (2016) ‘Simulation du matériel de numération « Bûchettes »’. Paper presented at 43^{ième} Colloque Copirelem, 14-16 June 2016, Le-Puy-En-Velay, France.
- Chan, T. W. et al (2006) ‘One-to-one technology-enhanced learning: an opportunity for global research collaboration’, *Research and Practice in Technology Enhanced Learning*, Vol. 1 No. 1, pp. 3-29.
- Dillenbourg, P. (2013) ‘Design for classroom orchestration’, *Computers & Education*, Vol. 69, pp. 485-492.
- Dillenbourg, P. and Evans, M. (2011) ‘Interactive tabletops in education’, *International Journal of Computer-Supported Collaborative Learning*, Vol. 6 No. 4, pp. 491-514.
- Dillenbourg, P., Järvelä, S. and Fischer, F. (2009) ‘The Evolution of Research on Computer-Supported Collaborative Learning’, in Balacheff, N. et al (Eds), *Technology-Enhanced Learning*, Springer Netherlands, pp. 3-19.
- Dillenbourg, P. and Jermann, P. (2010) ‘Technology for Classroom Orchestration’, in Khine M.S. and Saleh I. M. (Eds), *New Science of Learning*, Springer New York, pp. 525-552.
- Dillenbourg, P. and Tchounikine, P. (2007) ‘Flexibility in macro-scripts for computer-supported collaborative learning’, *Journal of Computer Assisted Learning*, Vol. 23 No. 1, pp. 1-13.
- Dillenbourg, P. et al (2013) ‘Classroom Orchestration: The Third Circle of Usability’ in *CSCL 2013: To See the World and a Grain of Sand: Learning Across Levels of Space, Time, and Scale*, International Society of the Learning Sciences, Madison, USA, pp. 510-517.
- Do-Lenh, S. et al (2012) ‘TinkerLamp 2.0: Designing and Evaluating Orchestration Technologies for the Classroom’, in *ECTEL 2012: 21st Century Learning for the 21st Century Skills*, Springer, Saarbrücken, Germany, pp. 65-78.
- Fong, C. et al (2013) ‘Common Knowledge: Orchestrating Synchronously Blended F2F Discourse in the Elementary Classroom’ in *CSCL 2013: To See the World and a Grain of Sand*:

Chao: a framework for the development of orchestration technologies

Learning Across Levels of Space, Time, and Scale, International Society of the Learning Sciences, Madison, USA, pp. 26-29.

- Fong, C., et al (2015) 'The 3R Orchestration Cycle: Fostering Multi-Modal Inquiry Discourse in a Scaffolded Inquiry Environment' in *CSCL 2015: Exploring the material conditions of learning*, International Society of the Learning Sciences, Gothenburg, Sweden, pp. 39-46.
- Ifenthaler, D. and Schweinbenz, V. (2013) 'The acceptance of Tablet-PCs in classroom instruction: The teachers' perspectives', *Computers in Human Behaviour*, Vol. 29 No. 3, pp. 525-534.
- Jahnke, I. et al (2015). 'Changing Teaching and Learning Practices in Schools with Tablet-Mediated Collaborative Learning (#TMCL2015): Nordic, European and International Views' in *CSCL 2015: Exploring the material conditions of learning*, International Society of the Learning Sciences, Gothenburg, Sweden, pp. 888-893.
- Kharrufa, A. et al (2013) 'Extending tabletop application design to the classroom' in *ITS 2013: ACM international conference on Interactive tabletops and surfaces*, ACM, St. Andrews, Scotland, pp. 115-124.
- Kobbe, L. et al (2007) 'Specifying computer-supported collaborative scripts', *International Journal of Computer-Supported Collaborative Learning*, Vol. 2 No. 2-3, pp. 211-224.
- Kollar, I. and Fischer, F. (2013) 'Orchestration is nothing without conducting – But arranging ties the two together!: A response to Dillenbourg (2011)', *Computers & Education*, Vol. 69, pp. 507-509.
- Looi, C. K. et al (2011) '1:1 mobile inquiry learnin experience for primary science students: A study of learning effectiveness', *Journal of Computer Assisted Learning*, Vol. 27 No. 3, pp. 269-287.
- Martinez-Maldonado, R. et al (2013) 'MTClassroom and MTDashboard: supporting analysis of teacher attention in an orchestrated multi-tabletop' in *CSCL 2013: To See the World and a Grain of Sand: Learning Across Levels of Space, Time, and Scale*, International Society of the Learning Sciences, Madison, USA, pp. 320-327.
- Martinez-Maldonado, R., Kay, J. and Yacef, K. (2010) 'Collaborative concept mapping at the tabletop' in *ITS 2010: ACM international conference on Interactive tabletops and surfaces*, ACM, Saarbrücken, Germany, pp. 207-210.
- Niramitranon, J., Sharples, M. and Greenhalgh, C. (2010) 'Orchestrating Learning in a One-to-One Technology Classroom', in Khine M. S. and Saleh I. M. (Eds), *New Science of Learning*, Springer New York, pp. 451-467.
- Prieto, L. P. et al (2011) 'Orchestrating technology enhanced learning: a literature review and a conceptual framework', *International Journal of Technology Enhanced Learning*, Vol. 3 No. 6, pp. 583-598.
- Prieto, L. P., Sharma, K. and Dillenbourg, P. (2015) 'Studying Teacher Orchestration Load in Technology-Enhanced Classrooms' in *ECTEL 2015: Design for Teaching and Learning in a Networked World*, Springer International Publishing, Toledo, Spain, pp. 268-281.
- Roschelle, J., Dimitriadis, Y. and Hoppe, U. (2013) 'Classroom orchestration: Synthesis', *Computers & Education*, Vol. 69, pp. 523-526.
- Roschelle, J. et al (2010) 'From handheld collaborative tool to effective classroom module: Embedding CSCL in a broader design framework', *Computers & Education*, Vol. 55 No. 3, pp. 1018-1026.
- Roschelle, J. et al (2007) 'Ink, Improvisation, and Interactive Engagement: Learning with Tablets', *Computer*, Vol. 40 No. 9, pp. 42-48.
- Sobreira, P. and Tchounikine, P. (2012) 'A model for flexibly editing CSCL scripts', *International Journal of Computer-Supported Collaborative Learning*, Vol. 7 No. 4, pp. 567-592.
- Sobreira, P. and Tchounikine, P. (2015) 'Table-based representations can be used to offer easy-to-use, flexible, and adaptable learning scenario editors', *Computers & Education*, Vol. 80, pp. 15-27.

P. Wang, P. Tchounikine, and M. Quignard

- Tchounikine, P. (2008) 'Operationlizing macro-scripts in CSCL technological settings' *International Journal of Computer-Supported Collaborative Learning*, Vol. 3 No. 2, pp. 193-233.
- Tchounikine, P. (2011) 'Computer Science and Educational Software Design - A Resource for Multidisciplinary Work in Technology Enhanced Learning', Springer-Verlag Berlin Heidelberg.
- Tchounikine, P. (2013) 'Clarifying design for orchestration: Orchestration and orchestrable technology, scripting and conducting', *Computers & Education*, Vol. 69, pp. 500-503.
- Tchounikine, P. (2016) 'Designing for Appropriation: A Theoretical Account', *Human-Computer Interaction*, pp. 1-41. (in press, downloadable from <http://www.tandfonline.com/doi/full/10.1080/07370024.2016.1203263>)
- Tempier, F. (2016) 'New perspectives for didactical engineering: an example for the development of a number resource for teaching decimal number system', *Journal of Mathematics Teacher Education*, Vol. 19 No. 2, pp. 261-276.
- Wang, P., Tchounikine, P. and Quignard, M. (2015) 'A Model to Support Monitoring for Classroom Orchestration in a Tablet-Based CSCL Activity' in *ECTEL 2015: Design for Teaching and Learning in a Networked World*, Springer International Publishing, Toledo, Spain, pp. 491-496.
- Wang, P., Tchounikine, P. and Quignard, M. (2015) 'Orchestration Challenges Raised by Transposing a Paper-Based Individual Activity into a Tablet-Based CSCL Activity: An Example' in *CSCL 2015: Exploring the material conditions of learning*, International Society of the Learning Sciences, Gothenburg, Sweden, pp. 523-528.
- Zurita, G. and Nussbaum, M. (2004) 'Computer supported collaborative learning using wirelessly interconnected handheld computers'. *Computers & Education*, Vol. 42 No. 3, pp. 289-314.