

**RoomCast: an Authoring Environment and Runtime System for Classroom
Orchestration of Digital Resources**

BY

MATTEO PALVARINI
B.S., Politecnico di Milano, Milan, Italy, 2013

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2016

Chicago, Illinois

Defense Committee:

Tom Moher, Chair and Advisor
G. Elisabeta Marai
Franca Garzotto, Politecnico di Milano

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Prof. Tom Moher for the opportunity to work on this project and the support he gave me. His valuable advice has been fundamental and his passion for this field of research a source of inspiration.

I want to thank Prof. Liz Marai for having guided me and taught me so much while working with her for a User Interface class at UIC.

I would like to thank Alessandro Gnoli for all the help and the great time spent together, both as colleagues and friends.

Thanks to Gianluca, Dario, Paolo and all the friends who have shared with me the wonderful study and life experience in Chicago.

A sincere thank you to the teachers and the students at the primary school where we tested the system built for this thesis for the kindness they showed and the great work they did from the first day we met. Thanks to Brenda as well for the great support during the pilot study.

I wish to thank, above all, my parents, Eleonora and Gabriele, my brother, Francesco, and my grandparents. Your support has been fundamental to reach this goal. This work is dedicated to you.

MP

TABLE OF CONTENTS

<u>CHAPTER</u>		<u>PAGE</u>
1	INTRODUCTION	1
1.1	The problem of classroom orchestration	1
1.2	Motivations and novelty	3
1.2.1	Pedagogical foundations	3
1.2.2	Learning Technologies foundations	5
1.2.3	HCI foundations	7
1.2.4	RoomCast pedagogical contribution	9
1.2.5	RoomCast technological contribution	11
1.3	Research question and design goals	12
1.4	Research methodology	13
1.5	Thesis overview	14
2	RELATED WORK	16
2.1	Theoretical foundations	16
2.1.1	Orchestration as a metaphor and beyond	17
2.1.2	Scripting	19
2.1.2.1	Scripting vs improvisation	22
2.1.2.2	Flexibility and constraints	23
2.1.3	Orchestrating	24
2.1.3.1	Orchestration: the core principles	24
2.1.3.2	The “5+3 aspects” framework	30
2.1.3.3	Orchestration as management of constraints	36
2.1.3.4	The “kernel and rings” model	38
2.1.3.5	The third circle of usability	40
2.1.3.6	Orchestration technology vs orchestrable technology	42
2.1.4	Scripting, orchestrating and conducting	45
2.1.5	RoomCast’s vision	46
2.2	Existing technologies for orchestration	47
2.2.1	Tentative taxonomy	47
2.2.2	Script authoring environments	49
2.2.3	Knowledge Community and Inquiry	53
2.2.4	Awareness tools	54
2.2.5	GroupScribbles	56
2.2.6	GLUEPS-AR	58
2.2.7	Multi-display environments	60
2.2.8	RoomCast’s technology overview	62

TABLE OF CONTENTS (continued)

<u>CHAPTER</u>	<u>PAGE</u>
3 ROOMCAST	64
3.1 Requirements	64
3.2 Description	66
3.3 The cable television metaphor	71
3.4 Hardware architecture	73
3.5 Software architecture	74
3.5.1 nutella framework	74
3.5.2 RoomCast interfaces	81
3.5.2.1 RoomCast channel creator	87
3.5.2.2 RoomCast package creator	91
3.5.2.3 RoomCast app for browser	97
3.5.2.4 RoomCast app for iOS	102
3.5.2.5 RoomCast teacher controls	106
3.5.3 RoomCast bot	111
3.5.4 RoomCast logging system	117
3.5.5 RoomCast nutella APIs	121
3.6 Orchestration analysis	121
4 PILOT STUDY	136
4.1 Wallcology	136
4.1.1 The context	136
4.1.2 Description	137
4.1.3 RoomCast setup	138
4.2 Method	141
4.3 Wallcology unit: design and enactment	149
4.4 Results	152
4.4.1 Orchestration analysis	154
4.4.2 Teacher apprenticeship	173
5 CONCLUSION AND FUTURE WORK	178
5.1 Conclusion	178
5.2 Future work	179
APPENDICES	183
Appendix A	184
Appendix B	190
CITED LITERATURE	193
VITA	204

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Schema for the ‘kernel and rings’ model.	41
2	“Orchestration technology” and “orchestrable technology”.	44
3	The representation of a unit of content, the channel, inside RoomCast.	67
4	Mapping of available channels over packages inside RoomCast.	69
5	Mapping of channels for Activity 2.	70
6	Mapping of channels for Activity 3.	71
7	RoomCast client app.	72
8	Overview of the software architecture of RoomCast.	75
9	Commands to create and start a new nutella application.	79
10	Landing page for a nutella application.	80
11	Overview of the channel creator interface.	88
12	Selection of a channel card to be modified.	88
13	Back side of the card.	90
14	Overview of the package creator interface.	92
15	Package creator: contextual mode with channel selected.	94
16	Package creator: detail of the dropdown menu to manage activities.	94
17	RoomCast app: login with nutella application name.	98
18	RoomCast app: login with nutella run name.	98
19	RoomCast app: login with package name.	99
20	RoomCast app: tab bar for channel selection.	100
21	RoomCast app: channel during execution.	101
22	RoomCast teacher controls: selected activity.	106
23	RoomCast teacher controls: activity in the process of being launched.	107
24	RoomCast teacher controls: a flexible grid of activities.	110
25	Summarized view of the messages exchanged between bot and interfaces.	114
26	Graphical representation of the orchestration analysis of RoomCast.	133
27	Wallcology: screen of RoomCast channel creator.	140
28	Wallcology: RoomCast login screen.	141
29	Example of simple RoomCast configuration used in the training session.	148
30	Evaluation process and data gathering techniques for the pilot study.	150
31	Wallcology design: RoomCast teacher controls.	152
32	Wallcology design: RoomCast package creator.	153
33	RoomCast design in Toronto: activity with distributed controls.	167
34	RoomCast design in Toronto: activity with centralized controls.	168

LIST OF ABBREVIATIONS

RC	RoomCast
HCI	Human-Computer Interaction
TEL	Technology-Enhanced Learning
CSCL	Computer-Supported Collaborative Learning
GS	GroupScribbles

SUMMARY

Modern whole class collectivist pedagogies focus on the development of community knowledge through activities in which young learners engage in collaborative inquiry over multi-week instructional units (1). Instruction is organized as a sequence of temporal phases. As students move from phase to phase, the resources that the children need to do their work may change. Within each phase, students serving in different roles may require specialized resources as well. The orchestration (2) of those time-varying, heterogeneous resource sets imposes substantial design and enactment challenges on teachers, and transitions often come at the cost of significant instructional time, even (or perhaps especially) when those resources are digital.

There is an ongoing debate in the learning technologies community about the theoretical definition of orchestration and how to design systems and tools to support it. This thesis starts by exploring models and theories proposed in the existing literature, which are then leveraged to present the framing and the design for a novel system for classroom orchestration. If we consider the actual implementation of technologies for orchestration in previous works about this topic, we can roughly distinguish them between those which are designed to coordinate a specific domain of interest and those more general systems created to support a wide range of pedagogies. The former includes systems such as The TinkerLamp Environment (3), a tabletop tangible simulation to enhance learning in the specific domain of logistics. The second category includes tools that support specific aspects for orchestration (such as awareness or time management), and systems oriented towards a more inclusive support for orchestration like Group Scribbles

SUMMARY (continued)

(4), a tool similar to a shared whiteboard with virtual stickers, which is both an authoring and an enactment tool for the coordination of class activities.

This thesis introduces RoomCast, an authoring environment and runtime system for the orchestration of interactive, screen-based software resources in multi-week, whole-class instructional units. Adopting a cable television metaphor, RoomCast positions the teacher as the provider of individualized “channel packages” to students, with each “channel” representing a specific pedagogical resource. Prior to instruction, teachers design the set of different “channel packages” needed within each phase and corresponding to student roles or identities. During enactment, students “log in” on individual and public devices, and are presented with a “channel lineup” based on their role in the current phase, and can “change channels” depending on their needs; teachers, instead, use an interactive tool to transition among phases, and to reallocate channel packages during a phase (if, for example, a student must play a role different than expected because of the absence of another student).

RoomCast embodies several design principles for orchestration: in accordance with the “5+3 aspects” conceptual framework (5), it includes features such as the ability to design and plan the learning activities that will be coordinated, the real-time regulation of those activities, the management of resources, time, groups and workflow and the flexibility to allow intervention of the teacher to reshape the design on-the-fly in order to adapt to emergent occurrences.

RoomCast also presents several unique aspects and contributions to the research in this field. From a pedagogical point of view, it proposes a new paradigm for the orchestration of screen-based software resources, centered on the idea of orchestrating the resource set, not the in-

SUMMARY (continued)

structional script, as most of the other technologies do. As a provider of resources rather than directed tasks, RoomCast does not enforce what students have to do at a specific time, nor it requires to define the pedagogical approach: the low-level scripting task (as opposed to the conduction work) is left to the teachers who design the content of the “channels”. In this sense, it introduces new capabilities to implement principles for orchestration based on the underlying metaphor of the content provider.

From a technological point of view, it represents a new infrastructure built from scratch with the purpose of implementing the proposed paradigm for orchestration. To do this, the system supports the integration and distribution of digital resources to an ecology of devices over multiple parallel classes and it leverages dedicated user interfaces to maximize the usability for all the actors under particular constraints imposed by the classroom setting.

In addition to describing the design and implementation of RoomCast, this thesis presents the results of an exploratory study of the use of the RoomCast authoring and runtime features by teachers and learners in three 6th grade classrooms. We trained three teachers in the use of the authoring environment and assessed their ability to use RoomCast to configure activities, roles, and resources during weekly workshops and for the enactment of WallCology (6), a unit in which students investigate persistent, public displays of simulated ecosystems. During enactment, we monitored the teachers’ use of RoomCast to manage phase transitions and real-time configuration adjustments using software instrumentation and hand-held cameras to capture their interactions with the system. We use these data to evaluate the teachers’ ability to use RoomCast to effect these changes during enactment, and to identify barriers to that use.

SUMMARY (continued)

We also conducted summative semi-structured interviews with the teachers to gain insight into their perceptions of the cable network metaphor, the usability of RoomCast, and the value of RoomCast as an orchestration tool.

Finally, we used software instrumentation and fixed-camera recording to capture student interactions with the RoomCast user interface during the WallCology unit. We use these data to characterize their use of the orchestration system (e.g. from the point of view of the frequency of channel switching) and to develop evidence of usability strengths and/or weaknesses.

From the results of the pilot study, we can conclude that this first experience provides evidence to support that RoomCast has value as an orchestration technology and to suggest possible future iterations on the design of the system.

CHAPTER 1

INTRODUCTION

This chapter provides a contextualization for the problem that will be studied in this thesis, that is classroom orchestration, together with the rationale and the motivations behind the need to investigate it. We will also anticipate the contributions and novelties that this work, and the system built with it, will introduce. Finally, we will state the questions and goals that drove this research and the methodologies that were used to validate, by means of a pilot study, the novel system proposed.

1.1 The problem of classroom orchestration

In the field of Learning Technologies, we usually refer to **technology-enhanced learning** as the subject of our studies. How can technology really *enhance* learning? As Dillenbourg (2012) suggests, we can think of three main ways, which are not mutually exclusive but can happen synergistically: the first is to use technology to get “a deeper understanding of cognition, learning and teaching”, but this obviously is reached over long periods of time; a second way is to have technologies “radically shake the educational system, by setting up a new relationship between people and knowledge”; finally, the third road is to “impacting directly education through the development of new approaches, methods, and tools” (7).

If we concentrate on the third way, i.e. on the use of technology that consists of tools and

approaches that want to enhance learning by having a direct impact on it, we can see two phenomena happening (7):

- **“Integration”**: with the evolution of teaching practices, we have seen, especially in the field of Computer Supported Collaborative Learning (CSCL), the increasing need to integrate activities at different social levels (individual, team, class-wide) (8), but also different tools and technologies used and different physical and digital spaces in which learning can take place (9).

This need for integration of multiple factors brought to increasingly complex scenarios that, in addition, “have to be managed on-the-fly by teachers” (10).

- **“Technological spread”**: the availability of technologies, including computers and Internet access, nowadays is, at least in developed countries, widespread. But why schools still don’t leverage all these capabilities, including most of the tools that teachers and students normally use outside of the school context? As Dillenbourg claims, these “tools are underused because their design neglects the numerous constraints that teachers have to cope with” (7).

In this sense we can say that classrooms are peculiar in that they are complex and unpredictable; in addition, they strongly depend on the time as a very critical resource. For this reason, as Roschelle et al. underline, “such technologies too often either add too much complexity, fail to be flexible enough in the unpredictable world of the classroom, or simply take too much time” (11).

These phenomena highlight several among a series of problems whose management has been recently expressed under the concept of **classroom orchestration**. A first definition that was provided by Fischer and Dillenbourg (2006) described this notion as “the process of productively coordinating supportive interventions across multiple learning activities occurring at multiple social levels” (2). As we will see, this is a multi-faceted concept, that will have to be carefully analyzed, taking into account several positions, models, and frameworks that have been proposed in the literature.

1.2 Motivations and novelty

The work presented in this thesis lays its foundations on several concepts, theories, and technologies belonging to different fields of study. In this section we will present these foundations from three main different domains; we will then give an overview of the specific contributions that this thesis aims to provide.

1.2.1 Pedagogical foundations

We can find pedagogical foundations about the notion of **orchestration** even before introducing it in the context of technologies for orchestration: “orchestration” as a metaphor with the meaning of “arranging to achieve a desired effect” has been used by researchers, such as Jurrow, who described teachers as “orchestrating classroom discourse because they have particular pedagogical goals, and decide who will speak, what ideas will be pursued, and when a lesson is over” (12; 13; 14). In a similar fashion, Moon defined orchestration as “the process of managing a whole learning group in such a way as to maintain progress towards the learning outcomes and improvement of practice for all” (15). In this sense, orchestration has been frequently analyzed

as developing in interplay with **improvisation**: “effective classroom discussion is improvisational because the flow of the class is unpredictable and emerges from the actions of all the participants, both teachers, and students” (16).

This first notion of orchestration presented in the Learning Sciences literature represents the need to structure the **classroom discourse** as a way to handle the instructional complexity that comes from expressing the desired pedagogies.

If we consider now the introduction of technology to enhance learning, this introduced a new level of complexity into the classroom and new challenges with respect to management and coordination of the educational activities. To understand how this process of change happened, from a pedagogical perspective we need to consider the consequences of the introduction of a few modern **learning theories** among which:

- **Constructivism** is a theory of knowledge that states that humans generate knowledge and meaning from the combination of their ideas with their experiences. Piaget (17), psychologist and philosopher that contributed to the theory, argued that individuals construct new knowledge from their experiences through processes of “accommodation” (learning that comes from the experience of failure) and “assimilation”.
- **Situated learning**, first introduced by Jean Lave and Etienne Wenger (18), “takes as its focus the relationship between learning and the social situation in which it occurs” (19). Lave and Wenger argued that learning should be considered as a social process where knowledge is collaboratively constructed within a group of learners called “community of practice” (20), that is made of people who share interests and abilities.

From these theories came the idea that learning takes place with active inquiries and exploration, and that critical is the “social interaction with peers ” (21). First, we should notice that constructivism and collaborative learning did not imply decreased importance of the role of the teacher, but instead they made the teacher’s effort to coordinate the class even more critical (7). In addition, as we will see from the perspective of the Learning technologies, these learning theories shaped how technology had to be applied in the classroom, to implement and support the increasing complexity required by the new pedagogical designs.

Finally, from a pedagogical point of view, for this work we will concentrate on whole class collectivist pedagogies, focused on the development of community knowledge through activities in which young learners engage in collaborative inquiry over multi-week instructional units (1).

1.2.2 Learning Technologies foundations

As we have seen from the previous considerations, modern learning theories have shaped the way in which technology has been progressively inserted into the classroom setting. If, together with this, we also take into account the strong influence that Weiser’s vision of **ubiquitous computing** (22) had in the field of Human-Computer Interaction (HCI), we can understand how this has inspired “a number of studies aiming to transform the classroom into a participatory learning space where students collaborate through a network of interoperable computing devices of various shapes and sizes” (23).

In this sense, in the field of Learning Technologies some relevant learning technology frameworks were introduced. Moher (2006), by coining the term **embedded phenomena** (24; 25), suggested to simulate natural phenomena at the scale of the classroom to help students, seen as a

community of practice, develop scientific thinking. Even before this, a variety of different learning environments had been proposed and built by researchers to provide learners first-person access to simulated dynamic representations (26; 27).

Embedded phenomena applications represent examples of **simulated investigations**, that is units where the learner is situated as someone who physically moves around, observes and interacts with an external spatial simulated phenomenon for the purpose of learning something about it. There is, in fact, a mapping between the physical space of the classroom and the virtual space of the simulated phenomena; in this sense the location of the devices within the physical space of the classroom is relevant because it provides access to different parts of the simulated phenomenon. An example is RoomQuake (28), in which students are required to study seismic waves originated by earthquakes that take place, virtually, inside the classroom.

On the other hand, in **participatory simulations** (29; 30; 31) learners are situated as participants inside the simulation — they are in the middle of the action, that is internal agents rather than external scientific observers. In this sense, students do not explicitly “collect” data through observation, but instead they “create” data through their actions in the simulation. From a technological perspective, this “involves input devices and/or sensing technologies so that students can influence the simulation through decision-making and role-play” (23).

The previous examples, and more in general all the collaborative learning techniques such as those studied in **CSCL** (Computer-Supported Collaborative Learning), show a tendency of the complexity of management of the classroom to increase due to the added presence of a variety of technologies and ways to use them.

The problem is evident: “teachers trying to enact learning activities in one of these technology-enhanced classrooms, will have to coordinate the different tasks that are to be performed using the variety of available technologies, be they ICT (networked computers, digital whiteboards, etc.) or not (pen and paper, blackboards, books, and the like)” (32). In other words, the intrinsic need for coordination of the classroom scenario, besides being required by the increasing complexity of the pedagogies enacted, is becoming even more critical due to the variety of available technologies.

To have these technologies make their maximum positive impact on learning (i.e. to *enhance* it), there is the need to solve not only the problem of their **integration** into everyday classroom settings, but also that of having teachers be able to easily set up, manage and orchestrate the technology. As we will see, the problem is one of having technology that is itself easy enough to be coordinated, but also an additional *layer* of technology dedicated to the orchestration of the whole scenario. As Dillenbourg puts it, “we add **orchestration technology** to support the teacher in monitoring the situation, deciding what adaptations are necessary and then performing these adaptations” (33).

Finally, this problem of orchestrating complex educational settings is made even harder if we consider that the technology has to be used under some particular constraints such as time, curriculum, limited energy and effort of the teachers and many others.

1.2.3 HCI foundations

From the point of view of the broader field of Human Computer Interaction (HCI), several ideas, concepts and technologies contributed to this work for different reasons:

- If we consider the advancements described earlier in the Learning Technologies literature, such as the notion of embedded phenomena, these reflected influences from HCI paradigms among which the already mentioned notion of **ubiquitous computing** (22), but also **tangible computing** (34), that is the idea of using physical objects as tangible user interfaces augmented with computational capabilities (examples go from early forms such as the mouse to modern technologies like interactive tabletops). Another influencing trend was that of **social computing** (35), that is the idea of incorporating sociological meaning and understanding into interface design. Both tangible and social computing, as Dourish argues (35), contributed to the notion of **embodiment**, that explores interaction as connected and contextualized to the setting in which it occurs, thus considering the physicality and social interactions proper of that space. In this sense, embedded phenomena apply this idea to the classroom setting, for example when children walk from one display to another to carry out some task or activity that correlates the space of the classroom to the actual, simulated investigation.
- Especially relevant for this work will also be the notion of “**multi-display environments**”, that is the use, in our case in the context of the classroom setting, of multiple shared displays to support the collaboration of multiple users or groups of users. These displays can be represented by a variety of devices such as mobile phones, laptops, tablets and (multi-)touch tablets.

We can find many studies in HCI on this topic: as an example, “Hawkey et al. (36) use a pen-based digital whiteboard system in combination with pen-based tablet computers

to investigate the effects of distance between collaborators and the digital whiteboard on interaction and awareness” (37). As Bligh describes them, “multi-display systems is a collective term used to describe a set of new technologies which have interesting, emerging pedagogic affordances within certain learning situations. Usually developed to support business meetings, various examples of such technologies have been adopted within education institutions due to their perceived potential for student engagement and increased group interaction” (38). As we will see, the system introduced by this thesis, RoomCast, will focus on the orchestration of screen-based software resources distributed to a variety of potentially heterogeneous displays: the notion of classroom as a multi-display environment will be critical to leverage useful capabilities to support orchestration offered by RoomCast, for example by distinguishing the set of **public displays** from that of **private displays**.

- In HCI, we can also find traces of the use of the term “orchestration” itself, such as in David et al. (39), that proposed formalisms similar to musical notation to model individual and collective activities, and in “Orchestration modeling of interactive systems” (40). As we will see, the use of the term takes inspiration from its metaphorical meaning: even if it has been used in a variety of different domains to express similar concepts, we will have to carefully disambiguate its true connotation in the specific context of education.

1.2.4 RoomCast pedagogical contribution

RoomCast aims to bring a few significant innovations and contributions to the discussion, in the field of Learning Technologies, around the concept of classroom orchestration.

From a pedagogical point of view, RoomCast is proposed as an **innovative paradigm** with respect to a **specific domain of application**, that is the **orchestration** of whole class collectivist pedagogies, focused on the development of community knowledge through activities in which young learners engage in collaborative inquiry over multi-week instructional units. We also focus on learning settings and activities that leverage multiple displays present in the classroom on a potentially heterogeneous set of devices. The system that has been built is then one of the possible concrete implementations of this paradigm.

The core idea of RoomCast is expressed through the **metaphor** from which the name of the system itself and of many key notions within it come from - that of the cable television that provides content to the users. In this sense, as we will see, the main contributions that come with this work are:

- The idea of **separating the core pedagogical concerns** from those of orchestrating the classroom scenario: RoomCast as a paradigm suggests that orchestration is about the coordination that happens, at different levels, outside of the core instructional design. RoomCast as a system, in fact, represents a technology for orchestration, that is an additional and thin layer of technology dedicated to coordinating the activities, actors, and technologies in the setting.

We will argue that this separation brings many advantages among which reduced mental load for the teachers, thus easier orchestration.

- The **implementation** of the **orchestration system** as composed by two main parts: an authoring environment to design and set up the plan for the class and a runtime system

to support the orchestration during the class period, i.e. the *enactment* phase.

As we will see, peculiar to RoomCast will also be the possibility to use the authoring capabilities at runtime, to support the need for adaptability of the instructional design to the occurrences that usually happen in classrooms.

These and other features represent unique capabilities that the system introduces to better tackle the challenges raised by the problem of classroom orchestration.

- A **pilot study** to test the contributions and innovations mentioned earlier in a real educational scenario of a primary school. The results of this evaluation will hopefully contribute to the debate about how to design for orchestration.

This study will also involve an interesting evaluation of **teachers' apprenticeship**, with respect to their understanding and learning of the orchestration system, through a dedicated set of workshops.

1.2.5 RoomCast technological contribution

RoomCast introduces technological solutions and contributions from different perspectives:

- From an HCI and Learning Technologies point of view, it represents a novel system to support classroom orchestration that presents technological solutions to problems such as the **integration and distribution of digital resources** (that in our case will be screen-based interactive software resources) in multi-display environments and over an “ecology of devices” (going from desktop computers and laptops to tablets).

- Particularly relevant for this work is also the attention to the design and implementation of effective **user interfaces**: orchestration, as we will see, represents a significant challenge in terms of **usability**, since its ultimate goal is to maximize the usability for all the users (teachers, students and also researchers/developers) and at different levels, going from **individual usability** to comprehensive **usability at classroom level**, that is considering space and social interactions.
- From the point of view of the whole system proposed, it represents a completely new **infrastructure**, built from scratch leveraging some existing technologies (e.g. communication protocols) and composed by a distributed set of components, mainly graphical interfaces, but also backend modules and databases. Being built ad-hoc for the purpose of deploying the proposed paradigm for orchestration, it will be interesting to study how each different component of the system has been designed to contribute to a specific function inside the architecture and to solve specific challenges of orchestration.

1.3 Research question and design goals

With this work, we want to propose and create a **novel system for classroom orchestration of screen-based software resources**. To do this, we will first have to study the existing frameworks, models and technologies for orchestration to have a clear understanding of the current state of the research on this topic.

The system introduced will then be considered innovative if it will be able to:

- leverage the findings from previous works and effectively embed the principles for orchestration already identified;
- implement these traits of orchestration considering the specific domain of application on which we are concentrating;
- propose new principles and design choices that can be effective for our domain and, hopefully, also as a contribution to the wider debate on the problem of generic classroom orchestration;
- provide evidence that it satisfies the characteristics mentioned above after a carefully planned and structured pilot test and evaluation study.

For these reasons, the main research question that will drive this work is:

Can RoomCast serve as an effective technology for classroom orchestration of an ecology of screen-based software resources?

1.4 Research methodology

This thesis will describe our process of development of RoomCast from the initial conceptualization to the design and implementation phase and, finally, to the pilot study that was carried out in a primary school classroom.

To achieve the goals that we have set for this work, we will leverage a **design-based research** methodology (41; 42), an approach commonly used by researchers in the fields of learning sciences and learning technologies. This methodology relies on “prototyping-testing cycles with the participation of all classroom actors” (3) and, in one of the existing formulations, was defined as “a systematic but flexible methodology aimed to improve educational practices through

iterative analysis, design, development, and implementation, based on collaboration among researchers and practitioners in real-world settings, and leading to contextually-sensitive design principles and theories” (43). In this sense, this methodology doesn’t make a clear distinction between design and research: the design process does not represent the implementation of a previously consolidated theory, but rather it is embedded as part of the research itself, with continuous exchanges between design and implementation phases.

As Dillenbourg suggests, “Design-based research is the best method for studying classroom orchestration, but we must acknowledge that the generalization remains a problem” (3), meaning that even if we iteratively improve the design, the difficulty lies in finding “methods that ‘work well’ beyond a single context (even if no method will ‘work well’ universally)” (3).

From this perspective, the work presented in this thesis has to be considered a first step, within a broader iterative design process, in which we started by gathering requirements from teachers and researchers, we then developed a first version of the system from scratch and finally analyzed its value in a real classroom context.

As we will show in Chapter 4, the pilot study of RoomCast was performed following a **qualitative research study** approach supported by data gathered from multiple sources during a multi-week enactment period of a collaborative instructional unit. What we learned from this pilot study, as will be presented in Chapter 5, will serve as a guide for future revisions of the design of this technology for classroom orchestration of screen-based software resources.

1.5 Thesis overview

The rest of this thesis is structured as follows:

- Chapter 2 presents a literature review of the works about the concept of orchestration, from the point of view of both theoretical foundations and concrete implementations;
- Chapter 3 contains the detailed description of RoomCast, analyzing both the technical implementation and the support for orchestration;
- Chapter 4 describes the pilot study that was carried out in a primary school of Chicago to test the system with respect to the research questions and challenges that the present work wants to investigate;
- Chapter 5 draws the conclusions of this work taking into account the contributions presented together with their evaluation derived from the pilot study. Hints to future work are also provided.

CHAPTER 2

RELATED WORK

In this chapter, we will perform a literature review centered around the concept of classroom orchestration. The chapter is divided into two main sections: the first is dedicated to analyzing the theoretical foundations of orchestration, from the origin of the term to the most recent models and frameworks, while the second section describes the existing technologies that implement principles for classroom orchestration.

2.1 Theoretical foundations

As we will soon see, even if the discussion around the notion of “orchestration” is relatively recent in the learning technologies community, it is a result of theories and research that came long before it. We will start by describing the origin of the term “orchestration” and the metaphor that comes with it. We will then analyze the work on *scripting* to show how it contributed to the debate about orchestration. After an in-depth review of the main models and principles that define the concept, we will try to reconcile, in a bigger picture, many of the different theories that define this multi-faceted notion of classroom orchestration. Finally, we will anticipate where the system introduced by this thesis, RoomCast, has to be positioned in this picture.

2.1.1 Orchestration as a metaphor and beyond

If we look up for the verb “orchestrate” in the Merriam-Webster dictionary, it defines it either as “to compose or arrange (music) for an orchestra” or “to arrange or combine so as to achieve a desired or maximum effect” (Merriam-Webster Dictionary, 2015).

While the first, more obvious meaning is the one directly related to the musical definition, we should keep in mind the second one, which expresses a broader concept that will turn out to be critical for our work.

Orchestration, as a **metaphor** derived from the musical definition, has been used in a wide range of different research areas: for example, in *service-oriented architectures* the term has been used to indicate the “combination of services to create higher level services and processes” (44) while in *Human Computer Interaction* David and Chalon analyzed and modelled the orchestration of interactive systems (40).

But the orchestration metaphor has also existed in the learning sciences literature a long time before it started to be applied to our specific context of technology-enhanced learning: teachers, in a way similar to music composers, have (pedagogical) goals to achieve by planning the flow of activities. In this sense, Jurow talks about “teachers orchestrating classroom discourse” (14) and Moon defines orchestration as “the process of managing a whole learning group in such a way as to maintain progress towards the learning outcomes and improvement of practice for all” (15). We can even go back and find usages of the term such as in McCutchen, who in 1994 used the term to express “how writing processes must be activated and coordinated by a control structure” (45).

But if in music orchestration refers to composing the score that will be played by an orchestra, we should notice that it doesn't have any connection with the activity of the director who is conducting the orchestra when playing. This point should give a first intuition about one of the ways in which our problem of classroom orchestration can diverge from its musical counterpart and thus from the metaphor itself.

If we consider the field of technology-enhanced learning, instead, the use of the term “orchestration” is relatively recent and, as we will see, there has been an increasing debate around its definition and clarification. Historically, we can find a first definition in the field of Computer-Supported Collaborative Learning (**CSCL**) by Fischer and Dillenbourg who, in 2006, defined “orchestration” as “the process of productively coordinating supportive interventions across multiple learning activities occurring at multiple social levels” (2).

Between that very first definition and the present days, a lot of work about orchestration has been increasingly produced. As an example, in Europe an EU-funded project, the STELLAR Network of Excellence, proposed the orchestration of learning as one of the “grand challenges” in Technology-Enhanced Learning (TEL). To convey to the reader a first idea on how this topic of research is under continuous revision and improvement, let us skip for a moment to more recent claims: Dillenbourg, who had been one of the first leading proponents of the orchestration metaphor, in 2012 stated “I prefer to drop the metaphoric claims that generate unproductive debates and to use “orchestration” as a concept on its own” (7). He also added: “Orchestration is not yet a theory but a collection of opinions. I hope we can elaborate a theory that structures the different fragments that I listed [...]” (7).

We will now describe what happened in between, that is the set of theories and models about orchestration that were formulated over time and that contributed to defining such a complex and multi-faced concept, up to the point that Dillenbourg himself preferred to stop trying to force the mapping of traits of orchestration on the correspondent ones from the musical metaphor. On the other hand, we will see that the definition of the theory, even if not yet complete, has reached a quite mature level of understanding. The present work will try to add some additional proof to clarify this early theory and to suggest a few innovative aspects.

2.1.2 Scripting

Before delving into the analysis of orchestration, we want to talk about *scripting*, which is an approach that was historically introduced before the debate about orchestration and which in many ways, as we will see, could be consider one of its main precursors in the technology-enhanced learning field, in terms of topics and principles around which it evolved.

Scripting, indeed, is a common approach used in CSCL as a way to support teachers while *designing collaborative activities*. It is interesting to notice that this term also derived from a metaphor that viewed teaching as a performance: teachers act like actors which play a predefined script, originally represented by the textbook (46; 47).

This idea of scripted learning was particularly useful as a way to *structure* both face-to-face and (later) computer-mediated collaborative learning. Its roots can be found in the learning sciences literature: in 1992 O'Donnel and Dansereau defined a **collaboration script** as “a set of instructions regarding how the group members should interact, how they should collaborate and how they should solve the problem” (48).

In particular, in the field of CSCL, scripts became a popular way in which to structure collaboration among learners in scenarios where teachers want to plan and encourage interactions that contribute to foster learning. Typical activities and interactions involve argumentation, explanations, and mutual regulation(49).

The theoretical framing of the concept of script brought to divide scripts into two main categories, based on the granularity with which they provide guidance and best practices to design group activities:

- **Micro-scripts:** they are “dialogue models, mostly argumentation models, which are embedded in the environment and which students are expected to adopt and progressively internalize. For instance, a micro-script may prompt a student to respond to the argument of a fellow student with a counter-argument” (50). Given their nature, these scripts usually take place in only a few minutes.
- **Macro-scripts:** they are “pedagogical models, i.e. they model a sequence of activities to be performed by groups” (49).

As we can notice from this definition, while micro-scripts refer to supporting learning in specific, small-scale situations, macro-scripts are related to a *large-scale management* of classroom enactment and are “more focused on the coordination of didactic methods that facilitate the generation of educationally productive interactions among learners” (51). In other words, macro-scripts deal in some way with *orchestration* at classroom level: our goal now is to clarify the concepts and theories of macro-scripts and orchestration and to understand their relation-

ship. For this reason, from now on we will always talk about macro-scripts, thus simply calling them, for simplicity, “scripts”.

Research in CSCL has proposed several conceptual frameworks to support and formalize the description and design of scripts (52; 53; 54; 55; 56). Let us consider the conceptual framework proposed by Dillenbourg (52) for the design and comparison of scripts. It states that the **specification** of a script is usually made of a sequence of **phases**, where each one is characterized by the following **attributes**:

- “**Type of task**” to be carried out
- “**Group formations**” of students and their composition
- “**Distribution of the task**” among and within groups, specifying the *roles* of the students and the *tools* they can use
- “**Type and mode of interaction**”, e.g. text-based vs. voice based, synchronous vs. asynchronous, co-located vs. remote, etc.
- “**Timing**” aspects for the phase

Several other conceptualizations and ways of specifying scripts have been proposed. As an example, many researchers in CSCL have tried to formalize collaboration scripts to be machine-understandable (57), so that they can be passed as input to systems which use that specification to “orchestrate” the enactment of the class (55).

Finally, scripting can be seen as an approach which is part of the more general field of **Learning Design** (58).

At this point, we can highlight the first and most important **problem** with scripts, a problem that has primarily to do with the **nature of teachers' enactment of classes**.

2.1.2.1 Scripting vs improvisation

“The concrete form that teachers' enactment of classes take (e.g. the words used in the discourse, the gestures, etc) is not predefined to the last word in advance, but rather it is **largely improvisational**” (59).

It is almost impossible, indeed, to specify a script at a level so low that it matches the actual final enactment. In fact, “the element of improvisation has always been present, in one way or another, in the art of teaching” (16).

Over the last decade, there has been a revival of interest about improvisation in teaching. As an example, in 2002 Humphreys suggested the parallel between teaching and the jazz improvisation performed by a jazz soloist (60). Moreover, if we stay in the musical context and we go back a few years, Hudak and Berger in 1995 said that “improvisation can be seen as a form of **real-time composition**, a mutually recursive process between the performer, the instrument and other performers or actors” (61). But if, as we said, the enactment in teaching is, in a good amount, about improvisation, which in turn can be considered as composing in real-time, then we can draw a **relationship between the design and enactment** phases of teaching. This idea will be central to define orchestration and main concepts of this thesis. In this sense, as an example, Brown and Edelson define teaching as *disciplined improvisation*, a “dynamic process involving a combination of planning and improvisation” (62).

Thus, we should think of any lesson as if it was placed “at some point in a **continuum between structure/script and flexibility/improvisation**” (63).

2.1.2.2 Flexibility and constraints

As we have just seen, teaching requires a balance between structure and improvisation. For this reason, one frequently cited drawback of CSCL scripts is that they could involve a too rigid support, bringing to the “**risk of over-scripting**” collaboration (52): constraining collaboration too much, “by inhibiting the ‘natural’ peer interaction mechanisms” could be counterproductive, since it can make scripting ineffective when unexpected events suddenly occur. As a consequence of these findings, there has been an increasing interest in having **flexible scripts**. According to Dillenbourg and Tchounikine, “scripts must be flexible, i.e. adaptable by students and/or teachers, in order not to over-constrain the pedagogic setting” (64). In order to obtain this flexibility, they propose to leverage two new notions:

- **Intrinsic constraints:** “characteristics that are crucial for the useful interactions to occur. They can be considered as the script design rationale, i.e. the principles from which the script has been generated and that must be respected in order to maintain the specific mechanisms that underlie the script”. In other words, they represent the **pedagogical intentions** for the class.

As an example, an intrinsic constraint could state that ‘each student must argue with somebody who has a different opinion’ or that ‘students must take individual decisions about a choice and then come up with a shared agreement for the group’.

- **Extrinsic constraints:** “the arbitrary implementation decisions that make up the rest of the script, and which could be changed without the script losing its sense”.

They could be represented by issues such as technological choices, contextual factors (e.g., the grading system adopted by a specific school) or arbitrary decisions (e.g., an activity could be enacted in several different ways, such as with 4 or 6 groups, and we arbitrarily choose one).

Ideally, thus, to achieve flexibility we should allow, at runtime, the **modification of the extrinsic constraints** of the script, while leaving the intrinsic ones unchanged.

We will come back to the concept of constraints when we will see how they have been applied in the specific debate about orchestration. In addition, in Section 2.2.2 we will analyze some of the several approaches to scripting and their existing implementations.

2.1.3 Orchestrating

At this point, we want to continue our review of the process of definition and formalization of the concept of orchestration. We will do it by analyzing the main models and frameworks that have been proposed in the literature and finally, hopefully, we will be able to provide a unified picture that takes into account the different contributions to define this multi-faceted and debated concept.

2.1.3.1 Orchestration: the core principles

Let us start the analysis of the core principles of orchestration by continuing from the very first definition by Fischer and Dillenbourg (2006) that we have already seen earlier:

“Orchestration is the process of productively coordinating supportive interventions across multiple learning activities occurring at multiple social levels” (2).

Dillenbourg, more recently (8), precised that orchestration refers to “**two types of interplay**”:

- the “interplay between different **activities**”
- the “interplay between **individual cognitive processes and social processes**” within the same activity

He also extended that first definition by expliciting the main different **forms of coordination** of the supportive interventions in the class, when considering activities that occur “at various **social levels** or planes (e.g. individual, group or class level), across different contexts (classroom, home, laboratory, field trips, etc.) and media (with or without computers, video, etc.)”:

- Orchestration of the **workflow of activities** at different social planes, achieved for example by using macro-scripts and including the flow of data between different activities.
- Orchestration of the different **scaffoldings** used at different social planes. The term *scaffolding* refers to many resources present in a classroom, such as teachers, peers, material, and software. Tabak used the term *synergistic scaffolding* (65) to indicate a design that leverage an integrated use of these different resources at different social levels, expressing the idea that “these scaffolds might develop synergies when they are part of an effective

overall strategy”, i.e. only if they are properly orchestrated. On the other hand, “scaffolds on different social planes and from different sources can interact negatively. For example, the scaffolding done by the teacher during whole class sessions might work against group scaffolding by a CSCL script” (65).

- Orchestration of “**self-regulation and external regulation**”. This has to do with the problem of using an “appropriate level of instructional guidance” for a specific collaborative learning situation by means of regulation mechanisms.
- Orchestration of “**individual motivation and social processes**”. There has been an increasing interest in literature towards motivation as a process to be studied not only at the individual level, but also at the social level. Some studies in particular have analyzed the experiences of students from the emotional and motivational perspectives during computer-supported collaborative learning activities and found out that “students with different socioemotional orientations interpret these novel instructional designs in ways which lead to different actual behaviours” (66; 67). More in general, “learning through collaboration is not something that just takes place whenever learners come together. In any joint venture, team members must be committed to ongoing negotiation and continuous update and review of progress and achievement. This involves both cognitive and motivational engagement” (8).
- **Adaptation and flexibility** of the activities with respect to classroom occurrences. As an example, the *degree of scaffolding* should be tuned based on **runtime occurrences**; to do this, teachers should have ways to monitor the progress of groups. As we have

already seen, if teachers are executing scripts, then these should have parameters that can be tuned at runtime (64).

- **Role of the teacher as conductor** of the orchestration. In CSCL, the principle of **socioconstructivism**, i.e. the vision that students can learn by interacting with each other, does not imply that learning becomes “teacherless”, but it “changes the role of the teacher to be less of a knowledge provider and more of a ‘conductor’ orchestrating a broad range of activities” (8).

The above list of principles, related to different forms of coordination that take place in the classroom, will be critical for our discussion about the concept of orchestration, together with its design and implementation.

If we consider the first of the above-mentioned forms of coordination (orchestration of the workflow of activities), we can easily identify it with the concept of **scripting** that we have described in Section 2.1.2: we can say that scripting, a form of coordination that was proposed and studied before the introduction of the orchestration metaphor itself, represents **one of the aspects** within the broader concept of orchestration. In the following discussion, we will understand more in depth what exact role scripting plays inside the realm of orchestration.

When talking about scripting, we described the need for lessons to have both some kind of structure (e.g. being based on a script) and also some degree of flexibility to allow improvisation and management of unexpected or unpredictable events. This is a general characteristic of classroom enactment, particularly significant in the context of collaborative learning, that we can generalize: orchestration implies a “ **combination of design/planning and real-time**

adaptation” (3). In this sense, going back to the musical metaphor, Dillenbourg clarifies that “the key difference between music orchestration and classroom orchestrations is that, when orchestrating a classroom, the score has often to be modified on the fly” (3).

In his work “Technology for Classroom Orchestration” (3), Dillenbourg also suggests **9 design factors** that represent traits or general principles of design in order to support classroom orchestration:

1. **Leadership**: “teachers act as the drivers of the whole scenario and lead the class-wide activities”. We can talk about “**teacher-centric constructivism**” to express that teachers maintain the leadership of the whole scenario even if the constructivist and collaborative approach require that students directly learn through their activities.
2. **Flexibility**: teachers are allowed to change the learning scenario at runtime, on the fly, when needed and changing decisions suggested or taken by the system.
3. **Control**: teachers should be able to maintain the level of interest and concentration needed to successfully perform activities.
4. **Integration**: it “refers to the combination, within a consistent scenario, of individual, small group and class-wide activities”. This idea is the same proposed for *integrative scripts* (54), that is scripts that include activities at different social planes (individual, group or class level) and usually represented with a notation similar to music scores, where each activity is positioned at a level corresponding to a social plane.

5. **Linearity**: as in music the notes are put in a very precise sequence, most learning scenarios are “simple sequences of activities that almost all students will perform at almost the same period”. In other words, sequentiality is normally preferred: it is easy for students to understand and it makes orchestrating easier for the teacher. However, when needed, flexibility should permit teachers to break linearity.
6. **Continuity**: “the output of an activity is used as input for subsequent activities”, i.e. the same data and data structures flow from an activity to the other ones. This data might describe groups, assignments, objects or any kind of piece of information needed to support the pedagogical content.
7. **Drama**: it refers to the way teachers orchestrate the learning scenario by alternating phases characterized by different emotions expected from students. Highly emotional moments can increase the level of engagement, while phases of tension create expectation and accumulate energy.
8. **Relevance** of each topic of study or activity with respect to the **time** dedicated to it: these two factors should be proportional between each other. As we will see, time is one of the main variables that has to be carefully orchestrated. As an example, the flexibility factor should also allow to adapt the enactment of activities based on considerations about the time variable.
9. **Physicality**: “orchestration refers to the concrete layout of the physical space in the classroom and to the physical movements of the different actors”. The teacher, as the

conductor, has to be physically engaged; the physical layout of the classroom and spaces influences how activities are performed.

This first list of design factors expressed most of the traits and principles that are usually considered in the debate around orchestration, sharing indeed many ideas with the different forms of coordination that were presented before. In the next sections of this thesis, we will analyze other formalizations and clarifications of the concept of orchestration, which will be necessary as background before introducing RoomCast and its specific vision, design, and implementation.

Let us close this section with a second, descriptive definition of orchestration by Prieto. This definition tries to elaborate more with respect to the one by Dillenbourg presented at the beginning and it takes into account many of the principles we have just seen:

“Orchestration is the complex process of coordinating a teaching/learning situation, from the point of view of the teacher. Orchestration aims to manage (or subtly guide) the different activities occurring at different educational contexts and social levels, using different resources and tools in a synergic way. The orchestration is especially critical in the transitions and concurrences between those elements, and it is often guided by a design (in the form of a script or not), that may be flexibly modified during the enactment (automated or not) of the activity, in response to emergent occurrences” (51).

2.1.3.2 The “5+3 aspects” framework

As we have started to see in the previous section, we can find in the literature related to the concept of orchestration and orchestrated learning many alternative proposals on how to frame

the concept. These works usually present series of principles or traits that are very similar to each other, with many aspects in common, but they don't seem to come up with a unique and universally accepted formalization.

In his work “Orchestrating technology enhanced learning: a literature review and a conceptual framework” (2011), Prieto adds that there are several research works in the learning technologies field that “mention orchestration and delve into some particular aspect of it, but do not give explicit definitions or conceptual frameworks for it” (5), suggesting the need for a more shared understanding and framing, to provide a tool that could be useful for researchers to analytically detect and study challenges and solutions for the problem of classroom orchestration.

Thus, Prieto introduced a framework for orchestration that became known as the “**5+3 aspects**” **framework**. The framework's idea is to “cluster the main aspects mentioned in the available literature, presenting these in thematic groups” (5). The framework includes 8 aspects. The first 5 aspects are critical for **characterizing** orchestration:

- **Design/planning:** the preparation, planning and organization of the learning activities before their enactment in class. It is often performed by the teachers. This aspect is related to instructional and learning design and it can be clearly linked to the notion of **scripting**, as we have already seen earlier.
- **Regulation/management:** “the many forms of coordination that take place during the enactment of the learning activities: classroom management, time management, group management, workflow management, and so on”.

We can notice that this aspect takes into account many factors that before the introduction of this framework had been considered only separately and in a fragmentary way. For example, it includes considerations about self-regulation and external regulation (8), but also “issues related to class, time, workflow and group management, which appear in numerous orchestration works” such as (54; 3; 68).

- **Adaptation/flexibility/intervention:** the adaptations to the designed learning activities, “to cope with unexpected or extraneous events, take advantage of emergent learning opportunities, or adapt to student learning progress” (69). This aspect relates to the view of orchestration as intervention. To achieve this, flexibility is a key design factor to overcome a too rigid design structure, as we have seen when the risk of over-scripting arises.
- **Awareness/assessment:** “the perceptual processes aimed at modeling what is happening in the learning situation, for example, students’ learning progress and actions: teacher monitoring, formative and summative assessment, peer awareness, and group awareness” (69).

This aspect is important in any learning scenario, since, as we have seen with the previous factor, interventions in response to emergent occurrences and based on the classroom context are key for a good orchestration and teachers need to be able to assess them.

Examples of how to create simple *awareness mechanisms* to improve orchestration for specific types of classrooms are shown in (70) and (3).

- **Roles of the teacher and other actors:** in the debate on orchestration, researchers mostly agree on focusing on the **perspective of the teacher**; we can refer to this approach as **teacher-centrism**, that is the view of the teacher as the conductor and guide of the orchestrated learning scenario and not as the knowledge source (71). In this sense, we have already presented before the idea that teacher-centrism and constructivism can coexist well and we can argue that the whole idea of orchestration is about making their relationship as much fruitful as possible.

This factor does not only describe the role of the teacher, but it states that the roles of all the actors in the scenario have to be clearly identified, together with their relationships, to maximize the effectiveness of the orchestration process.

Finally, since teachers are seen as guides, researchers have suggested that there can be a degree of flexibility when considering the **level of guidance** of the teacher: we have already talked about the **continuum between structure/script and flexibility/improvisation**, that brings the idea that we can tune the importance and impact of the role of the teacher. For example, we could allow for a more **learner-driven orchestration**, which has been studied in works where “learners directly affect the awareness mechanisms” (70) or where scripts are “progressively faded” (72). Moreover, some authors argue that on the students side scripts are gradually **internalized**: this happens “the more learners are acting in accordance with the script’s contents” (73). Thus, we can talk about **internal scripts** “to describe individuals’ generalized knowledge structures that come to guide their understanding of actions in a specific class of situations” (73). As a consequence, as the

internalization process grows, **external scripts** (i.e. the traditional structured scripts imposed by the teachers) can be progressively faded.

There are 3 more aspects that shape **the way** in which orchestration is performed:

- **Pragmatism/practice:** in order to design for orchestration there's the need to pragmatically consider "the intrinsic and extrinsic contextual **constraints** that the actors have to cope with in compliance with the mandatory curriculum, limited amount of time available for a lesson, need for discipline in the classroom, available economic resources, and so on" (69).

We will dedicate the next section to the analysis of the interpretation of orchestration from the perspective of the constraints imposed by authentic classroom settings. These constraints determine if a certain system, in practice, is scalable, sustainable and effectively usable by every average teacher.

- **Alignment/synergy:** the coordination, as the definition of orchestration directly suggests, of all the elements that participate to the learning scenario, that is tools, people, activities at different social planes and, in general, all the *scaffoldings* used. This concept is linked to the idea of *synergistic scaffolding* suggested by Tabak (65) and applied by many researchers to the context of orchestrated learning (8; 3).

The idea is that we can maximize the effectiveness of orchestration by aligning as many of the available scaffoldings as possible to achieve the desired learning goals.

- **Models/theories:** “the mental models that different actors have about how the scenario should be orchestrated: teachers’ pedagogical beliefs, attitudes and ideas about ‘what works’ in the classroom, researchers’ own models and theories, and even student’s internal models of how they should work within the scenario” (69).

If we consider the theories formulated by researchers, as we have seen there’s a lack of agreement on the topic of orchestration which this framework tries to solve. In Prieto’s words, “a number of models of orchestration are proposed in the literature, but they are strongly heterogeneous and very context specific. [...] A deeper theoretical analysis backed up with empirical data is needed to alleviate the lack of general theories and models that can guide researchers and practitioners in orchestrating learning” (69).

We argue, as Prieto himself does, that there are still other aspects that can influence orchestration and that will have to be discovered, clarified and added to this framing. One of them has to do with the **motivational and emotional aspects** that had already been described as a coordination factor by Dillenbourg (8). These aspects require further research to be understood in relation with the problem of orchestration and how they affect it.

Prieto derived from this framework another and more comprehensive definition of orchestration:

“Orchestration is the process by which teachers and other actors design, manage, adapt and assess learning activities, aligning the resources at their disposal to achieve the maximum learn-

ing effect, informed by theory while complying pragmatically with the contextual constraints of the setting” (51).

The “5+3 aspects” framework was also evaluated by two panels of learning technologies and CSCL researchers (one of the international experts on orchestration, another with younger researchers) (51). The purpose of this **evaluation** was to reach an agreement on the usefulness of the framework as a tool of value to support researchers in their studies and works on orchestration. These 31 learning technology researchers performed a consensus-based validation of the framework: they were presented a set of detailed questionnaires aimed at assessing the framework’s completeness and usefulness. The result was that they “valued positively the framework in terms of its completeness, and saw its value as a holistic overview or as a checklist for researchers in the field”. However, some of the experts expressed criticism towards the framework and the need to refine it, for example by giving more importance to the role of technology in it. The framework was also considered useful in general, especially for younger researchers that need to have a good and comprehensive overview on the field. All the details about the evaluation method, data gathered and results can be found in (51).

2.1.3.3 Orchestration as management of constraints

In Section 2.1.2.2 we talked about two categories of constraints that can be identified for scripts: internal constraints are the ones strongly related to the pedagogical approach that the teacher wants to follow, while extrinsic ones contain all the other constraints that are not part of the core pedagogical purpose of the script.

This distinction has been generalized to the concept of orchestration in a quite straightforward way: the same constraints that were applied to scripts, by definition, have to be considered as affecting the learning scenario as a whole and, thus, being meaningful as part of the discussion about orchestration.

But we can say more. As Dillenbourg points out, “instructional design has to cope with 3 **intrinsic constraints**: the contents to be taught (what), the learner’s characteristics (who) and how brains build knowledge (how)”. These intrinsic constraints, as we can notice, deal again with the core pedagogical intentions that characterize the learning scenario.

Other than these intrinsic constraints, instructors also have to deal with several **extrinsic constraints**. Dillenbourg highlights a few of them:

- “Time constraints”
- “Curriculum relevance”
- “Discipline constraints”
- “Assessment constraints”
- “Energy constraints” (to take into account the limited effort that teachers are able to invest)
- “Space constraints” (3)

For these reasons, Dillenbourg defines orchestration also as a “multi-constraints management problem” or as the “**management of constraints systems**” (3).

We will clarify the role of constraints after having presented the “kernel and rings” model.

2.1.3.4 The “kernel and rings” model

Based on what we have seen up to this point, we can say that orchestration has to be considered an expansion of *instructional design*: before the discussion about orchestrating the learning scenario, instructional design was mainly concerned on designing the **core pedagogical plan**, i.e. the instructional sequence and strategy for the class, part of which could be specified by what we called script. Dillenbourg suggests to consider these aspects under the term “**kernel**”: “kernel design has to solve a crucial and difficult equation with several parameters: mainly the learning objectives, the learner’s characteristics and the learning processes”. (7). Thus, since we can identify the **kernel** with **educational design**.

On the other hand, orchestration includes many things that happen around the kernel, those that Dillenbourg calls **rings**. These rings are described as activities of different nature that extend around the kernel:

- “**Emergent activities** (designed but contingent)”: they are activities that have been designed as part of the kernel, but they are also based and built upon what students have to produce or have produced in earlier phases. For this reason, these activities are not completely predictable and require special attention and support.

An example can be represented by an activity of “debriefing about student-generated content” (51).

- “**Envelope activities** (non designed but necessary)”: they are activities meant to consolidate the kernel, but they are not designed and part of it. An example can be represented by

the activity of “making students copy into their notebooks the contents of the blackboard” (51).

- “**Extraneous events** (unavoidable)”: they are unavoidable and unexpected events, such as “dropouts, latecomers, or kids copying the solution from their peers” (51). We have to notice that technology-enhanced learning increases the probability for these events, such as failures, to happen, thus making adaptation even more difficult.
- “**Infra activities** (non designed but necessary)”: they are activities that interfere with the kernel and which are not a desired part of the core pedagogical design, but they are still necessary to support the enactment.

A common example for these activities is represented by “logistic issues, such as setting up the computers for the class or opening books on the right page” (51).

As Dillenbourg puts it, referring to these rings extending from the kernel, “there is indeed a continuum of activities from those intrinsic to the scenario to activities extrinsic to learning.” (33). Thus, we can say that the kernel corresponds to **intrinsic activities**, i.e. the ones strictly related to the core pedagogical design (or *educational design*), while the rings represent **extrinsic activities**, that is activities that, at different levels, are extraneous, or in the periphery, with respect to learning.

We can now precise the **role of orchestration**, by linking this new notion of intrinsic and extrinsic activities to the concept of intrinsic and extrinsic constraints: “**orchestration involves adapting both intrinsic and extrinsic activities to both intrinsic and extrinsic constraints**” (33). We can, however, add that “what is novel in orchestration is to **care for**

the rings” (7), as opposed to before the introduction of the concept, when researchers cared exclusively for the kernel.

Figure 1 shows a graphical representation of this model, where the rings are placed around the kernel and their different distance from the kernel conveys the idea of the extent to which they are related to the core instructional design as opposed to being just events in the periphery that have to be taken care of.

2.1.3.5 The third circle of usability

The problem of orchestration has been analyzed from many different viewpoints, one of them was represented by the work of Dillenbourg et al., who suggested analyzing “classroom orchestration as a question of usability in which the classroom is the user” (74).

These authors identified three different **circles of usability**.

The first circle is related to studying how individuals interact with learning environments, that is the problem of **individual usability** that has been traditionally studied in HCI.

The second circle is related to studying the design for teams, that is the problem of **usability at team level** concerned with the collaboration tools used by learners to interact with each other and central topic of study of CSCL.

The third circle is introduced as a further abstraction, which concerns how the technology-enhanced collaborative tools and environments are **integrated at classroom level**.

To analyze usability we first have to identify a **user**: if for the first circle the user is an individual person and for the second circle a team, then for the third circle the user is represented by the classroom as a whole.

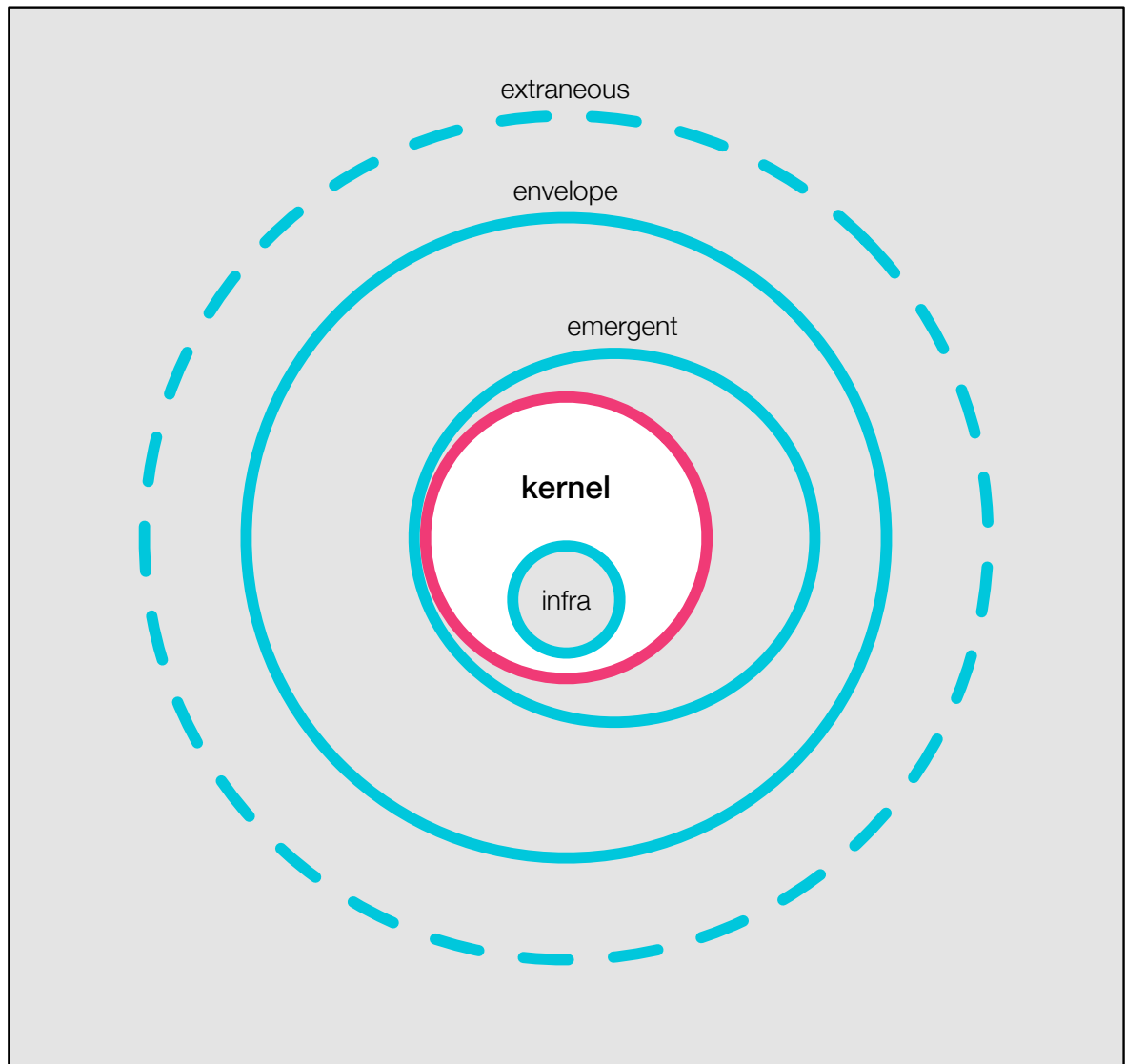


Figure 1: Schema for the 'kernel and rings' model: "educational design concerns the kernel, orchestration is about the rings".

We can then look at the **constraints** that affect usability at each circle. For the first circle, they are represented by the cognitive load of the individual, his prior experience and background, his motivation, etc. For the second circle, the constraints are the relationships between peers inside the team, the level of understanding that each team needs to carry out a task, etc. Finally, at the third circle, that is at the level of the **classroom ecosystem** (3), teachers have to cope with a series of constraints that are the same we identified before when describing intrinsic and extrinsic constraints. Thus, we can say that this third circle, concerned with **usability at the classroom level**, can be identified with the already familiar concept of **design for orchestration**. For this reason, we can consider the problem of orchestration as “a question of usability in which the classroom is the user” (74).

2.1.3.6 Orchestration technology vs orchestrable technology

An additional distinction about the concept of orchestration was made by Tchounikine in 2013, when he proposed to clarify how to design for orchestration by introducing the distinction between “orchestration technology” and “orchestrable technology” (75):

- “**Orchestration technology** is technology that achieves or supports the **activity of orchestrating**”.

This technology can be represented by tools that offer teachers support for managing the classroom, tools for monitoring or intervention, systems to manage the flow of activities, and, in general, any technology that helps dealing with part of the orchestration task.

- “**Orchestrable technology**: is technology which use can be decided or adapted (before the session and/or at run-time) by the players in charge of the orchestration (the teacher,

a system) while orchestrating the setting, in the same way that other parameters of the setting (the timing, the groups, the task, the physical space, the teaching objectives, etc.) may be adapted”.

This technology can be represented by tools “reifying in some way or another some given **pedagogical intentions**” and “whose usage is likely to create pedagogically rich events”, but still maintaining *flexibility*, i.e. allowing adaptation, tuning, and control by the teachers.

We could also see the whole system of orchestration technologies, as Sharples suggests, as an “**orchestration layer**” (76), that is a structure that acts like a middleware between the orchestrable technologies used by students for learning and the teacher who orchestrates the scenario. These concepts are simplified and depicted in Figure 2.

In addition, since the idea of adding a layer to manage orchestration automatically adds a further degree of complexity to the whole system, we will see that the concept of “**functional minimalism**” (7) will be also critical for our discussion about how to design for orchestration.

Finally, we can already notice that with these definitions we are separating the technologies to be used in the classroom between those dedicated to express and create the core pedagogical contents and those useful for coordination purposes, that is for orchestration. This distinction will be particularly useful when we will describe the vision for RoomCast and the design choices to support orchestration that were made to build the system.

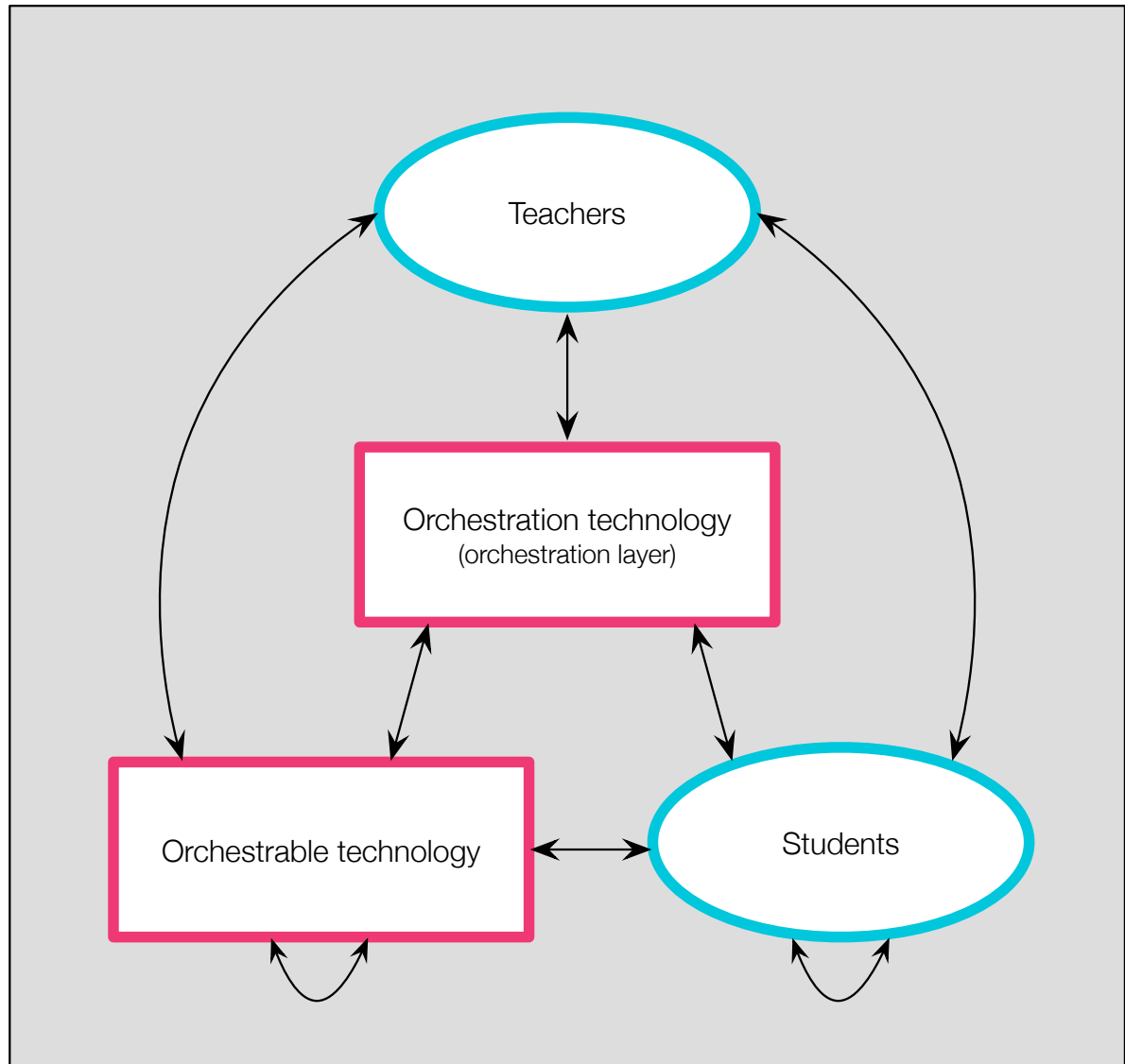


Figure 2: Graphical representation depicting the distinction between “orchestration technology” and “orchestrable technology”.

2.1.4 Scripting, orchestrating and conducting

In his clarification of the design for orchestration, Tchounikine also helps us unifying the concepts of *scripting* and *orchestrating* into a bigger picture, to explain their relationship and to understand even further the role and meaning of orchestration.

Up to this point, we have seen how orchestration has to deal with two main dimensions: a design/planning phase (pre-session management) and an adaptation/enactment phase (real-time management). Tchounikine suggests that we can further clarify this distinction: orchestration can be seen as a “**combination of scripting and conducting**” (75):

- **Scripting**, as it has been defined in Section 2.1.2, is primarily concerned with the planning phase that takes place *before* the class enactment period, but it also has a role *during* a learning session. To clarify this distinction we can introduce two further notions:
 - “**Primo-scripting**”, i.e. “defining the initial script and the associated technological decisions” (75). This is the traditional definition of scripting, which includes designing activities to reach specific pedagogical objectives, based on the adopted pedagogical principles and learning hypotheses. Thus, the intrinsic and extrinsic constraints are first set up in this phase.
 - “**Run-time scripting**”, i.e. reconsidering, during enactment, the pedagogical objectives and/or the means to achieve them. This implies reconsidering the *intrinsic constraints* of the script, e.g. “as a way to adapt to the actual performance and, for

instance, avoid a breakdown or take an opportunity” (75), and the *structural model*, i.e. the way in which scaffoldings, such as technology, are used.

- **Conducting** “is about contextually communicating directions to performers and adapting the setting components or their articulation” (75). It is exclusively referred to all the orchestration aspects that happen at runtime (during the session) and excluding those that concern reconsidering the pedagogical design and objectives (for which, in fact, we talk about run-time scripting). In other words, conducting is about adapting the direction of the class enactment “within the scope of the extrinsic constraints” (75).

2.1.5 RoomCast’s vision

Let us conclude the literature review of the debate around the concept of orchestration by anticipating what is the main idea and vision behind the system that is introduced by this thesis, RoomCast.

As we will see more in detail in Chapter 3, and as stated in the title of this thesis, RoomCast is a technology for classroom orchestration which is divided into two main components:

- an **authoring environment**, i.e. the part of the system which allows to design the distribution of pedagogical resources to the class before enactment (offline), including planning which resources will be used by which actors and what activities will be (potentially) executed;
- a **runtime system**, i.e. a system which supports orchestration during enactment, including both conduction, such as launching activities, and runtime scripting, i.e. using the

authoring environment to manage and adapt the pedagogical design choices at runtime (online).

The core idea behind RoomCast is that the system represents an *orchestration technology* that is placed inside the classroom ecosystem as an *orchestration layer* with which all the actors of the scenario interact. With this layer, which deals with the management of the aspects that Dillenbourg called “*rings*”, we wanted to **abstract** the role of the orchestration system from that of the *orchestrable technology*, which is represented by the screen-based software resources distributed to the classroom through the system and that represent the core pedagogical contents (i.e. the *kernel*).

In Chapter 3 we will describe in detail the system and its design for orchestration, while in Chapter 4 we will present the pilot study that was used to test RoomCast in a real classroom ecosystem to find evidence about its usability both at individual and classroom level and, more in general, about the effectiveness of its design for orchestration.

2.2 Existing technologies for orchestration

In this section, we want to describe a selection of the existing technologies for orchestration that have been proposed in the literature. To do this, we will first try to provide a tentative taxonomy of these technologies, with the purpose of suggesting a simple classification based on the characteristics and nature of these systems.

2.2.1 Tentative taxonomy

While, as we have described in the first part of this chapter, the theoretical definition and conceptualization of the meaning of orchestration have seen an increasing interest and variety

of positions, the actual implementation of technologies explicitly designed to support classroom orchestration has been much less explored.

Among all the proposed tools and systems, a first distinction could be made between those technologies that are **domain-dependent**, that is environments oriented towards the orchestration of specific kinds of activities, i.e. designed for specific contexts, and those more general systems created to support a wide range of pedagogies.

In the first category, we can find environments such as the **TinkerLamp** (77), “an augmented reality system designed to run tabletop tangible simulations”. It allows students in the specific domain of logistics to build a model of a warehouse: they can place small-scale shelves and other architectural elements on a table, then a camera “recognizes the visual markers on the objects, computes a model of the warehouse and [a projector] displays information on the table such as shelf contents and forklift movements” (77). In addition, to control the simulation the students use sheets called “TinkerSheets”, on which they can place tokens in specific positions that are recognized by the system, thus allowing to tune some parameters of the simulation. If we analyze the TinkerLamp environment from the point of view of the *9 design factors* for orchestration suggested by Dillenbourg and presented in Section 2.1.3.1, we can say that the use of the sheets allows to plan activities at different social planes, e.g. for different groups or as homework to be done at home and delivered later (factor 4, integration), but the teacher can also change the planned activities (factor 2, flexibility). Moreover, the workflow of activities is set (and made tangible) by passing different sheets (factor 6, continuity across activities). The

use of an additional projector to present the work of a group to the whole class also contributed to integration (factor 4) and physicality (factor 9).

We can easily notice the limitations associated to systems such as the one just presented: they satisfy a few characteristic principles of orchestration, but their application is limited to the specific context for which they were designed.

Let us now consider systems designed for orchestration of a wider range of pedagogies and applications. They can be divided into two main sub-categories: tools whose application in the classroom satisfies a specific subset of features and traits of orchestration and, on the other side, those systems that have been specifically designed to be general purpose *orchestration technologies*. Obviously, these are not to be seen as two clearly separate sets: we can identify a **continuum** of technologies and solutions that, based on the level of support for orchestration, are nearer to one category or to the other.

A few significant examples of systems from these last two categories, i.e. the ones more useful to be compared with the solution that will be introduced in this thesis, will be presented in the rest of this section. We will start by introducing common *script authoring environments*, that is systems dedicated to implementing *scripting*, thus, as we have seen, primarily concerned with planning and structuring the flow of activities.

2.2.2 Script authoring environments

Several script authoring tools have been proposed and developed in the last two decades. Before showing some of them, let us describe two common scripts that have been embedded in these environments:

- **ArgueGraph** (52) “aims to trigger argumentation by forming pairs of students with conflicting opinions” (3): first, the students individually answer questionnaires, then the system derives from them a map of opinions of the students and puts them in pairs with opposite opinions. At this point, the pairs are asked to answer again together the same questionnaire. At the end of the script there’s a debriefing session. The teacher also has a tool which keeps track of all the answers and allows her to navigate every question and answer and highlights who has changed opinion.
- **ConceptGrid** (54) “aims to trigger explanations” (3): each group of students has to produce some definitions given a set of papers; each student is assigned a subset of papers and individually submits the definitions that he has been assigned; then the group has to assemble the concepts in a map called ‘concept grid’, able to show their relationships. At the end, the teacher, with the help of a tool to navigate the grid, asks questions and explanations during a *debriefing* session.

Among the proposed script authoring tools we can find:

- **CeLS** (Collaborative e-Learning Structures) (78): “a web-based system designed to create and reuse Activity Structures”, i.e. “stages of interaction between a learner and the system”. The system provides the “ability to use learners’ products from previous stages and to conduct complex, multi-stage, structured activities”. In addition, teachers can search for re-usable, pre-designed Activity Structures, but also create new ones using basic building blocks.

The authors aim at “creating and conducting structured asynchronous collaborative activities and incorporating them in the existing instructional setting for all subjects and levels”.

- **LAMS** (Learning Activity Management System, lamsfoundation.org): an open source Learning Design system that allows to “design, manage and deliver online collaborative learning activities”. Teachers are provided with a visual authoring environment for designing the flow of activities (individual tasks, small group work or whole class activities).
- **ManyScripts** (manyscripts.epfl.ch): proposed by Dillenbourg and Hong, was defined as “a web-based environment where teachers may prepare the script they want to use with their students. Later on, the student will log in to do the different activities that compose the script”. The system supports the implementation of a pre-defined set of scripts, among which ArgueGraph and ConceptGrid and it allows for some flexibility, for example with respect to the size of groups and potential dropouts. Its drawbacks are that it can only be used with the restricted set of scripts that have been implemented, it has a high complexity, thus it is difficult to be managed by the teachers and it doesn’t support adaptation and handling of activities at runtime. As we will see, most of these lacks have been addressed by systems designed specifically to satisfy the principles for classroom orchestration.

Finally, let us show two tools that have been proposed in CSCL to specifically add flexibility to collaborative scripting systems (to avoid the risk of over-scripting):

- **Adaptive Collaboration Scripting (ACS)**: it is a framework, proposed by Demetriadis and Karakostas (79), whose core idea is to manage the intrinsic constraints of scripts separately from the extrinsic ones (which, as we have seen, is one of the most effective approaches to increase the overall flexibility of the system). As an example of this, the framework allows for changes of group formation at runtime and permits to provide additional scaffolding to novice learners. It also includes mechanisms to adapt to emergent occurrences. Unfortunately, the actual implementations of the framework are still fragmentary and limited to one script at a time.
- **WikiPlus** (80): a flexible macro-scripting engine which consists of a “modified version of a *wiki* to regulate learners’ activities” (4).

The idea of extending the basic functionalities of a wiki, that is making it more dynamic by letting non-technicians create static web content, makes the system much easier to use for teachers, who can also modify the “scripts” at runtime. In addition, the system can easily accommodate any kind of script, thus without supporting only a limited variety such as most scripting environments (although “technical knowledge is still initially required to generate wiki templates that represent a macro script”).

As a result, the system has “almost the same potential as specialized script engines”, but with “the big advantage of WikiPlus, that is the flexibility in the regulation phase” (80), which is given by the flexible nature of wikis.

2.2.3 Knowledge Community and Inquiry

“Knowledge Community and Inquiry (KCI) describes a class of pedagogical designs where students engage in carefully scripted inquiry and collaborative knowledge construction to achieve science learning goals” (81; 82; 83). The curriculum includes “aggregated observations”, “scripted collaborations within and between groups” and other complex forms of interaction. Slotta et al. (81) presented the implementation of systems that support the orchestration of KCI curricula, with the purpose of coordinating “complex collaborative inquiry designs that are not fully specified at their outset” (84), that is which depend on occurrences at enactment time such as student contributions that influence factors like groupings and flow of activities. These systems, based on the specification and adaptation of complex scripts, allow real-time coordination of “devices, tools and materials, student group and role assignments across multiple contexts” by “actively monitoring student-contributed data in real time”.

Let us look at one of these systems for KCI curricula: **EvoRoom** is a “room-sized simulation of a rainforest environment designed for a smart classroom in a high school biology curriculum in topics of biodiversity and evolution” (81). In this room, students perform collaborative inquiries in the role of researchers and make observations on their tablets; they are also grouped in teams to work together to understand connections between different species in the ecosystem. The orchestration is implemented by signaling the steps of the activities on the personal tablets, while the teacher has a dedicated tablet that allows her to control all the other tablets in the room, e.g. to “advance the room through various evolutionary time periods”, that is to manage time and pedagogical flows. In addition, intelligent agents that take into account the

data produced by the students are able to provide “contextualized instructions” about the next phases. Given these aspects, the system supports many principles for orchestration such as design/planning, management of time, groups and activities, adaptation at runtime, awareness (related to the information about the status of the enactment that is provided to the teacher on the tablet) and clear identification of the roles of the actors.

2.2.4 Awareness tools

In the “5+3 aspects” framework, one of the 5 critical aspects for characterizing orchestration was **awareness**, that is the ability of the teacher to assess the state and progression of the educational session, at individual, group and class level.

We can find in literature studies that specifically address awareness mechanisms:

- As Dillenbourg notes, “an early example was given by Roschelle and Pea (85): when a student walks across the classroom to share PDA data by infrared instead of sending them wirelessly, this publicly visible walk provides the teacher and other students with the awareness of the actual data flow. The kernel (math problems) would be the same in a wireless version, but the orchestration is different” (7).
- The **Lantern** (70), by Alavi et al., is a multi-colored lamp used, in university contexts, as tool to enhance awareness: each team has a lamp on its table and the students can either turn it, to indicate the exercise they are starting to work on, or push it on the top, to ask for help from the TA.

This “helps teachers in planning their support and in optimizing their itineraries through the classroom”, i.e. it concentrates on “enhancing the awareness processes of the teacher”

(51).

Other than awareness, the other principles for orchestration that are supported are time management, control (or partial group management, since groups can continue working while waiting for help) and physicality.

- **Fireflies** (86), by Bakker et al., is “**an open-ended** peripheral interaction design developed for primary schools”, which consists of two main elements: light-objects and a teacher-tool. A light-object is a multi-colored lamp, placed on the desk of each student, that provides information to or about a single child (e.g. action or activity that he can do). In addition, all the light-objects together “form a distributed display that provides information about the class as a whole”. The teacher-tool provides small buttons, one for each student, to control the color of the corresponding light-object.

The authors state that the purpose of the system is to “support secondary activities of primary school teachers”, including the ability to “monitor if children are working well, keep track of which children had their turn, stimulate and correct children as well as keep an eye on the time-schedule”. It is clear that there are several orchestration traits supported here: time and workflow management, a certain degree of adaptation at runtime by using the teacher-tool and centrality of the role of the teachers who, due to the open-ended design of the system, “can specify for which goal and in which manner it is used, so that it will fit their personal way of working”.

The examples that we have just presented belong to the category of tools that show a specific subset of features and traits of orchestration, in this case both centered on the concept

of awareness of what is happening in the learning scenario. Fireflies goes somewhat further by introducing capabilities that give to the teacher flexibility and power to conduct the enactment at runtime in multiple potential and different ways.

Finally, using Prieto’s words, we could say that “even if these works share the keyword orchestration, and all pertain to the field of CSCL, [...] all of them can be seen as partial solutions to the complex challenges that orchestration poses for teachers” (51).

As Muñoz-Cristóbal et al. notice, “several proposals recently reported in the literature try to reduce teachers’ orchestration burden by means of authoring tools [...] These proposals are restricted to specific technologies, pedagogies, or to a very limited range of activities” (87). We will now introduce two systems, GroupScribbles and GLUEPS-AR, which fall into the category of technologies designed for general purpose classroom orchestration to support a wide range of pedagogies and specifically created to overcome the orchestration challenges that we have presented in the first part of this chapter.

2.2.5 GroupScribbles

GroupScribbles (GS, groupscribbles.sri.com) is a software originally developed by SRI International with the purpose of supporting the sharing of ideas by using notes (“scribbles”), either graphical or textual, written at individual or group level. “The whole idea of the application revolves around the metaphor of sticky notes and public/private whiteboards (88), and it is especially suited to be used with tablet PCs or other input devices that allow for easy and expressive drawing” (4).

If we look at the tool, the lower part of the interface represents the *user's private dashboard*, where he can draw or write scribbles. The top of the interface, instead, is occupied by a set of public boards that can be seen and modified by everybody (e.g. by moving notes in or out of the board, by reordering them or by directly drawing on a public board). In addition, a bar at the top allows to select some tools to draw, write, place forms, etc. To control the system, teachers have an additional set of dedicated options to enforce certain constraints on the use of the tool (e.g. the permission to write on boards, the ability to see or not scribbles on a specific board, etc).

The most peculiar feature of GS is that it acts at the same time as both an *authoring tool* and an *enactment tool*, since there's no difference between the actions of designing an activity and enacting it. This means that it allows for high flexibility and adaptability, with activities that can be redesigned *during* the enactment when needed. Moreover, as Prieto suggests in his analysis of the tool, "GS was designed with face-to-face scenarios in mind, and it is especially well suited to be used in classrooms with an electronic whiteboard and tablet PCs" (89): while the enactment of pedagogies supported by multiple displays will be key also for our system, RoomCast, we will see that in RoomCast the resources distributed to the displays will represent the core pedagogical content (kernel), as opposed to GroupScribbles where the interfaces on the screens represent just a support for the activities and need to be complemented by a high amount of face-to-face interaction.

If we analyze more in detail the system from the point of view of the support for orchestration principles, we can say that GS is **highly flexible**, for example by allowing the dynamic formation

of groups and creation of tasks (e.g. leveraging multiple boards), but also an easy modification of the flow of activities. *Contingency* is managed by adapting to runtime occurrences, e.g. by having additional boards. This flexibility implies a good support for *improvisation*. In addition, since the tool allows the different actors to be able to see what is happening in the scenario and what the other individuals or groups are creating on the public boards, the *awareness* of both teacher and student is enhanced.

On the other hand, the tool also presents some drawbacks: the overall support for orchestration is so thin, because of the **lightweight design** (e.g. when considering the level of velocity and difficulty to re-design activities), that “the demands on the teacher are high in all phases of the activity: the teacher has to design from scratch the activity flow, improvise, supervise, react, and evaluate based on ephemeral data” (4). As a consequence of this, the system in average requires a **high mental load** for the teachers, which could be difficult to manage, especially for novices.

2.2.6 GLUEPS-AR

GLUEPS-AR (9) is the latest version of a series of systems built “to support teachers in the orchestration of **across-spaces learning situations**”. In particular, the system allows to deploy learning designs, using multiple authoring tools, into multiple **DLEs** (Distributed Learning Environments). DLEs (90) is a term proposed to express the simultaneous usage of one or more different technologies among which *VLEs* (Virtual Learning Environments, i.e. online platforms to support the aspects of courses of study such as students, groups, materials, assignments, etc) and Web 2.0 tools (such as wikis, social networks, share web applications). In

addition to DLEs, GLUEPS-AR allows to integrate also *AR* (Augmented Reality) applications, such as those based on QR code readers placed in physical spaces.

This variety of heterogeneous learning environments are *integrated* by the system, together with a series of authoring tools for the teacher (e.g., *WebCollage* or *Pedagogical Pattern Collector*), and they can be leveraged to design a wide range of different pedagogies using different technologies and with the capability of managing the design and tune parameters at runtime. More specifically, this main interface for the teacher “allows to modify, complete and configure the initial design (e.g., including concrete participants/groups, adding tool configurations, etc.)” (87).

The central idea of the system is that “a learning design defined in any of the integrated authoring tools could be deployed in any of the enactment technologies integrated with GLUEPS-AR” (91). This allows for decoupling between pedagogical choices and the technologies used for their implementation. Moreover, the orchestration of the system is based on the coordination of a series of existing technologies (DLEs), that are properly set and configured to implement the desired pedagogy. As we will see, these two ideas will be also critical when we will present the design choices behind RoomCast.

GLUEPS-AR’s support for orchestration has been studied in several works (91; 89) and we could find that it supports, in specific ways related to its features and implementation, most of the traits identified, for example, by the “5+3 aspects” framework. These evaluations tend to underline that this technology is “specifically designed with the aim of overcoming four basic orchestration challenges”: “deployment of learning activities” using the variety of

available DLEs, “run-time flexibility” with adaptation to changes and technological support, “time-effective management of the DLE” and “usage by teachers in authentic practice” which results acceptable and sustainable for all the actors and taking into account all the constraints and complexities of the scenario. A complete analysis of the orchestration support features of GLUEPS-AR, based on the “5+3 aspects”, is presented in (92).

A drawback of the current implementation of the system is that it is *strongly teacher-centric*, thus lacking the possibility for the students to self-regulate, at least even partially, at runtime, while still having the teacher maintain pedagogical control.

Finally, we need to underline that the system has been thought to be deployed in the context of higher education, so we will need to consider this when comparing it to RoomCast, which instead has been designed mainly for primary school classrooms, where it was also evaluated. Furthermore, another noticeable difference is that while GLUEPS-AR’s main purpose is to support learning **across physical and web spaces**, e.g. switching from the classroom to a field trip, to the streets of a city or a museum, but also across web spaces such as Wikipedia and VLEs, RoomCast is focused on enhancing learning **inside the classroom ecosystem**, e.g. leveraging the notions of *participatory simulations* and *simulated investigations*, with the support of a **multi-display environment**.

2.2.7 Multi-display environments

As we will see, RoomCast’s capabilities are thought to be leveraged at their best in multi-display classroom environments, since the system’s core idea is to orchestrate the distribution of pedagogical, screen-based software resources to multiple displays at the same time.

In Section 1.2.3 we have seen that the topic of multi-display environments has been explored in HCI, where it has been applied to a wide variety of different contexts. Among these, research about the introduction of multi-display environments in the classroom has been also carried out, for example in CSCL when proposing concepts such as the “classroom ecology of devices” (93) or the “multi-display classroom systems” (38).

If we consider the literature in the field of technology-enhanced learning, we can find works specifically related to the orchestration of multi-display environments. In “Orchestrating a Multi-tabletop Classroom: From Activity Design to Enactment and Reflection” (94), Maldonado et al. explore the process of orchestrating a classroom using multiple interactive tabletops: students are divided into groups, each one working on a multi-touch tabletop; the teacher, instead, owns an *orchestration tool* that allows to start and stop activities, to move from one activity to the next one and to broadcast a message to the tabletop devices. The authors suggest that these multiple, shared displays, if well orchestrated e.g. using the orchestration tool, are useful to “promote students collaboration”, by having the students of each group actively participating at the discussion around their display and showing different levels of contribution and different group dynamics. The tabletop technology was used to distinguish which student performed each touch, thus measuring the effectiveness of the orchestration conducted by the teacher.

In “UniPad: Orchestrating Collaborative Activities Through Shared Tablets and An Integrated Wall Display” (23), Kreitmayer et al. present a “face-to-face, digital simulation for use in classroom settings that runs on shared tablets and a wall display”. The purpose of this combination of multiple displays is “to encourage students to talk, collaborate and make decisions

together in real time, by switching between working on shared ‘small group’ devices and a ‘whole classroom’ public display”. This setting is useful to “promote collaborative learning”: showing information about what the groups are doing and have done both to the teacher and to the students on the public display helps “promoting mutual awareness”, while moving the attention from group work on the tablets to the public display helps “supporting synchronous switching between small and large group activities”. More in general, they argue that “the combined use of shared tablets and public displays can be a catalyst for peer discussion and lightweight teacher orchestration in classrooms” (23).

2.2.8 RoomCast’s technology overview

To conclude this review of the existing implementations of technologies that support or are dedicated to classroom orchestration, we want to briefly anticipate the core design characteristics of RoomCast.

In Section 2.1.5 we provided a first overview of the vision of the support for orchestration that the system is meant to provide. In particular, we underlined that RoomCast is both an **authoring environment** and a **runtime system**. We also characterized it as an **orchestration technology** that takes care of distributing resources that represent **orchestrable technology**: these screen-based software resources, that we will call **channels**, in the current implementation are represented by either web content or native iOS applications. Thus, each channel can contain any kind of interactive content, which in RoomCast is meant to constitute a basic **pedagogical resource**. While these pedagogical resources are implemented (outside of RoomCast) based on the educational requirements expressed by the teachers, the role of the orchestration technology

is to distribute, as a **content provider**, the channels to the specific actors inside the classroom ecosystem that are allowed to use them.

In this sense, the role of the authoring environment is first to create the channels and then to configure how the distribution of channels has to take place, with the possibility of defining a flow of **activities**, each one correspondent to a different configuration. This, as we will see, defines a peculiarity of the system, that is the flexibility with respect to how the teachers can **tune the level of guidance** in the class: even if the teacher is always at the center of the orchestration process, she can decide, just by leveraging the core capability of the system of **selectively assigning pedagogical resources** to individuals or groups of students, to what extent the students have freedom in using the set of available resources. In this way, based on the context and necessities, the teacher can dynamically decide the **type of orchestration** in the continuum between the completely teacher-centric and the more learner-driven positions.

On the other hand, the runtime part of the system allows to adapt the distribution of pedagogical resources based on emergent occurrences: the system is designed to have high **flexibility**, since all the operations available during the design phase can also be leveraged to change the design (from the viewpoint of how the resources are distributed) at runtime. During enactment, the teacher also owns a very simple tool dedicated to switching between activities.

Finally, as already pointed out, RoomCast’s capabilities are thought to be leveraged at their best in multi-display classroom environments, where each display is allowed to access only the resources that the “role” associated to that screen has been authorized to use.

The next chapter will delve into the complete description of the system.

CHAPTER 3

ROOMCAST

This chapter is dedicated to presenting in depth the novel technology for classroom orchestration introduced by this thesis. We will start by stating the set of requirements, with different nature, that the system had to satisfy. We will then describe a simple scenario of real usage of the system to give a general idea of the main functionalities. The rest of the chapter will analyze in depth the features and the underlying architecture. Finally, a detailed analysis of how the system supports orchestration will be provided.

3.1 Requirements

The idea behind the origin of RoomCast comes from a very pragmatic necessity, that of improving the coordination of the activities in the context of instructional units based on collaborative inquiry using a set of distributed screen-based software resources. This is the area of interest in which our research group has been working for several years, expressing the growing need for a system like RoomCast. This demand didn't come only from the researchers, but also from the teachers themselves who have had to deal with the increasing complexity of the units proposed and of the technologies involved to support them.

For this reason, we had first to clearly identify the **pedagogical requirements** to address for the design of the system. These requirements came from two main sources: on one side all the literature on orchestration that have been presented in Chapter 2 together with the

principles and traits that had already been identified by previous works; on the other side, the experience of our research group coming from the enactment of several past instructional unit was critical to suggest a novel framing dedicated to our domain of interest.

Among the key requirements that were identified we have high **flexibility**, **management of resources** on multiple parallel classes and **reduced orchestration load** for the teacher.

At the same time, we had to identify **technological requirements** for the new system to be built. First of all, it had to be **reliable**, since this non-functional requirement is the main concern when we deploy technological tools in the classroom environment, where we don't want unexpected downtimes to interfere with the instructional unit and with the expectations of teachers and kids. Other fundamental requirements included **high usability** of the system for all the actors, achieved by effective user interfaces, **integration** of heterogeneous resources (e.g. web content and native mobile apps) to implement the desired support for orchestration and the **modularity and extensibility** of the architecture to accommodate future improvements and the support of new technologies.

From the requirements a first set of **specifications** was derived. We started then a process of quick successive iterations following an **agile** software development methodology, where requirements and specifications were continuously revised and new prototypes produced based on the feedback and findings coming from meetings with teachers and other researchers.

The rest of this chapter describes the current status of implementation of the system, that is the one that was also used for the first pilot study of RoomCast.

3.2 Description

Let us first give a general overview of the system. To show the variety of different tools and capabilities that RoomCast offers to both teachers and students, we will start by describing a scenario of concrete use of the system, based on a very simple example of whole class collectivist pedagogy. In particular, we will describe the steps that a teacher can take to orchestrate the classroom with RoomCast by showing relevant parts and subsets of the system, useful to convey a general understanding of the technology. A more thorough analysis of each single component of the architecture will be carried out in the next sections.

Let us consider the case of a collectivist pedagogy in which a teacher wants to have the kids engage in a collaborative inquiry to study an ecosystem populated by predators and preys. Moreover, we are considering the case in which we have the availability of a set of resources, possibly designed and implemented ad-hoc by learning technologies researchers or obtained by other means, under the form of graphical interfaces to be shown on a set of displays in the classroom.

In this situation, the teacher, before starting the class activity, can leverage the help of RoomCast to setup how, when and with which level of granularity that content is provided to the class.

Let us consider the already mentioned set of available resources, which we will call *channels*:

- Habitat: an animated simulation of the ecosystem
- Statistics: charts showing data about the species in the ecosystem
- Predator actions: interface to execute actions available to predators

- Prey actions: interface to execute actions available to preys

To have this “lineup” of four different channels available inside the system, the teacher (or whoever is in charge of providing the content) will have to represent it inside RoomCast. A *channel* represents a unit of content distributed throughout the system. In Figure 3 we show how a channel looks like. We will describe in detail in the next chapters how to create a catalog of available channels.

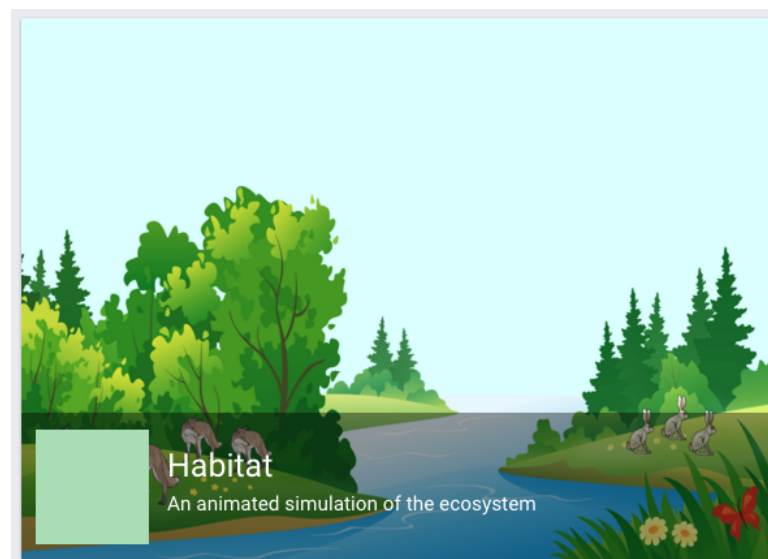


Figure 3: The representation of a unit of content, the channel, inside RoomCast.

The next step for the teacher is to decide how to distribute the channels. RoomCast embeds many different capabilities to give to the teacher a wide range of possible different choices to craft her design for classroom orchestration.

For this first, basic example let us suppose that our teacher wants to start the first activity with the class by having two public displays in the classroom which will be used by two respective groups of kids. At the beginning, each group will be able to access, on its screen, just the Habitat channel: this is clearly represented in RoomCast (Figure 4), where both Display 1 and Display 2 are associated to the Habitat.

As we will see more in detail in the next sections, we will call each identity or role that can be assigned to devices inside the classroom with the term *package*. Thus, for example, we will say that “Display 1” is the name of the first *package* in our list.

The teacher, at configuration time, i.e. still before the enactment of the pedagogy in class, can already foresee that at some point later on during the development of the multi-week instructional units, the kids will need to access more channels than the ones assigned at the beginning. For example, after a few initial units dedicated to the exploration of the habitat, the teacher might want the students to get access to the Statistics channel to let them study analytically the evolution of the species. Since the teacher at this point still doesn’t know when the new channel will be needed, she can create a new RoomCast *activity* called “Activity 2”.

We can notice that in this second activity (Figure 5), both the two displays have access to one more channel, i.e. Statistics.

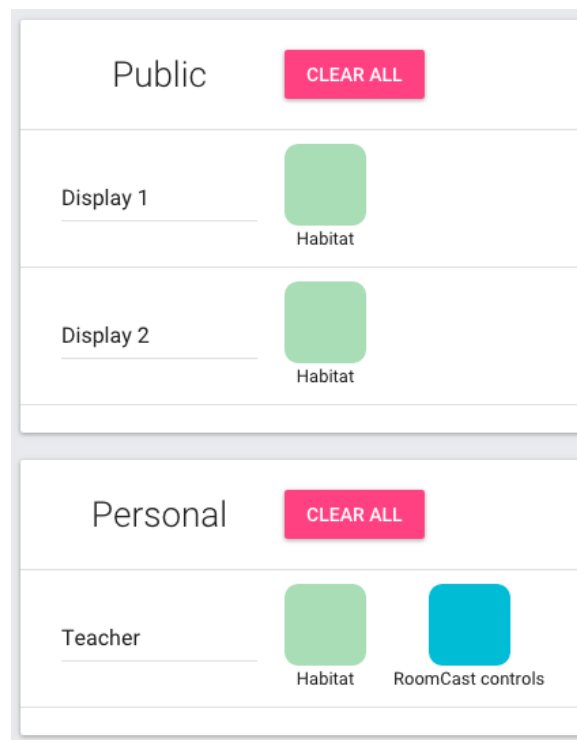


Figure 4: The representation of a mapping of available channels over packages inside RoomCast.

The teacher will now set up a third activity (Figure 6) in which she allows for two more channels: one display, and then the associated group of kids, will have the “Predator actions” channel, so they will assume the role of predators to influence the ecosystem. On the other hand, the second group will have “Prey controls” to act as preys. As we can notice, the teacher, by setting mappings of channels inside RoomCast, can enforce what each display shows, that is which visual resources each group of students in front of it can use at any point in time during the instructional unit.

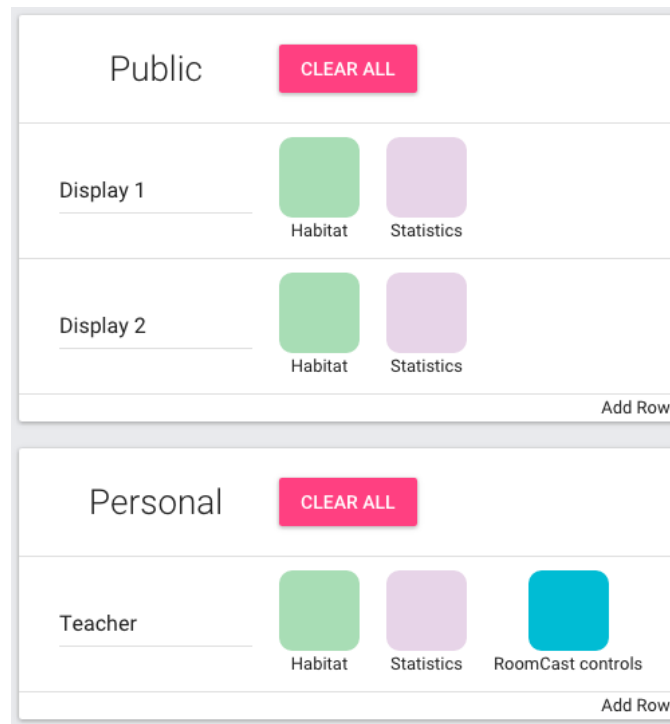


Figure 5: Mapping of channels for Activity 2.

As the reader might notice at this point, the teacher should also need a way to transition from one activity to the next one. This is achieved in this example through the package named “Teacher”, which gives her access to the “RoomCast controls” channel: as we will see, this is a dedicated interface for the teacher for real-time orchestration, including the launch of activities whose content had been previously created.

Finally, let us take a first peek at the RoomCast client application which runs on the two public displays: this is the main RoomCast application that shows the available channels and allows to “play” them. Figure 7 shows the sequence of states of this interface going from Activity

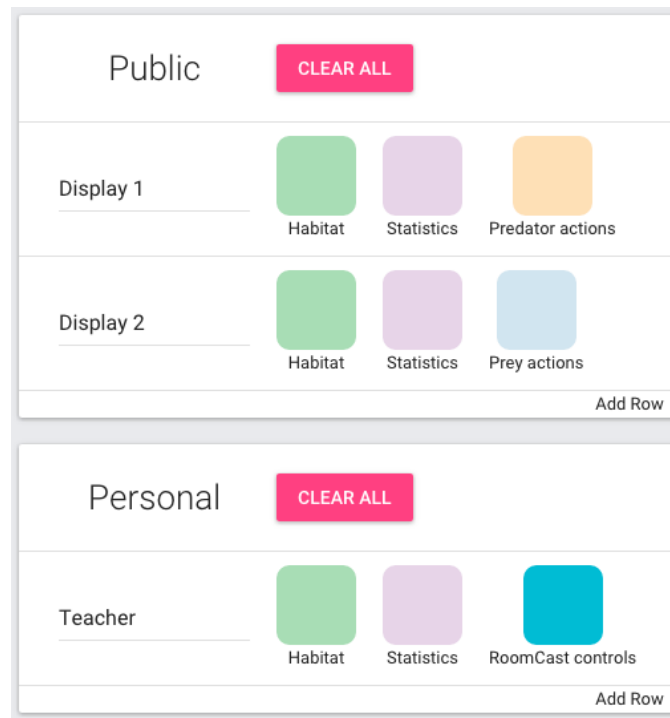


Figure 6: Mapping of channels for Activity 3.

1 to Activity 3 and it reflects the decisions taken by the teacher both at configuration time and real-time when transitioning between stages.

3.3 The cable television metaphor

While describing the high-level functionalities of the system, the reader could have noticed that terms such as “channel”, “package” and the name “RoomCast” itself are all related to a common domain. We indeed chose to denote specific RoomCast’s features and components by using a cable television metaphor: RoomCast is an orchestration technology based on the idea of delivering specific content to the desired recipients at the desired point in time.

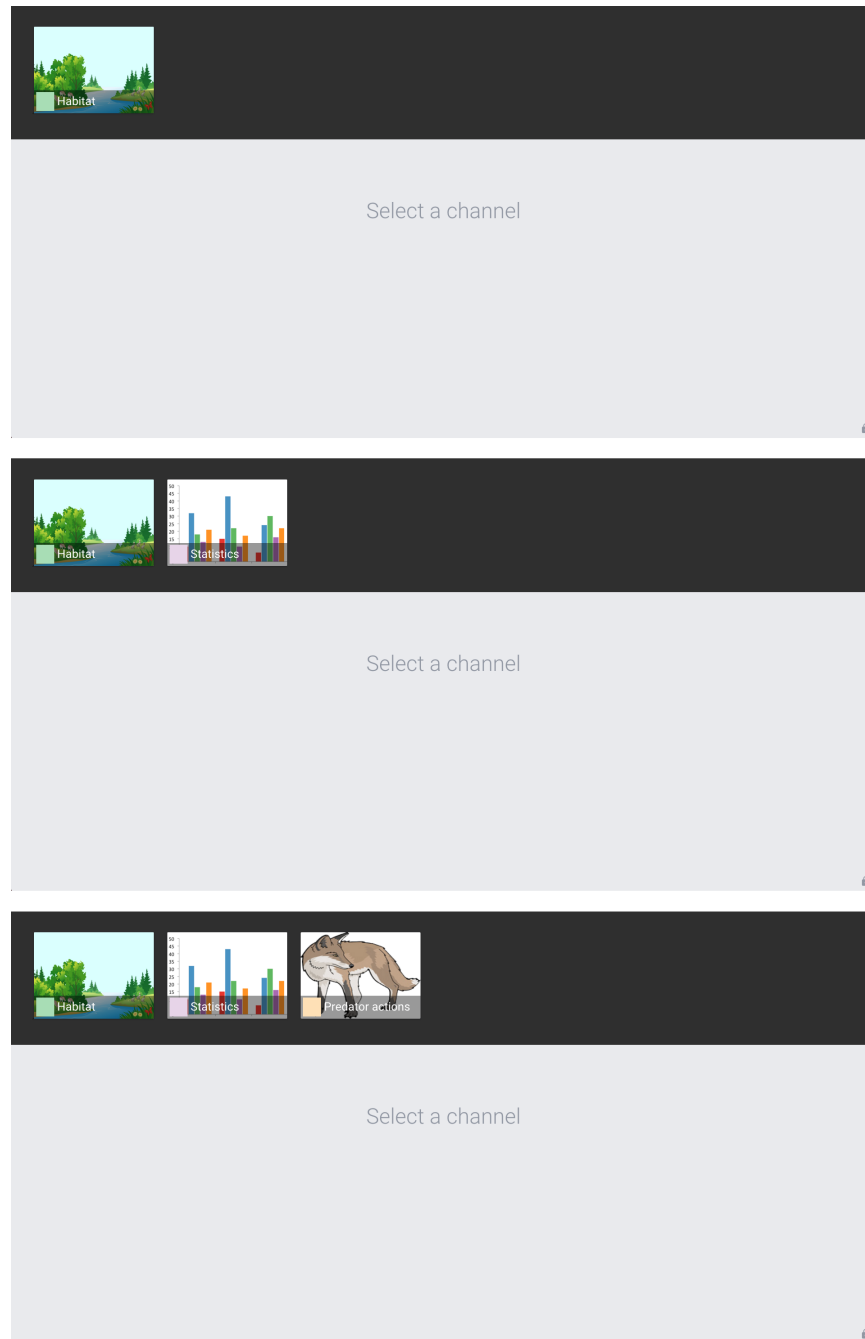


Figure 7: RoomCast client app: set of available channels for Display 1 going from Activity 1 (top) to Activity 3 (bottom).

More precisely, RoomCast positions the teacher as the *provider* of individualized *channel packages* to students, with each *channel* representing a specific instructional resource. Prior to instruction, teachers design the set of different channel packages needed within each *activity* (or phase). During instruction, students *log in* on individual and public devices using a package name as credentials, they are presented with a *channel lineup* based on their role in the current *activity*, and can *change channels* depending on their needs. During enactment, teachers use an interactive tool to transition among activities, and to reallocate channel packages during an activity (if, for example, a student must play a different role than expected because of the absence of another student).

3.4 Hardware architecture

RoomCast is mainly constituted by a series of web interfaces: this means that it works on a wide variety of devices, that is on any device that runs a web browser, such as desktop computers, tablets and potentially even smartphones.

As we will see while describing the software stack, RoomCast relies on the underlying nutella framework for network communication. nutella is powered by a message-oriented middleware, where all the messages exchanged by software components go through a message broker. This means that the hardware requirement for RoomCast, and for the whole nutella framework of which RoomCast represents a component, is to have a centralized server, on which nutella is installed, that acts as a broker.

One of the main components of RoomCast is the client application for iOS devices, which lets all the actors in the classroom access to their assigned contents. To run this app we need an iOS device (iPhone or iPad) powered by iOS version 8 or higher.

3.5 Software architecture

This section will describe in detail the software architecture of the system, that is the stack of technologies employed, the design of every component and the relationships between different components. For each module of the system, we will provide a technical analysis as well as, in the case of the interfaces, a description of how they can be used by the different actors, which actions are allowed and what is their educational purpose.

Let us first take a look at the graphical overview of the whole system (Figure 8). We will now proceed at reviewing each component of this architecture and we will keep providing references to this graphical representation to clarify how it works, what are the main design choices and in general what is the rationale behind the system.

3.5.1 nutella framework

While, as we will see, RoomCast could be considered a framework and system designed for the general classroom orchestration of activities and groups of actors in multi-display environments, the current implementation of the system relies on the use of a framework called “nutella” (95).

nutella (nutella-framework.github.io) is “an open source software framework and suite of services, tools, and design templates that support the construction, configuration, and delivery of simulated scientific phenomena to elementary school classrooms” (95). In particular, nutella is designed to support the construction and enactment of macroworlds. *Macroworlds* is a term used

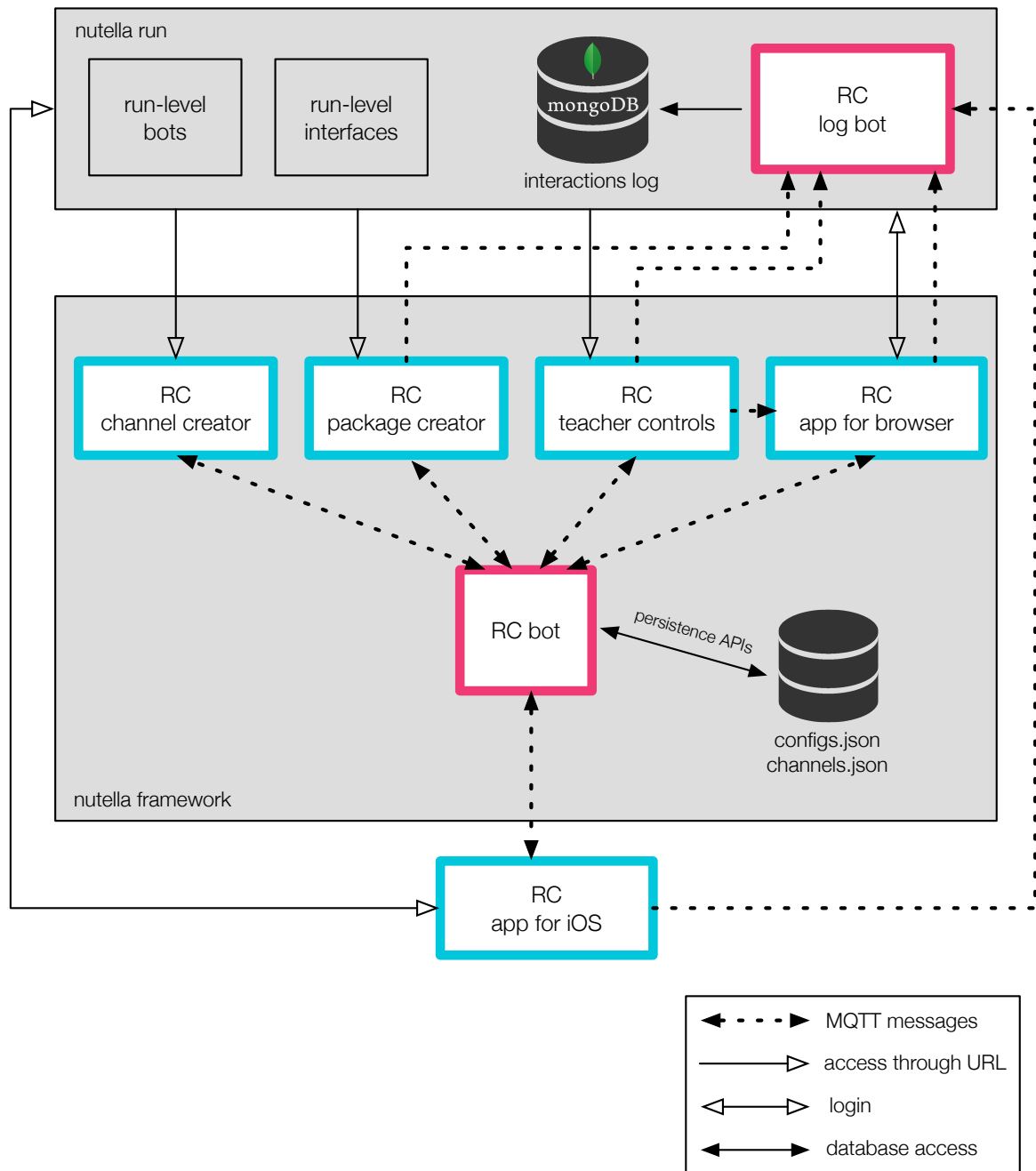


Figure 8: Overview of the software architecture of RoomCast.

to describe learning applications that support two main genres of class pedagogies: “*participatory simulations*” (30) and “*simulated investigations*”. As already described in the introduction to this thesis, in participatory simulations, students adopt a first-person perspective, interacting as agents to drive the simulation; the record of their behavior and the emergent results from that behavior become the objects of community inquiry (30). In simulated investigations, learners assume a third-person perspective as investigators of an algorithmically simulated phenomena rather than as actors in the simulation itself. For example, in RoomQuake (28), learners experience a series of simulated earthquakes reflected on simulated seismographs positioned at known locations around the classroom and work in teams to analyze and study the phenomenon.

nutella allows to build applications, represented by collections of software modules (called *components*) that can only be of two types:

- **interfaces:** user interfaces, i.e. the presentation layer that directly interacts with the users. Interfaces can also receive inputs from other interfaces or bots;
- **bots:** represent what we would usually call “back end” in computing, i.e. the software layer that is not directly accessible by the user. Thus, bots can only receive inputs from other interfaces or bots.

At its core, nutella is powered by a message-oriented middleware that allows nutella components to exchange messages between each other. In particular, all the messages exchanged by software components go through a message broker, which, in the case of nutella, is an open source software called mosca (mosca.io). All the exchanged messages must comply to the MQ Telemetry Transport (MQTT) protocol.

nutella nicely wraps this low-level communication layer with higher-level, user-friendly APIs, making it very easy for developers to have components communicate between each other with JSON (json.org) messages. The framework provides two separate communication strategies or messaging patterns:

- *request/response* (or pull): born to allow communication in client-server applications, it is useful when we want to directly fetch data from one or more components over a known communication channel
- *publish/subscribe* (or push): allows some components to “listen” on specific channels so that they can be instantly notified when the needed data become available from other components which will “broadcast” the data as a message through the respective channel.

Since in nutella applications data are rarely centralized in a single component, but they are instead distributed among different components, creating a situation where each component is, at the same time, a client and server for different data, it is easy to understand that the publish/subscribe mechanism is the main communication strategy for this kind of applications.

The layer on top of the nutella messaging protocol is represented by a set of libraries implementing the protocol itself and simplifying the interaction with higher-level components of the framework. nutella libraries have been implemented to support a set of common languages. For a detailed description of the protocol and available APIs please refer to the official nutella Github documentation page (github.com/nutella-framework/docs/blob/master/protocol/index.md).

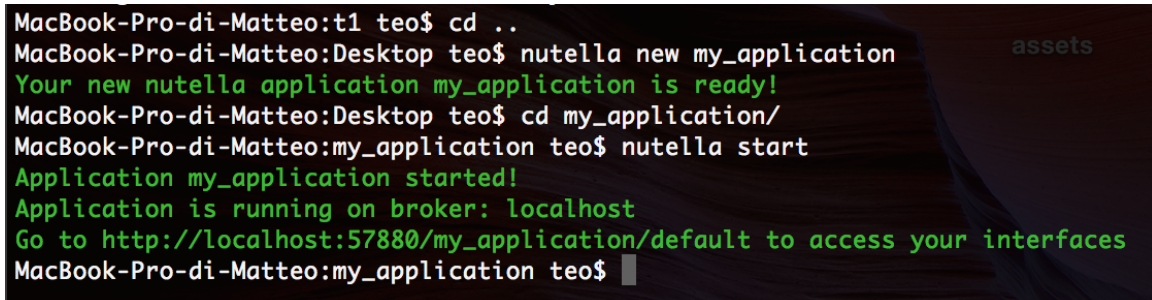
We can now describe the top-layer in the nutella software stack: the *components layer*. nutella components come in two different flavors: **framework components** and **user-defined components**.

Framework components are integrated within the framework itself and provide a set of high-level features (e.g. access to location, orchestration, logging) to all nutella applications. It should be now more clear to the reader why the software overview schema shown in Figure 8 depicted most of the RoomCast modules as being inside (or on top) of the nutella framework. RoomCast itself, in fact, represents one of the core nutella framework components. The advantage and importance of framework-level components are due to the fact that they are embedded as the top-layer of nutella and installed together with the framework, so whenever a nutella application is started, this will be able to leverage the capabilities offered by the suite of framework components. Since we want RoomCast, as an orchestration tool, to be always available along with any nutella application run in the classroom, the natural choice was to have it play this role.

Other than framework-level components, nutella allows to create user-defined components, that is the way designers and developers implement nutella macroworld applications and thus represent the main purpose for which the whole framework was built. For developers, once the framework has been installed on the machine using RubyGems (`gem install nutella_framework`), creating a nutella application is as simple as executing a single command from the Command Line Interface: `$ nutella new my_application`. This will generate a skeleton of a macroworld

application, where the developer's task will simply be that of implementing the desired bots and interfaces in the respective dedicated subfolders.

Then the developer can just navigate to the application's main directory and type the command: `$ nutella start` (Figure 9).

A terminal window screenshot with a dark background and light green text. The text shows a series of commands and their outputs in a shell environment. The commands are: `cd ..`, `nutella new my_application`, `cd my_application/`, and `nutella start`. The outputs include confirmation messages and a URL to access the application interfaces.

```
MacBook-Pro-di-Matteo:t1 teo$ cd ..
MacBook-Pro-di-Matteo:Desktop teo$ nutella new my_application
Your new nutella application my_application is ready!
MacBook-Pro-di-Matteo:Desktop teo$ cd my_application/
MacBook-Pro-di-Matteo:my_application teo$ nutella start
Application my_application started!
Application is running on broker: localhost
Go to http://localhost:57880/my_application/default to access your interfaces
MacBook-Pro-di-Matteo:my_application teo$
```

Figure 9: Commands to create and start a new nutella application.

At this point, the application will be immediately accessible via any web browser and its landing page (Figure 10) will show, separately, both user-defined and framework level interfaces to be accessed.

Let us describe one last, but very important feature of the framework: `nutella` allows to create, besides multiple different running applications, also multiple different *runs* inside a single application. A `nutella run` represents an instance of an application: similar to how objects are instantiations of classes in object oriented programming languages, a run is an instantiation of an application which owns its own data. This allows to create multiple runs, e.g. one for each

Welcome to *my_application*

Use the buttons below to launch a specific interface

Your interfaces will automatically be loaded here once you add them to your application

Framework level interfaces

Launch!	Room Monitor	Monitor macroworld applications
Launch!	Room Debugger	A debugger to help you build your nutella applications
Launch!	Room Places - Beacon cloud	Mange the list of iBeacons available to all applications
Launch!	Room Places - Classroom layout	Manage classroom space and resources
Launch!	RoomCast - Channel Creator	The interface to modify the channels catalogue of RoomCast.
Launch!	RoomCast	RoomCast's main app to log in and play channels.
Launch!	RoomCast - Package Creator	The interface to create activities as mappings of channels over packages.
Launch!	RoomCast - Teacher controls	The interface to control the class activities in real-time.

Figure 10: Landing page for a nutella application.

different classroom, that have the same bots, interfaces, and behaviors, but work on separate data. To start a new run with a custom name we just need to issue the command `$ nutella start run_2`.

Let us now take a look again at Figure 8: after having extensively described the nutella framework, we can notice that most of the modules of which RoomCast is made of have been

placed inside the framework itself as framework-level components to enhance nutella applications with support for orchestration.

More specifically, we can notice that RoomCast has four interfaces and one bot directly embedded inside the framework. The iOS client of RoomCast is not considered part of the framework since it comes as a standalone iOS application. Finally, we can notice that there are also some additional RoomCast components externalized and placed inside each nutella run: we will talk about this in the next sections.

3.5.2 RoomCast interfaces

We will now describe in-depth the different interfaces that allow the users to interact and take advantage of the features of RoomCast. This will hopefully provide a clear understanding of the mechanisms provided by the system. Only at that point it will be meaningful to analyze the bots, protocols and data structures that power the whole system.

For each interface we will first provide a textual description of its features and design, then we will summarize its technological and educational affordances, and finally we will show the most interesting and innovative technical details about its implementation.

Reactive user interfaces

There are a few, core design decisions that are common to all of the RoomCast user interfaces. To create these interfaces, in fact, we relied on **React** (facebook.github.io/react), a JavaScript library for building user interfaces released as open-source software by Facebook.

React is a technology particularly suited for the kind of user interfaces and user experience that RoomCast wants to offer to its users. Let us describe here the main characteristics of RoomCast's user interfaces, that have been mostly implemented leveraging React:

- **Modularity and composability:** React approaches building user interfaces by breaking them into components: this favors reusability and separation of concerns. It also satisfies the requirement of having many different RoomCast interfaces sharing a few, core common components. A significant example for this is the *card*, used as a metaphor for a *channel*, that has to be embedded in most of our UIs. The composability of these graphical components was also critical to manage the complexity of the DOM hierarchies of some interfaces.
- **Stateful components:** composability in React is made even more useful by the presence of a *state* that each React component holds. The state represents the essential data that a component needs to be correctly rendered. The strategy here is that if two components need the same state, then we put that state in a component higher in the hierarchy and then we pass down the data.

```
getInitialState: function () {
  return {
    channels: {},
    selectedChannel: null,
    backgroundMessage: null,
  }
},
```

In the example shown above, we are setting the initial state of the top component of the RoomCast interface that manages the lineup of channels. As we can notice, the state is represented by the set of channels, the id of the currently selected channel and a string to be shown as a feedback message for the user. While the `backgroundMessage` will be directly rendered at this level of the React hierarchy, `channels` and `selectedChannel` represent data that is needed by a variety of sub-components: they will have to be passed down the hierarchy.

- **Reactive updates:** in RoomCast, data (including configurations, channels, activities and so on) keeps changing over time. “React makes it easy to display data and automatically keeps the interface up-to-date when the data changes”, by propagating the updates down the hierarchy of components. For this reason, a careful and wise use of React’s features, integrated with nutella’s messaging paradigm, was critical to effectively support real-time configuration and orchestration.

Let us show an example of integration of some nutella primitives into the lifecycle of React:

```
componentWillMount: function() {
  var self = this;
  // Get current channels catalog
  nutella.net.request('channels/retrieve', 'all', function (response) {
    self.setChannels(response);
    nutella.net.subscribe('channels/updated', function (message, from_) {
      self.setChannels(message);
    });
  });
},
```

In this example, we are injecting nutella API calls inside the `componentWillMount` phase of the top-level React component of one of our interfaces. This function is invoked only once, immediately before the initial rendering of the component in the browser occurs. This is, thus, the right moment to fetch, with a nutella request, the current list of channels. We have also to subscribe to future updates of the channels catalog: every time the RoomCast bot will receive and store a new set of channels, a notification message called `'channels/updated'` will be broadcasted; the subscribed interface will then react to this notification and execute the given callback, which in this case, by calling `setChannels(message)` on the React component, will update the state of the component itself. The change of state will be immediately reflected in a re-rendering of the UI to show the updated data.

- **Consistency and unified design:** RoomCast has been designed as a series of several separate user interfaces for reasons that will be clarified when talking about the *design for orchestration*. This choice required us to be particularly careful when designing each interface, to make sure that the users would have been able to take advantage of this design, by maintaining a consistent experience and understanding of the set of tools that they are provided with.

This means that we had to look for high *internal consistency*, that is the degree to which the whole system is consistent with itself, to provide a clear understanding of the mechanisms for orchestration. An example for this is again the use of the card as a metaphor for the channel, i.e. the fundamental unit of content inside the system, that has to be perceived and understood consistently throughout all the interfaces.

Another significant choice to support internal consistency is represented by the use of the Material UI library of React components (material-ui.com), that, once properly customized and in some cases re-implemented, contributed to give graphical consistency in terms of unification of the overall design of the system.

As with every user interface, attention was also given to *external consistency*, that is providing UIs that implement, both in terms of graphical aspect and behavior, patterns and design choices that are commonly used and to which users are familiar with.

Affordances analysis

Before starting our review of the interfaces, we also need to clarify the notion of *affordance*, which will be critical while describing the system.

The term “affordances” was first introduced by the psychologist James J. Gibson in 1977 as “the potential behaviors available to an animal in a given situation or environment”(96). But it’s Donal Norman that in 1988 applied the term to the field of human-computer interaction (HCI). Norman categorizes the affordances of a tool into two main kinds: *real affordances* are all the “action possibilities that are readily perceivable by an actor”, while *perceived affordances* are only those “actions that the user perceives as possible”(97). In other words, while real affordances include all the actions that are allowed by the tool, perceived affordances are represented by the subset of actions that the user is able to perceive as possible. Thus, the former are related to a notion of *utility* of the tool, while the latter are associated more specifically to the *usability* of the tool. More in general, Norman defines an affordance as a *relationship*: “when actual and

perceived properties of an object are combined, an affordance emerges as a relationship that holds between the object and the individual that is acting on the object” (97).

If we look specifically at the learning technologies research field, we can find some direct applications of these concepts to the domain. In particular, Kirschner et al. (98), analyzing the context of CSCL, distinguish affordances into three kinds:

- **Technological affordances**, equivalent to the definition by Norman cited above and related to usability.
- **Social affordances**, “properties of the CSCL environment that act as social-contextual facilitators relevant for the learner’s social interaction” (99).
- **Educational affordances**, “those characteristics of an artifact that determine if and how a particular learning behavior could possibly be enacted within a given context” (100), that is those properties of a tool or an environment which, through the interaction with the user, are able to enhance the learning potential.

Taking inspiration from these studies and definitions around the notion of affordance, for each interface provided by RoomCast we will perform an analysis of the affordances to describe analytically and in depth the capabilities of the tool and, even more importantly, to lay out and frame a research context that will serve as a guide while performing the pilot study (chapter 4).

In particular, we will concentrate on two of the above-mentioned kinds of affordances: while the technological ones will list the perceived action possibilities of the tool, educational affor-

dances will be useful to describe at a higher level the educational contribution of the tool by means of its capabilities as an orchestration technology.

3.5.2.1 RoomCast channel creator

In the workflow of a teacher who is setting up RoomCast, the channel creator interface should be the first to be used: it allows to create, delete and modify the set of channels that will be available into the system. In some cases where teachers have low ICT knowledge, this step could be executed by the researchers or developers that implemented the channels.

Figure 11 shows how the channel creator interface looks like for the simple example of the predator-prey investigation described in section 3.2. This catalog is represented as a grid of channels, where each channel is represented through the **metaphor of the card**.

Once we click on a channel card, we go into selection mode for that channel (Figure 12). Here we can define name and description, upload a background image for the card and choose a custom color (from a dedicated palette) as icon.

The metaphor of the card is useful not only because it nicely supports the idea of a distributable unit of content, but also because of its characteristic of being double-sided: we leveraged this by using the *front* of the card to let the user model the graphical aspect of the channel, while the *back* of the card contains what's hidden behind and attaches the actual content to the channel.

If, indeed, we click on the “settings” button at the top-right corner of the card, the card will flip and show us its other side (Figure 13). Here we can set the connection of the card to its actual content, which for now can be of two types: either web content (in this case

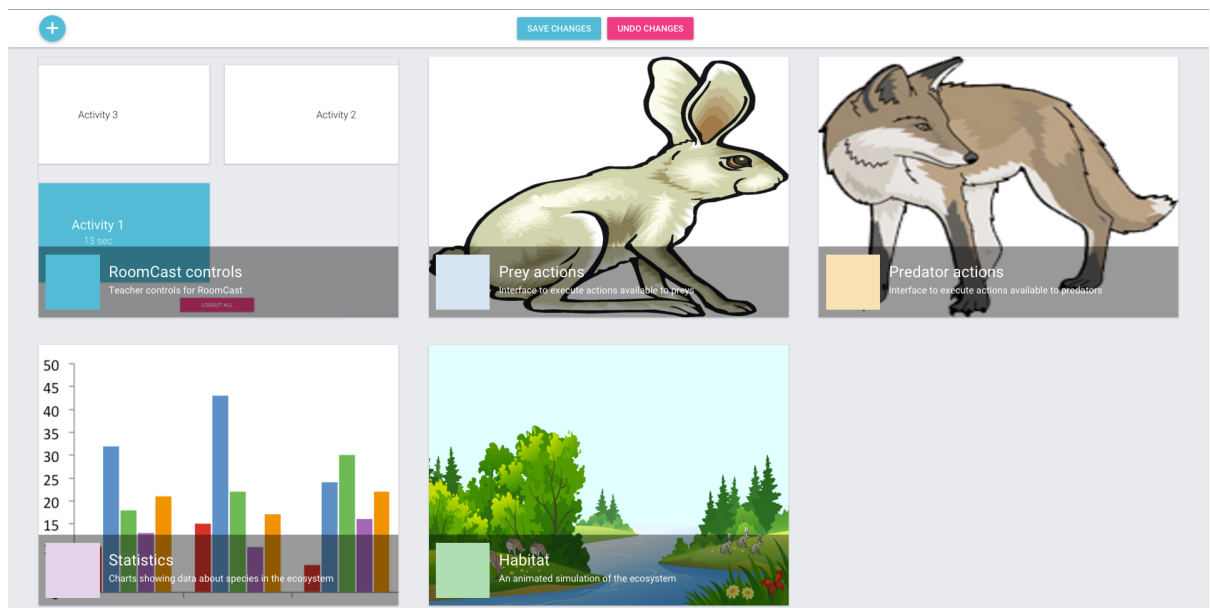


Figure 11: Overview of the channel creator interface.

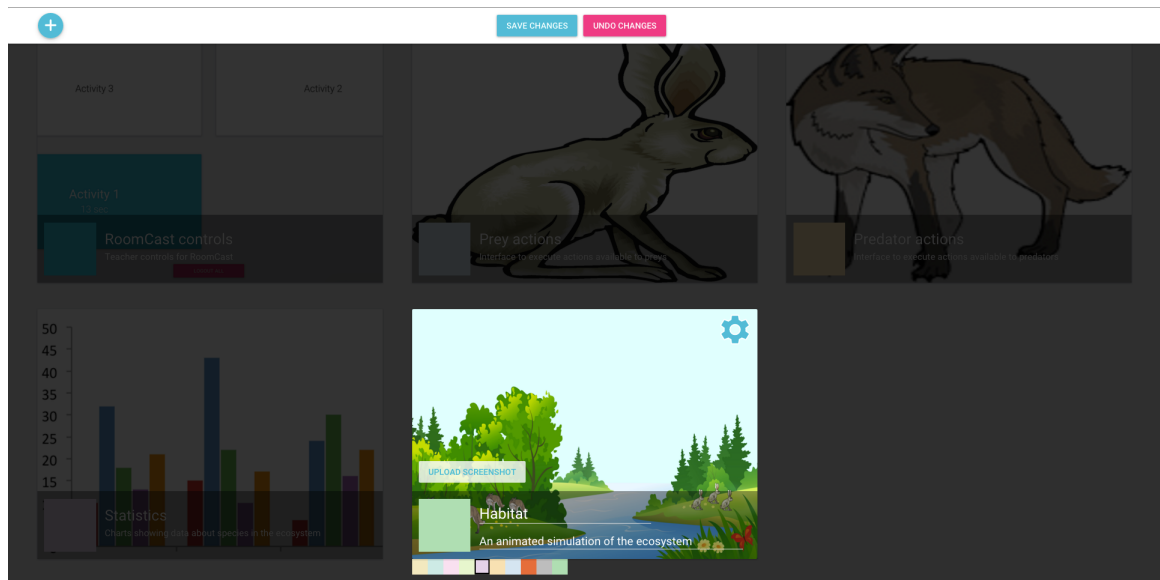


Figure 12: Selection of a channel card to be modified.

we will provide a URL) or a native iOS application (by inserting the name of the custom url schema that references that iOS app on an iOS device). Critical to enhance the capabilities of the orchestration system is also the possibility of passing additional parameters attached to the URL of the card: this allows several interesting uses obtained by creating **parametrized channel cards**. In addition, at runtime the system by default automatically passes as parameter to the channel interface the name of the package (e.g. “Predator”) that played that channel: this gives to the developer of the channel the ability to **personalize the channel based on the specific actor** that is using it.

We also need to emphasize here the notion of content associated to the channel: since this could be represented by any web or native mobile interface, we can have any kind of interactivity and capabilities involved to support, implement and improve class pedagogies. This is why we refer to digital (screen-based) software **resources** to characterize these channels distributed throughout the system.

At any time, we can use the top bar of the interface to save or undo the current changes. Once saved, the changes will be propagated in real-time to the whole system.

Affordances

- Create new channel (blue “plus” button)
- Delete existing channel (signalled by red cross signifier)
- Modify existing channel’s appearance (name and description text fields, icon color palette, image upload button)
- Flip card (signalled by blue “settings” button signifier)

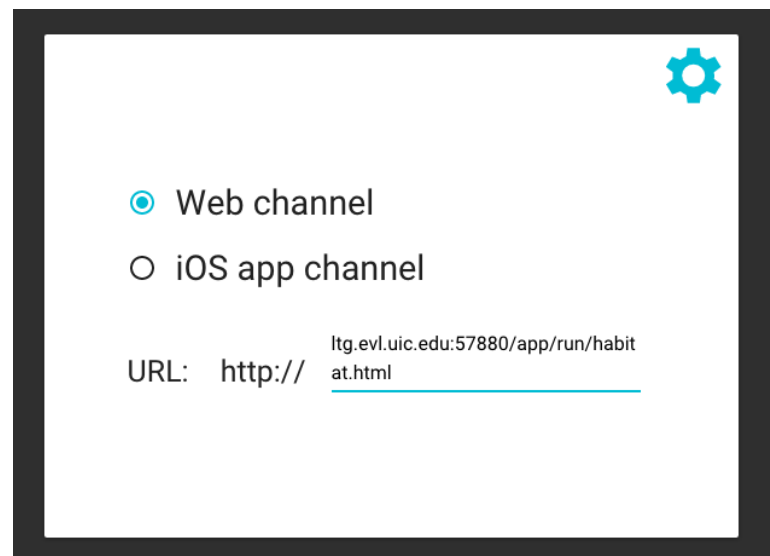


Figure 13: Back side of the card.

- Set URL of existing channel to link content to the card (radio buttons and text field)
- Save and undo changes (buttons in top bar)

Educational affordances

- Manage the individual pedagogical resources distributed to the class

Implementation details

Let us look at some of the most interesting technical challenges tackled to build this interface:

- **Animation of the cards:** flipping of a card from front to back was implemented with CSS3 transforms applied from JavaScript with an algorithm to keep track of the correct duration and direction of the animation, together with a smart use of HTML elements

properly layered. The importance of the presence of this animation is that it contributes to form and reinforce the **conceptual model** (97) that we want to express by consistently using the metaphor of the card.

- **Uploading of images:** when the user saves the changes, the interface processes the queue of all the newly added images to upload them to the server. This required to manage asynchronously the uploading process, but also to use a smart way to efficiently store these images: we implemented a framework-level nutella bot that takes care of hashing the images and uploading them only if the hash is not yet present on the server. This means that each image will be stored only once, even if it will be re-used and re-uploaded in the future.

3.5.2.2 RoomCast package creator

This interface is the main authoring tool for teachers: it allows to selectively assign sets of channels to specific roles or identities inside the classroom and to manage activities.

If we look at the interface (Figure 14), we can notice that it is divided into two main sections: on the right side we have the complete lineup of all the available channels (i.e. the one we created with the channel creator interface), while on the left side there's a sort of table that the user has to fill out.

Let us define here a few specific terms that are critical to understanding the system:

- **Package:** consistently with the content provider metaphor, we decided to call “package” a specific set of channels that is distributed to a device. Thus, when we first start a

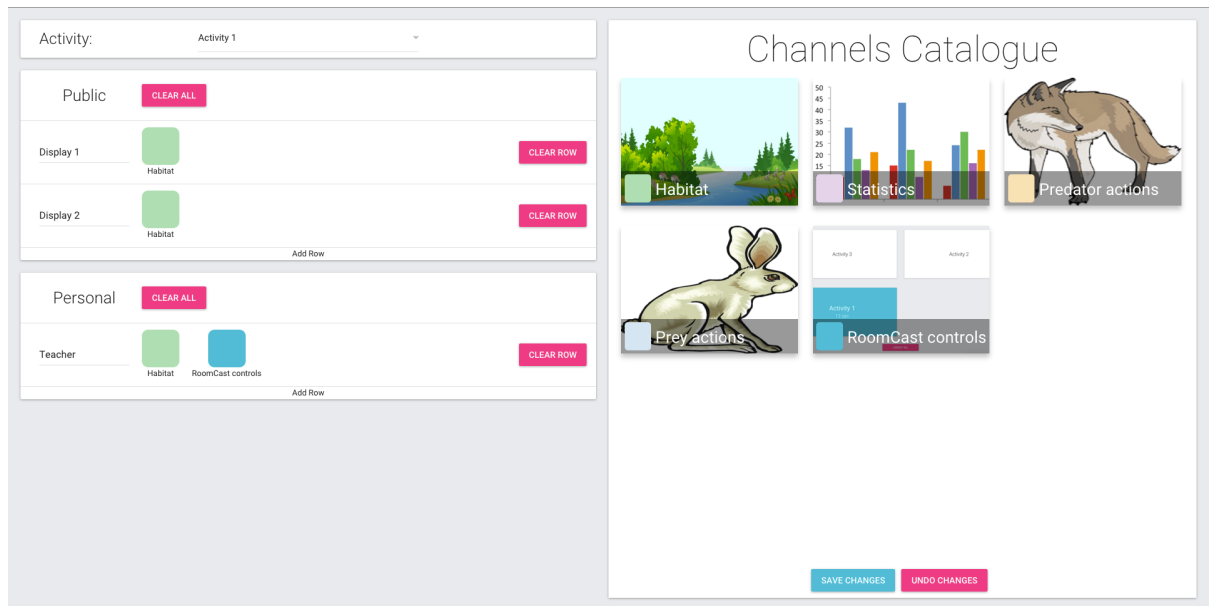


Figure 14: Overview of the package creator interface.

RoomCast session, as we will see, we will *log in* with a package name; in other words, the package name is the temporary *identity* or *role* that identifies each device (and as a consequence the people in front of it), determining the restricted lineup of channels that can be used.

- **Mapping:** we refer to “mapping” to describe the association of some channels to a single package. In Figure 14 each row of the table represents a package, with name and content. The result of placing a set of channels inside the package is a mapping.
- **Activity:** a RoomCast “activity”, also called **configuration**, is the final setup of mappings for all the packages that we created through the interface. This configuration, which

tells RoomCast *how to distribute* the resources, is given by the state of the table of the mappings.

In RoomCast, we can create as many different configurations as we desire and each one of them will represent a single activity that will be possibly started by the teacher.

The package creator (Figure 14) allows to execute a variety of actions involving the entities cited above. When the user clicks on a channel from the catalog (Figure 15), the interface enters a contextual selection mode: rounded, colored buttons will appear on each row of the table of the mappings. These buttons are “smart” because they change appearance based on the selected channel and the state of the mapping: if the channel is already inside a row, that is it has already been put inside that package, the button will be pink and with a “minus” icon, while if the channel is not yet inside the package, the button will be blue and with a “plus” icon.

In the same way, a channel can be selected directly from the table of the mappings. We can also notice that the table itself has been divided into two main sections, “Public” and “Personal”, as a way to help the teachers to logically distinguish packages belonging to public displays from those meant to be used on personal devices such as iPads.

In the Activity bar at the top, we can find another component critical for the whole system: the management of activities. This takes place in a single UI component represented by the custom dropdown menu shown in Figure 16. Each different activity corresponds to a different configuration or table of mappings and it’s possible to switch from one to the other to personalized the system as needed.

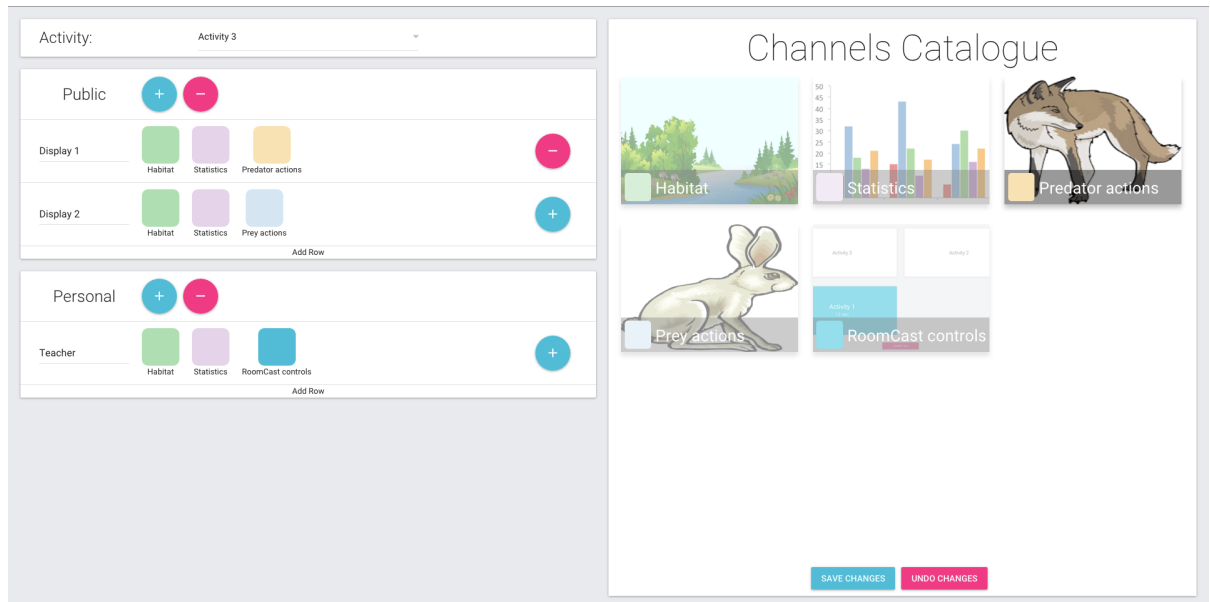


Figure 15: Package creator: contextual mode with channel selected.

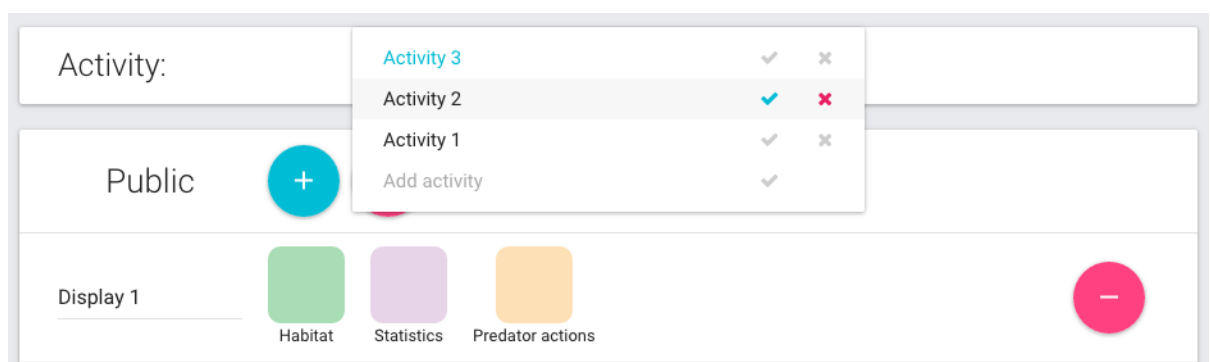


Figure 16: Package creator: detail of the dropdown menu to manage activities.

Affordances

- Create (“Add Row” signifier), rename (text field) and delete (pink bar on hovering as signifier) packages
- Select channels from catalog or table of mappings (signalled when hovering)
- Add/remove channels from one or more mappings (contextual and global buttons)
- Add, rename, delete and select activities (from the custom dropdown menu)
- Save and undo changes (buttons)

Educational affordances

- Configure what are the packages (or **roles**) needed
- Configure which resources are distributed to the class by means of those packages
- Create different activities to stage different configurations over **time**
- Execute the above actions either before the enactment of the instructional unit (i.e. **offline**) or at runtime during enactment (**online**).

Implementation details

- **Comprehensive activities dropdown:** the custom dropdown (Figure 16) was designed to contain all the affordances needed to manage activities in one single place. To do this, we started from the basic Material UI React dropdown component and customized it by adding a sub-hierarchy of stateful React components to provide support for creation, modification, deletion, and selection of activities.

- **Complex UI decomposition:** since the package creator is the most complex RoomCast interface in terms of number of UI elements and interaction, the UI has been broken into several React components to strive for separation of concerns, reusability and to place states at the right levels to implement the desired behavior.

The following snippet shows the rendering method of the top component of the hierarchy:

```
render: function () {
  return (
    <div className='outer-div'>
      <ResourcesPanel
        configs={this.state.configs}
        mapping={this.state.currentConfig}
        onUpdatedMapping={this.handleUpdatedCurrentConfig}
        channels={this.state.channelsCatalog}
        selectedChannel={this.state.selectedChannel}
        onSelectedChannel={this.handleSelection}
        onAddRow={this.handleAddRow}
        onChangeConfig={this.handleChangeConfig}
        onDeleteConfig={this.handleDeleteConfig}
        onAddEmptyConfig={this.handleAddEmptyConfig}
        onUpdateConfigName={this.handleUpdateConfigName} />

      <ChannelsPanel
        ref={'channelsPanel'}
        channels={this.state.channelsCatalog}
        selectedChannel={this.state.selectedChannel}
        onSelectedChannel={this.handleSelection}
        onSaveChanges={this.handleSaveChanges}
        onUndoChanges={this.handleUndoChanges} />
    </div>
  );
}
```

We can notice that the interface is clearly divided into two main parts, ResourcesPanel and ChannelsPanel, and each one gets passed the data needed for rendering its sub-hierarchy. This data includes both data structures for channels and mappings and callbacks to be called on this top-level component.

3.5.2.3 RoomCast app for browser

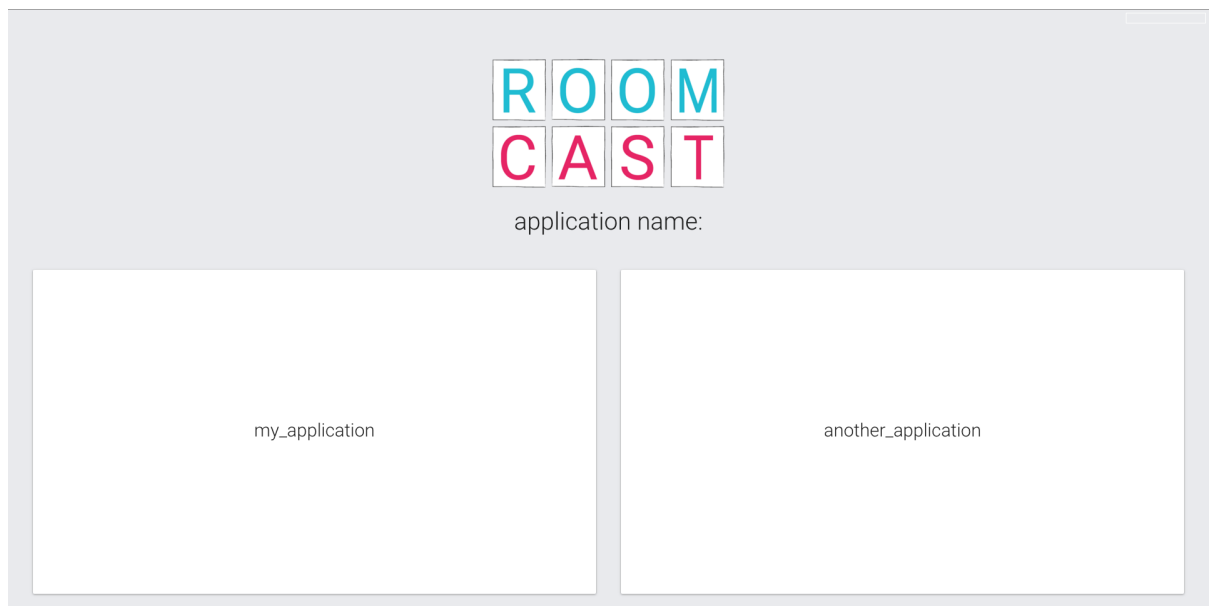
Up to now we have been describing two interfaces dedicated to developers and teachers to create and configure the distribution of resources to the class, that is the **authoring** part of the system.

We can now analyze the main RoomCast app, or *client*, that shows the actual resources to all the actors. As for the previous interfaces, the RoomCast app for browser can be launched from the nutella landing page (fig:sw-nutella-page), but also from any other page on any device simply by means of its URL. This is possible because we have abstracted the login phase of nutella to let the users authenticate themselves when first using the app.

When first launched, RoomCast app for browser shows the **login** phase (Figure 17): the user will first have to select the desired nutella application name.

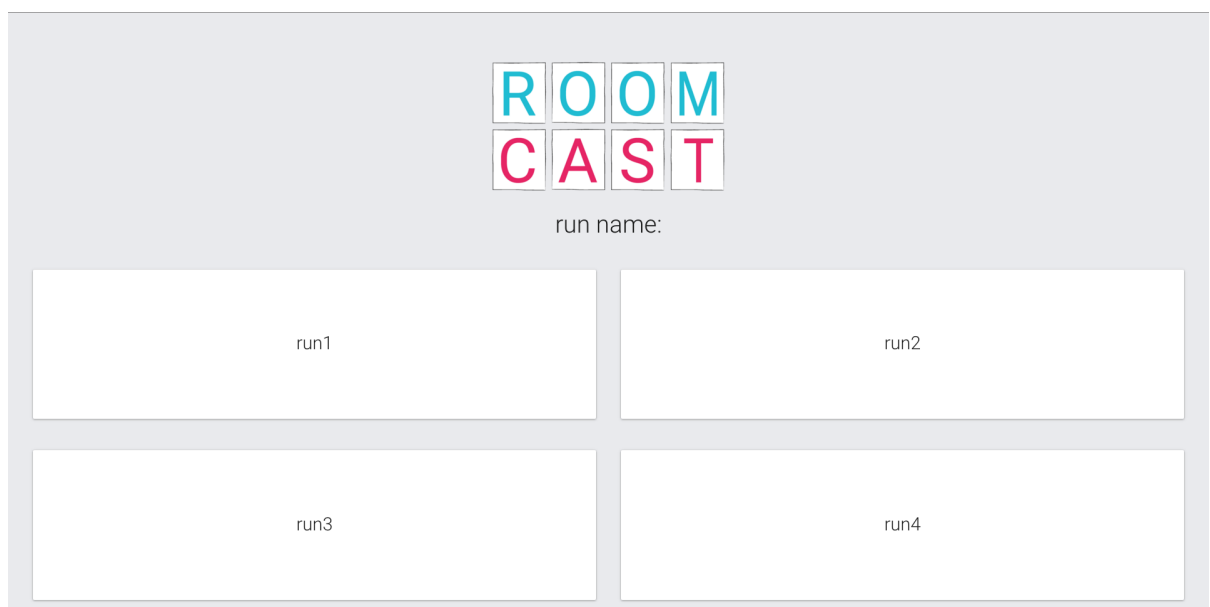
The next step (Figure 18) requires to select the nutella run name: as described in Section 3.5.1, a nutella run is an instance of a nutella application and typically corresponds to a specific classroom with its own data.

Once we are logged in the correct class, we need to select a package name: the grid of available packages (Figure 19) is the same we configured from the package creator interface for the specific activity which is currently running.



The image shows a web application interface for RoomCast. At the top center, the logo consists of the word "ROOM" in blue letters above the word "CAST" in red letters, each letter contained within a small square box. Below the logo, the text "application name:" is displayed. Underneath this text, there are two large, empty white rectangular boxes side-by-side. The left box contains the text "my_application" and the right box contains the text "another_application".

Figure 17: RoomCast app: login with nutella application name.



The image shows a web application interface for RoomCast, similar to the one in Figure 17. At the top center, the logo consists of the word "ROOM" in blue letters above the word "CAST" in red letters, each letter contained within a small square box. Below the logo, the text "run name:" is displayed. Underneath this text, there are four large, empty white rectangular boxes arranged in a 2x2 grid. The top-left box contains the text "run1", the top-right box contains "run2", the bottom-left box contains "run3", and the bottom-right box contains "run4".

Figure 18: RoomCast app: login with nutella run name.

It is important to notice that RoomCast, by requiring just a string as name for a package, allows for maximum **flexibility** when creating packages: this permits to log into the system *physical devices* (e.g. displays in the example), but also *roles* (e.g. Predator and Prey, thus *groups* of students) or single *individuals* such as students with their name or the teacher.

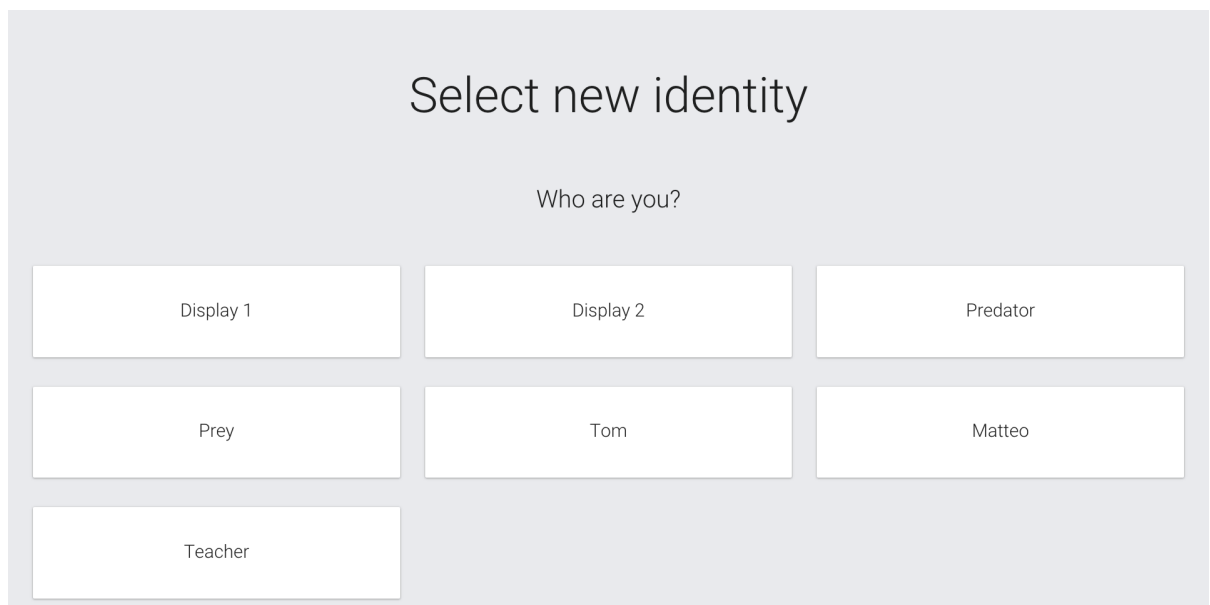


Figure 19: RoomCast app: login with package name.

Once we are logged into the system, we are able to see the main interface to play RoomCast's channels (Figure 20). The tab bar at the top of the screen contains the lineup of channels currently available; by clicking on a channel we will *play* the channel (Figure 21). In play mode, the tab bar is hidden and we have only two RoomCast buttons placed over the displayed

channel: the pink one allows to pull down the menu to switch between channels, while the blue one allows to reload the channel if needed.

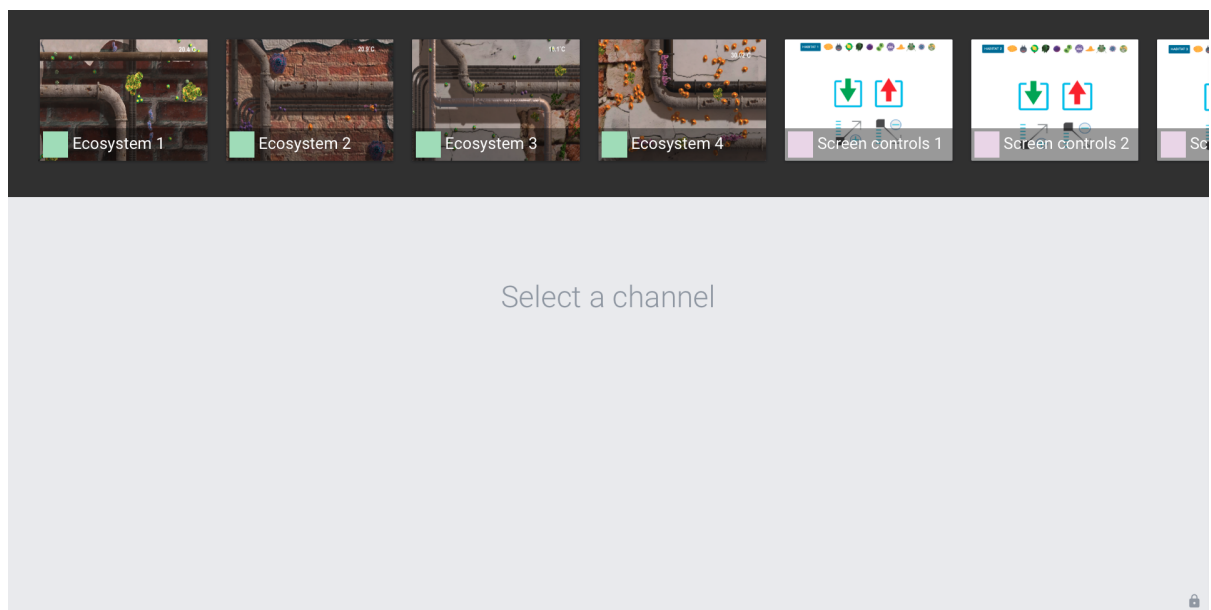


Figure 20: RoomCast app: tab bar for channel selection.

Affordances

- Login into nutella with application, run and package names (grids for selection)
- Show/hide tab bar (button)
- Play channel (from tab bar)
- Reload current channel (button)



Figure 21: RoomCast app: channel during execution.

- Select new identity (package name) when transitioning between activities (grid for selection)

Educational affordances

- Decide which resource (channel) to use at any time, among the available resources
- Visualize and/or interact with a channel to use that pedagogical resource
- Participate to regrouping phases while transitioning between activities

Implementation details

- **Single page application:** as we have seen, the RoomCast app for browser is implemented as a series of different screens (three steps of login, main interface, selection of

new identity), but we chose to use an **implicit routing mechanism**, which swaps pages when needed by modifying the same page at DOM level. There's no explicit routing between different pages, but just a re-rendering of the single page. This guarantees that there's always one single entry point for the app, through the login steps, and JavaScript takes care of storing and using the login values to advance to the next screens.

- **Internal list of “Player” components:** each RoomCast channel, in the browser version, is not simply an iframe that embeds the resource, but a more complex custom React component called “Player”. The app holds a data structure of many Player objects, one for each channel that has been launched. In this way, when the user wants to switch back to a channel that he had already started using in the past, even if it had been removed from teacher and added back again, the internal state of that channel is maintained and the user can continue his work.

The list of players is cleaned from old channels every time a channel is deleted from the catalog. The state of each Player component contains the boolean values **playing**, true if that channel is the one currently running, and **loading**, true for the period of time between its launch and its full loading. A custom RoomCast animation is displayed while loading.

3.5.2.4 RoomCast app for iOS

The RoomCast app for iOS is a standalone iOS client application that connects to nutella. This is why in Figure 8 it is one of the few RoomCast modules that are not shipped inside the nutella framework.

This application was meant to provide the exact same user experience as the browser version, but with a few significant enhancements that leverage the iOS platform.

To do this, we built a custom **hybrid** iOS application: it embeds the very same web interfaces as the browser version (login, player of channels, menus), that continuously communicate with the native part, written in **Swift**, to provide the following additional features:

- **Storage of the login values:** the values for application, run and package names are stored in the internal memory of the device, so that the login phase will be shown only after a manual logout. This is especially useful on mobile devices which usually belong to the same person for a long period of time.
- **Modal screens for new activities:** when new activities are launched, we leverage native iOS transitions and modal views to prompt the screen to select a new identity.
- **Native iOS channels:** this is the most powerful capability added to RoomCast. If we set a *custom url* for a channel card in the channel creator (Figure 13), when played from an iOS device that channel will be launched as a native application. This means that this kind of resource will be able to leverage any capability offered by native iOS applications, creating a wide range of possible new technology-enhanced pedagogies.

A native channel will have to be built by developers by using a **RoomCast iOS channel template**, which uses built-in code to provide a RoomCast button that will let the user return from the external channel's context to RoomCast's app context. This happens seamlessly and the user experience is the same as the one obtained when using traditional web channels.

An additional feature is that the RoomCast iOS channel template automatically blocks a channel app from being launched outside of RoomCast: it is necessary to go through the system to reach the desired resource, thus reinforcing the idea of controlling the distribution of content to the class.

In addition to this, even most of RoomCast’s web buttons were replaced by native iOS buttons to contribute making the hybrid app as much as possible feel like a native one, but keeping the overall “RoomCast experience” consistent with the browser version.

Affordances

The app presents the very same affordances as the browser version, since it is meant to provide a consistent experience, enhanced for mobile devices.

Implementation details

The **hybrid iOS app** is composed of a set of web views that embed web pages (the same found in the browser version). Each web view is contained in an iOS view controller, and view controllers are managed with native Swift code to transition between them. Particular attention was put to architecture the 2-way communication system between JavaScript and Swift.

From the iOS side, Swift allows to execute JavaScript calls on a web view. To leverage this, we created an exit point from the React.js code that could be reached from Swift, i.e. from outside of the React life cycle. In the following snippet of code, for example, Swift is calling the JavaScript function *requestPackageId()* on the top component of the React hierarchy, which is reachable from the global variable *ReactMain*:

```
func requestPackageId() {
    let script: String = "ReactMain.requestPackageId()";
    self.webView.evaluateJavaScript(script, completionHandler: nil)
}
```

On the other side, JavaScript will have, after the needed operations, to talk back to Swift. This was implemented by leveraging a mechanism of “mock” custom url calls that are intercepted by Swift and handled by calling native code, as if they were actual native function calls. In every JavaScript interface, we have a *mixin* component that provides a simple utility to make an iOS call:

```
iOScall: function(actionType, actionParameters) {
    var appName = 'roomcast';
    var url;
    if(actionParameters) {
        var jsonString = (JSON.stringify(actionParameters));
        var escapedJsonParameters = escape(jsonString);
        url = appName + '://' + actionType + "#" + escapedJsonParameters;
    } else {
        url = appName + '://' + actionType;
    }
    document.location.href = url;
}
```

This allows, at any point inside JavaScript and given a name for an action that will be recognized by our Swift code and some parameters as additional values, to make a call of the type `iOScall(actionName, actionParameters)`; this will directly trigger the respective Swift code that handles that specific action.

3.5.2.5 RoomCast teacher controls

At this point, the reader might have noticed that we have described the creation and use of different *activities*, that is configurations of the system, but we have not seen how these activities can be actually launched during class time. This is achieved by means of the RoomCast teacher controls interface.

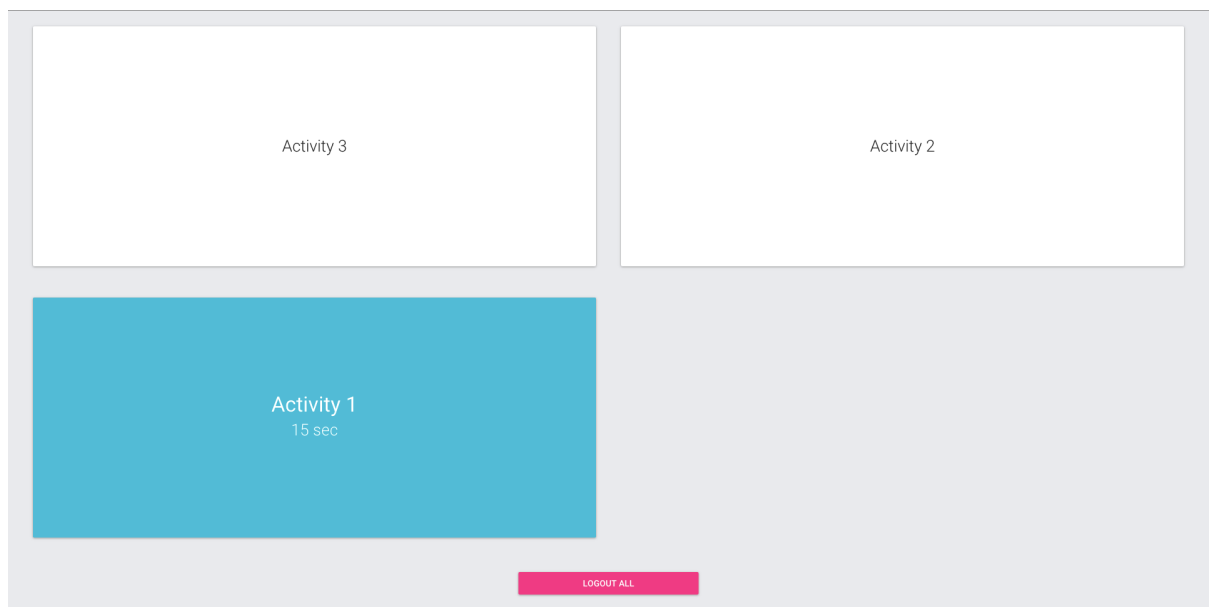


Figure 22: RoomCast teacher controls: the selected activity is the one currently running.

The interface was meant and designed to be the most simple and intuitive (Figure 22): it just shows, as a grid of tiles, all the activities currently existing for a specific run of an application.

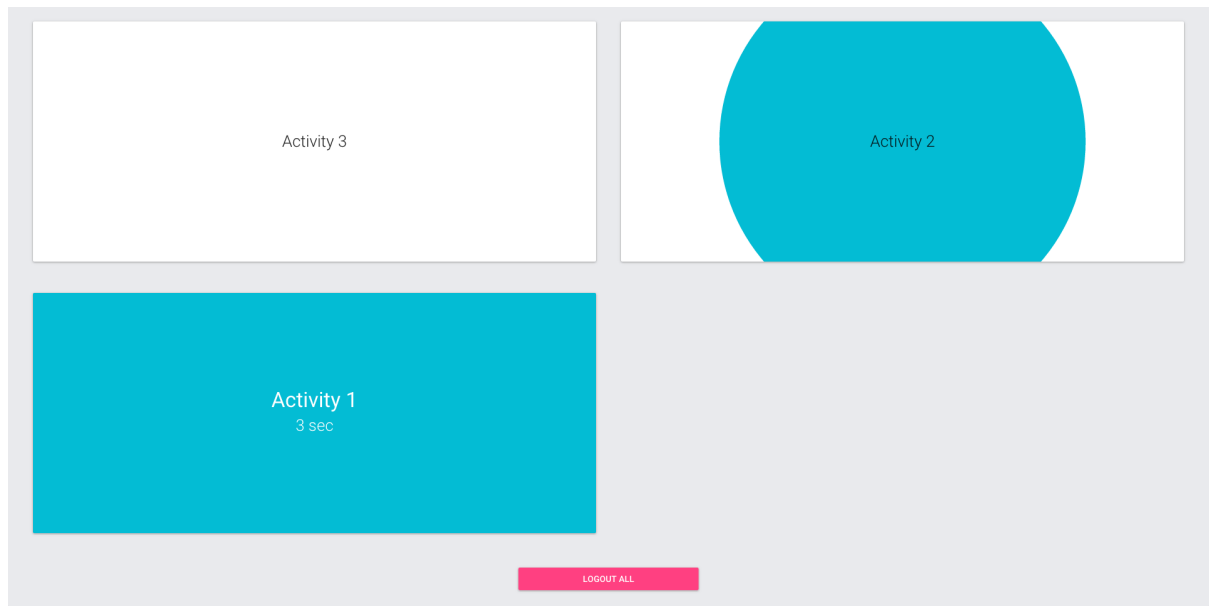


Figure 23: RoomCast teacher controls: activity in the process of being launched.

By long-pressing for 3 seconds on a tile (Figure 23), the correspondent activity will be instantly launched: from this point all the RoomCast clients that will connect to this run will be able to see the set of packages that have been assigned to them *in this new specific activity*. It is a sort of **remote control** for the teacher.

There are two alternative possible cases that can happen here and are properly managed by the RoomCast client app:

- the package name (identity) with which the device is logged into the system still exists in the new activity: in this case the transition of activity will be **transparent** to the user,

who will just see the available channels in the tab bar automatically updating according to the mapping assigned in the new activity;

- the package name (identity) with which the device is logged into the system does not exist anymore among the set of packages in the new activity: in this case the transition will be **signalled** and the user will be prompted with a screen to select a new identity, in the same way as when choosing a package name during the login phase.

This double choice brings to profound differences in terms of orchestration: just by deciding, from the package creator, whether to maintain the same names for the packages or not across activities will influence the behavior of the system. This behavior will be specific of the state of each device, so we could have one iPad that keeps its package name and then only receives the new updated list of channels and a public display that loses its role has to select a new one.

In addition, we have a timer that keeps track of how long the current activity has been going on and a “Logout all” button that allows the teacher to log out every client app from a RoomCast session.

Affordances

- Launch a new activity or re-launch the current one to reset timer (animation while pressing the tile, this is required as a **safety measure** to prevent unwanted touches that might launch activities).
- Log out all the device using a RoomCast client app on this run (button)

Let us underline that the simplicity and limited number of affordances of this interface has a precise meaning: the teacher controls represent the module of the system which are used at **runtime** by the teachers to quickly issue commands that influence the flow of the enactment of the pedagogy. This is a fundamental part of the **online** use of RoomCast and to satisfy these requirements the interface had to be immediate for the teachers to understand and use.

Educational affordances

- Keep track of the duration of each activity (**time** variable)
- Modify the availability of pedagogical content (by updating the resources (channels) available on each device when a new activity is launched)
- Force regrouping of students or change of roles (on devices that lose their package name during a transition of activity)
- Force or signal the end of class time or sessions (by logging out all)

Implementation details

- **Flex grid of tiles:** the grid of activities designed for this interface (that is the same component used also for the grid of the login phase of RoomCast app) has been designed to accommodate any number of tiles, correspondent to activities added from the package creator (Figure 24).

The implementation puts together two mechanisms: first, from JavaScript we calculate with an algorithm the number of tiles that have to be placed in the rows and columns of the grid and we derive the size that each tile should have:

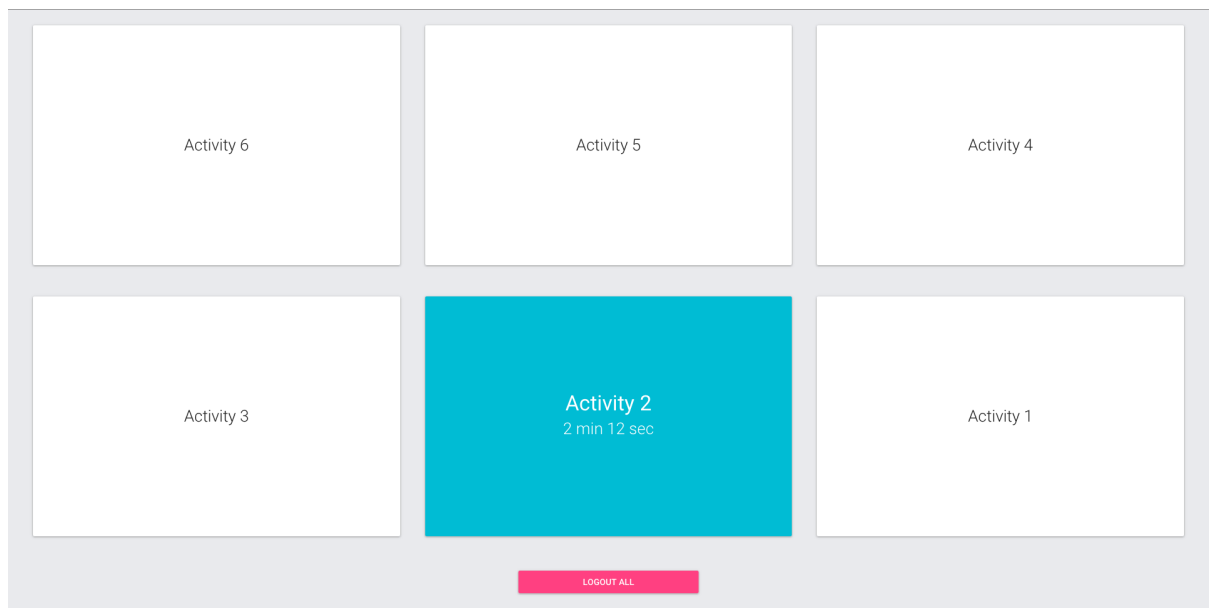


Figure 24: RoomCast teacher controls: a flexible grid of activities.

```

var N = Object.keys(configs).length; // num of total tiles
var i = 1; // num of rows
var j = 2; // num of columns
for(var n=1; n<=N; n++) {
    var p = i * j;
    if(n > p) {
        if(i < j) {
            i++;
        } else {
            j++;
        }
    }
}
var tileWidth = (window.innerWidth - this._externalMargin*2
                 - this._tileMargin*2*j) / j;

```



```
var tileHeight = (window.innerHeight*this._heightRatio
                  - this._externalMargin - this._tileMargin*2*i) / i;
this._tileSize = [tileWidth, tileHeight];
```

Second, we use an innovative CSS3 feature, flexbox (developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout), to control from CSS the aspect of the grid, i.e. position, direction and alignment:

```
// flex grid (container) properties
display: flex;
flex-direction: row;
flex-wrap: wrap;
justify-content: flex-start;
align-items: center;
align-content: stretch;
```

3.5.3 RoomCast bot

In this section, we will describe RoomCast bot, i.e. the core bot that acts as a hub for most of the messages exchanged throughout the system and as the manager of the persistent data that represents the *state* of each RoomCast instance.

The bot was implemented in the Ruby language, like all of the other framework-level bots and the core of the whole nutella framework.

The main purpose of the bot is to act as a mediator of the communications between all the modules of RoomCast. In particular, the bot has to persistently store data such as information on the channels and configurations that power the system. Every time this data is updated from one of the interfaces, a message is sent to the bot, which has a routine (associated to

that specific MQTT channel) that executes the needed operations, among which it stores the updated data. Finally, the bot publishes a message of acknowledge, containing the new data (or the needed parts of it), that is broadcasted back to the whole system.

Data structures

Let us first describe the two main data structures that represent the **state** of the system:

- **channels.json**: it's the representation of the complete lineup of available RoomCast channels or what we usually call *channels catalog*. It stores data such as name, description, color of the icon, image and URL for each channel. This structure is a *map* or *dictionary* which, access via a channel id (*key*), will return the content of that channel.
- **configs.json**: this is a more complex data structure that stores three main pieces of information:
 - **configs**: the complete specification of the packages together with their associated channel ids, that is the set of mappings. We just need to list here the id for every available channel, since this will reference, as *foreign key*, the actual channel in the channels.json data structure
 - **currentConfig**: the id of the current running configuration (or activity)
 - **launchTime**: the time when the current running activity was last launched (total seconds elapsed since the Epoch)

Appendix A provides the complete **JSON Schema** specification of these data structures, describing their expected fields, properties and patterns. *JSON Schema* (json-schema.org),

based on the concepts from XML Schema (XSD), “specifies a JSON-based format to define the structure of JSON data for validation and documentation”.

The reader might have noticed that we are using and directly storing JSON data structures: we are indeed leveraging the nutella **persistence** APIs, that are a wrapper around the standard Ruby *PStore* class. “PStore implements a file based persistence mechanism based on a hash, where user code can store hierarchies of Ruby objects (values) into the data store file by name (keys)”. nutella adds an additional layer that let us directly store JSON objects. The transactional behavior of PStore ensures that any changes succeed or fail together: this can be leveraged to make sure that the data stored is not left in a transitory state.

We have to clarify that this design choice was possible because we are likely to be always dealing with data structures that don’t reach huge file sizes, don’t become extremely complex to manage as a unique chunk and that are not used, especially for write operations, with high frequency of access. For these reasons, using PStore was the simplest and most lightweight solution that well supported the current state of implementation of the system. In the future, if needed, it will be possible to substitute the implementation of this part with another technology for persistence without much effort related to integration.

Messaging architecture

Now that we have a clearer understanding of the data structures, we can analyze how these are created, updated and kept synchronized among the different modules of the system.

Figure 25 shows an overview of the data (sent as messages) exchanged between bot and interfaces. This schema is very simplified with respect to the actual implementation and it has to be considered an abstraction of the main exchanges of data in the system.

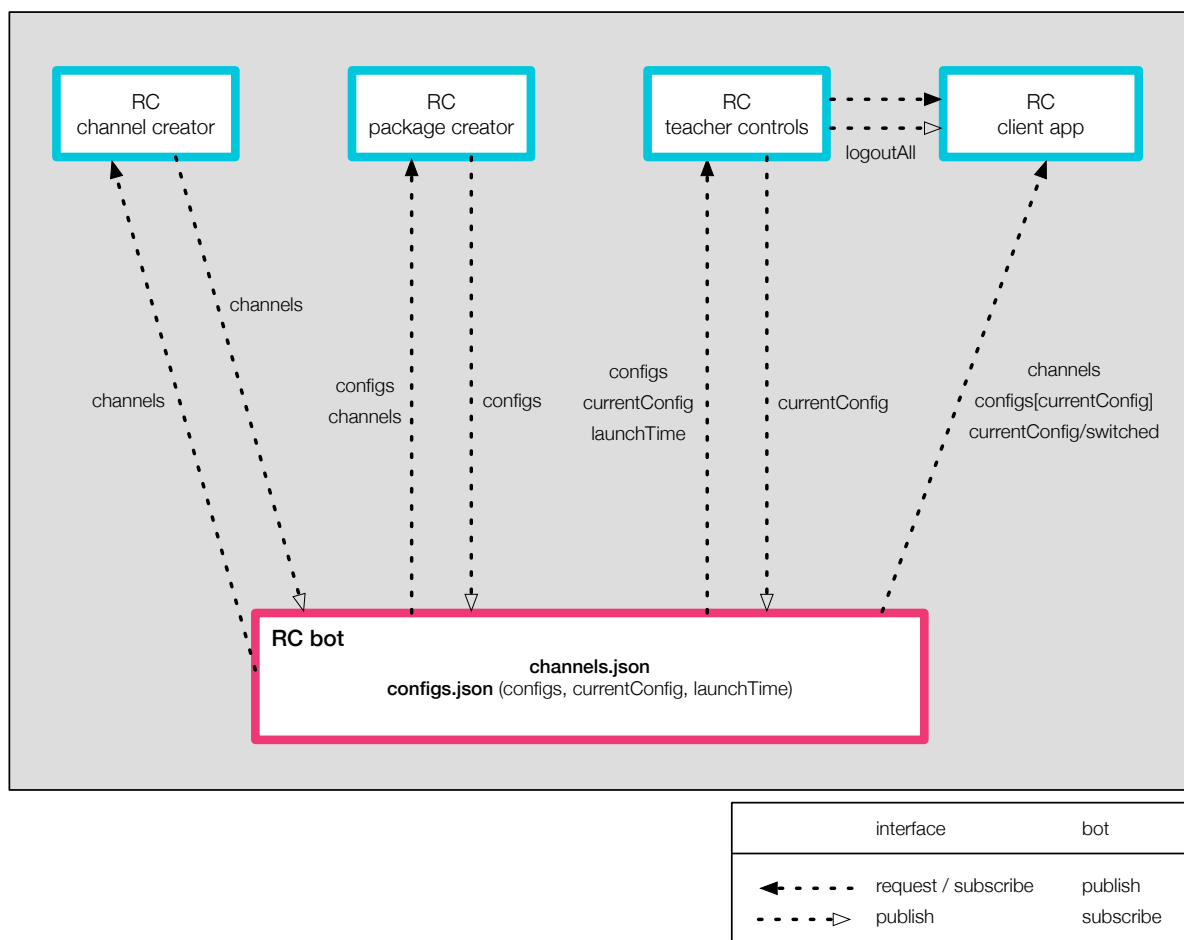


Figure 25: Summarized view of the data (messages) exchanged between bot and interfaces.

Finally, let us show how the implementation of these features generally looks like. The following snippet of code shows the way in which RoomCast bot can subscribe to an MQTT channel, in this case the one that receives messages with the updated data structure of configurations:

```
nutella.f.net.subscribe_to_all_runs('configs/update',
lambda do |message, app_id, run_id, from|
```

Based on the specific run from which the message was sent, the bot will receive, manipulate and store the data *for that specific run*:

```
configs_db = nutella.f.persist.get_run_json_object_store(app_id, run_id, 'configs')
```

This allows the bot to manage, with the same code and routines, all the different runs *separately*, but potentially also in parallel. This means that if we have multiple runs for many classes working contemporarily, their RoomCast sessions and data will be managed completely independently, but by the same framework-level bot.

Under the hood

If we want to analyze more in depth the implementation of RoomCast bot and the technical challenges that were solved to have it work reliably and according to the specifications, one of the best examples is represented by the **synchronization between data structures**.

As we have seen, the system relies on two main data structures, *channels* and *configs*, that have been designed to be managed separately, based on the most convenient way according to their usage by the different modules of the system.

Even if they are meant to work independently (and joined when needed), in a few cases we have to deal explicit synchronization problems between them. The most evident case is the

one in which someone *deletes a channel* from the channels catalog, i.e. `channels.json`. This will have to reflect immediately also on `configs.json`, since the configuration file still potentially contains the id of that deleted channel inside some of its mappings to packages. The technical solution to deal with this was to implement a routine inside the bot which is executed every time the data structure of channels is updated, to look for potential ids to be cleaned inside the configurations data. After the cleaning process, a message is published to notify the update of the configurations data. Here we have the second issue: since the two data structures are independently kept synchronized with the different parts of the system through the messaging technology, there are two alternative possibilities based on the order with which the messages are received by any single interface:

- the interface receives the updated configurations before the updated channels: this case is safe, since it will only use the updated ids of the mappings (only new channels) to access the channels structure;
- the interface receives the updated channels before the updated configurations: in this case we would get an error when trying to access the channels structure with an id of an old channel that no longer exist. This is dealt by the interface simply by adding a check on the existence of the key inside the channels structure.

To sum up, the idea behind all these strategies is that the bot manages the macro-routines to keep the data *consistent*, while the *synchronization* happens as a consequence, automatically, we just need to manage single specific edge cases that might arise when two messages can be received subsequently, without a fixed order, but influencing each other.

3.5.4 RoomCast logging system

If we observe once again the overall software schema (Figure 8), we can notice that we are still missing the analysis of one module of RoomCast, the one that looks to have a different position inside the architecture with respect to all the other components: RoomCast log bot.

This is a nutella bot that have been designed to **log user interactions** that happen inside the RoomCast interfaces. The primary purpose of this is to create a history of the usage of the system which allows for many different analyses and evaluations, e.g. to understand **patterns of usage** of the tool, to generate statistics or to investigate what happened at a specific point in time.

The bot, implemented in JavaScript (Node.js), is not placed at framework level, but has to be inserted inside each specific nutella application in which we want to leverage its capabilities (run-level). This choice was made in order to externalize the logging system from the core RoomCast modules inside the framework: the log bot will be added to a project (simply by downloading it, off-the-shelf, as a nutella plugin) only when needed and it will store its recorded data to a dedicated database that belongs to the **local data** of each single run, thus being completely separate from the core RoomCast framework data.

RoomCast log bot stores data to a dedicated **MongoDB** database. MongoDB (mongodb.org) is a document-oriented, non-relational database technology. “A record in MongoDB is a document, which is a data structure composed of field and value pairs” and similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

We chose to use this technology for the logging system since it is well suited to both store data in JSON format and also to later query that data by means of ad-hoc operations.

The **logging system** is divided in two parts:

- the **log bot**, whose sole purpose is to listen for messages on the MQTT channel 'roomcast-log-bot/store' and to store each new received message as a Mongo document representing a single user interaction:

```
nutella.net.subscribe('roomcast-log-bot/store', function(message, from) {
    storeMessage(message);
});
```

- a **mixin JavaScript component**, which is placed inside the code of every RoomCast interface, giving it the capability to log a user interaction by just calling a dedicated function:

```
// #LOG action
this.logAction('playChannel', app_id, run_id, {
    channel_name: this.state.channelsCatalog[channelId].name,
    package_name: this.state.packageId
});
```

The logAction function takes as parameters only the name of the interaction we want to record at a specific point of the code, together with an additional object that contains additional information specific to that action.

It is interesting to notice that all the important information about login values, time and device will be automatically computed and added by the logAction function. The following

JSON object represents an example of Mongo document stored in the database after a `logAction` call for the action of starting to play a RoomCast channel:

```
{
  "action" : "playChannel",
  "app_id" : "my_application",
  "run_id" : "run1",
  "device_id" : "9fqaa859.zec",
  "time" : {
    "timestamp" : "2015-10-19T13:20:58.447Z",
    "year" : 2015,
    "month" : 10,
    "day" : 19,
    "time" : "8:20:58"
  },
  "info" : {
    "channel_name" : "Predator Actions",
    "package_name" : "Predator"
  },
  "_id" : ObjectId("5624ee3afdfab6994ffa9f49")
}
```

In the document shown above, the “info” field contains the object that was passed manually from the code specific to a type of action. Finally, the “device_id” field represents a **cookie** that is stored in the browser the first time the interface is launched: this is a simple, but effective, way to distinguish user interactions originated from **different physical devices**, even if it is not currently possible to know the actual identity of a specific device.

Let us briefly list the user interactions that are currently logged by the different interfaces (where we specify, for each action, the name and the additional information that is stored for it):

- **RoomCast client apps:**

- playChannel [channel_name, package_name]
- login [package_name]
- logout [package_name]
- transitionActivity [old_package_name, new_package_name, transparent],
where *transparent* is a boolean that is true if old_package_name coincides with new_package_name, that is when the user is not asked to select a new identity during the transition

- **Teacher controls:**

- logoutAll
- launchActivity [activity_name]

- **Channel creator:** it doesn't currently support logging since this interface is mostly used offline to restructure the lineup of channels, thus not much interesting to study from the perspective of the user interactions.

- **Package creator:**

- savePackageCreator [configuration]

We will see in Chapter 4 an actual example of how to leverage RoomCast’s logging system during an evaluation study.

3.5.5 RoomCast nutella APIs

As one of the main components and services of the nutella framework, RoomCast provides APIs to be directly used by developers of nutella applications. These APIs are dedicated to getting information about the *activities* that are executed, through RoomCast, inside the classroom.

Any component in a nutella application can register a callback that is triggered whenever there is a configuration change, that is, from an educational perspective, an activity change:

```
nutella.cast.configurationChange(function(prev_config, new_config) {
    // React to change of configuration/activity
});
```

Components can also actively query RoomCast for the current activity at any moment:

```
var cc = nutella.cast.currentConfiguration
    // Use the configuration value, cc
```

3.6 Orchestration analysis

In this section, we will perform an analysis of the system and its design from the point of view of the support for orchestration. To do this, we will make references to the theories and technologies presented in Chapter 2 with the purpose of comparing the proposed design with respect to previous solutions, by showing both common traits and unique features.

This analysis represents a further step in order to answer our research question.

Let us start from the title of this work. RoomCast is mainly composed of two aspects: as we have seen while describing the functionalities, it is both an **authoring environment** and a **runtime system**. If we go back to the musical metaphor of orchestration (see also Section 2.1.1), with an extended interpretation we could say that by means of the authoring environment the orchestration is *planned* or designed, while with the runtime part of the system it is *performed* by the teacher during the enactment phase. Moreover, as we have already underlined, these two capabilities of the system are not disjoint, as the two opposite facets of the metaphor would lead us to think: one of the core ideas of RoomCast is that at runtime we can not only use the dedicated tools for the teacher, but also leverage the very same authoring environment used to plan before class to dynamically modify the *design* of the activities and adapt to occurrences that may manifest themselves during the enactment phase.

In order to develop a structured analysis of the support for orchestration provided by RoomCast, we will use the “**5+3 aspects**” **framework** introduced by Prieto in (5): this is a commonly used tool for researchers in this field (51; 92) for analyzing the aspects of orchestration and successively performing evaluations of how a system is able to effectively implement such principles in real learning settings. For a detailed explanation of the framework, please refer to Section 2.1.3.2. Based on the framework, we can identify features and capabilities (correspondent to the technological and educational **affordances** of the interfaces that we have described in Section 3.5.2) linked to specific aspects among the 5+3 mentioned by Prieto:

- **Design/planning:** in RoomCast, the design/scripting phase is performed by means the **authoring interfaces**, that is channel creator and package creator. Here the user can

modify as desired the catalog of available channels and then create activities, that is configurations of packages containing sets of channels selected from the catalog. Each package within an activity represents an identity or role which is used during the login phase to identify the device that is connecting to the system.

It is important to notice, once again, that the *channel*, which is the basic unit of content that is distributed by the system, is a screen-based interactive interface that represents a core pedagogical resource whose design is outside of the scope of RoomCast. With RoomCast, we **orchestrate the distribution of channels**, that is how and when they are delivered to the *ecology of displays* that populates the classroom.

- **Regulation/management:** this aspect is implemented throughout the entire system from many different perspectives:
 - **Resources management:** the main regulation mechanism of RoomCast is the selective assignment of channels to packages; this represents a sort of **parental control** system managed by the teacher which allows to enforce how the pedagogical resources are used by the students and groups inside the class.

This aspect also represents one of the unique capabilities that the system introduces and it will be interesting to evaluate how this feature can contribute the discourse on orchestration.

 - **Workflow management:** creation of different activities, that is configurations of channel packages, to be staged in the desired order and launched at runtime when needed.

One of the possible designs that use multiple activities could, as an example, leverage **transition activities**, a concept derived from the Learning Sciences literature (101) to indicate activities useful to either break a period of instruction that requires high concentration or to switch to new phases of the instructional unit.

- **Group management:** channel packages represent identities that the devices can assume and, as a consequence, they attribute a role or identity to the group of actors in front of them too; it is interesting to notice that groups' size can range from individual actors, e.g. **personal devices** for teachers and single students, to teams with many students, e.g. performing collective inquiry in front of a **public display**. In addition, the capability of **regrouping** during a transition of activity offers the way to restructure at runtime the composition of groups including, potentially, their granularity. This implements the support for orchestration at different **social planes** mentioned by Dillenbourg (8; 3).
- **Time management:** as we have seen, the timing of activities is implicitly taken into account when designing the activities, since they represent different advancements of the instructional unit that can be launched during enactment when needed. In addition, we also have explicit monitoring of the time elapsed for a running activity directly from the teacher controls interface.
- **Management of instructional units over multiple classes and login phase:** even if, in the current implementation, the creation (scripting) of different *applications* and *runs* (i.e. classes) is executed directly from the underlying framework, *nutella*,

by researchers or ICT experts, RoomCast has the fundamental role of retrieving the data about the running nutella instances and logging the user into the appropriate context. Once logged into the correct RoomCast context, each device is able to be managed according to the RoomCast setup for that specific setting.

From these different aspects, we can notice that this regulation, from a high-level point of view, is about the management of two critical variables: **people** (or actors) and **time**.

Peculiar to RoomCast is the characteristic that regulation is the critical part of the orchestration: if we consider the design phase, that planning is mostly not about designing the core pedagogies (that are instead expressed with the support of the channels and possibly described by their own scripts outside of the orchestration system), but it is about defining **how actors and resources are managed throughout different activities over time**. In other words in RoomCast orchestration is less about the core instructional design and more about the coordination and regulation of the scenario.

Finally, we must highlight that by tuning the set of available resources for the students, the teacher can design for different levels of regulation of the learning activities: for example, by leaving a margin of freedom to how the students can approach a problem by means of different resources, the teacher can allow a certain degree of **self-regulation** of the students (8).

- **Adaptation/flexibility/intervention:** in RoomCast, the support for flexibility is all about the capability of dynamically changing the distribution of channels to the displays. This is meaningful because it allows the teacher to flexibly restructure the way in which

the pedagogical resources are used. To make these changes, the teacher can embed the support for flexible instructional design, through the use of multiple activities, in the planning phase and then successively launch them during enactment from the teacher controls. An even more flexible way to adapt to unpredictable or emergent occurrences is to use the package creator at runtime to redesign the distribution of resources as needed, for example by assigning a new tool only to the first group that completes a previous assignment.

This flexibility is not only restricted to changes to the assignments of channels to existing packages, but also to all the capabilities of the authoring environment, such as redesigning activities and roles/groups on-the-fly. In this sense, it will be interesting to evaluate how these features for adaptation are actually used in a real setting where all the constraints of the scenario slow down and limit the ability of intervention of the teacher.

- **Awareness/assessment:** from an orchestration point of view, a form of awareness is provided by the package creator itself which tells at any time the teacher what is happening inside the class in terms of groups, activities, and resources available.

In addition, RoomCast, due to its nature of content provider, enables a very interesting form of awareness: given that in the lineup of channels we can insert tools dedicated to **monitor** any status or advancement of the class, including statistics on students' results or even detailed *dashboards* for the teacher, we can distribute these channel tools only to selected actors in the scenario. This leads to realizing two types of awareness: a **public awareness**, obtained by distributing class-wide information to all the actors, e.g. to every

public/personal display or to a single public display used by the whole class, but also a **private awareness**, for example by giving special assessment tools only to the teacher.

- **Roles of the teacher and other actors:** in RoomCast, in accordance with the models found in the literature of orchestration, the teacher is the guide responsible for the setup and coordination of the learning setting. In this sense, we have seen in Section 2.1.3 that there is a correlation and synergy between the idea of a teacher-centric orchestration together with the enactment of constructivist pedagogies.

However, the aspect that is peculiar to RoomCast is the flexibility with respect to how the teachers can **tune the level of guidance** in the class: even if the teacher is always at the center of the orchestration process, she can decide, just by leveraging the core capability of the system of selectively assigning pedagogical resources to individuals or groups of students, to what extent the students have freedom in using the set of available resources. In this way, based on the context and necessities, the teacher can dynamically decide the **type of orchestration** in the continuum between the completely teacher-centric and the more learner-driven positions, thus potentially allowing, as we have seen for the regulation aspect, for some degree of self-regulation of the students.

Finally, we can say that this flexibility with respect to the role of the actors and guidance of the teacher is allowed by the **open-ended design** of RoomCast, an idea that we already found in the ‘Fireflies’ interaction design by Bakker (see Section 2.2.4), meaning that the great flexibility of the system allows to design orchestrated learning scenarios with a wide

range of possibilities (e.g. in terms of how to think the use of different packages, channels, and activities) and without enforcing one specific way of usage of its capabilities.

- **Pragmatism/practice:** RoomCast is designed to be pragmatically used in practice and taking into account the variety of constraints that might influence the ability of the teacher to coordinate the scenario. In this sense, the teacher controls offer an extremely simplified way to execute basic operations to orchestrate and adapt to occurrences, but still considering the very limited amount of time that the teacher has during enactment. Moreover, all the interfaces have been designed to try to reduce the **cognitive load** for the teacher, in order to be used effectively at runtime, but also to **minimize the learning curve of the teachers**, thus being practically usable, after a short training period, by every average and non ICT-expert instructor.
- **Alignment/synergy:** since RoomCast is about distributing screen-based software channels, the main way to integrate scaffoldings in a synergistic way is to leverage this capability by designing a workflow of available resources, possibly staged across multiple activities, that best implements the desired pedagogies. As an example, we could first give the students a channel with a simulation to be analyzed, then once ready they would get access to a second tool to see some analytical data about that simulation; finally, they would get an interactive tool to make changes to the simulation: this would require them to use the previous channels in an integrated way to get useful insights and understandings on how to apply these changes.

Let us focus on how the **integration** of resources is supported by RoomCast. If on one

side the implementation of the integration between different channels in terms of how they communicate and exchange data is left to the developers of those channels, on the other hand, RoomCast plays a fundamental role with respect to the **integration of the channels into the orchestration system**: RoomCast, when launching a channel, takes care of passing to the channel critical information such as name of the application, name of the class, name of the package/identity that is requesting that channel to be launched and also potentially any additional custom parameter that the developer of the channel might need. This data clearly has a key role in the implementation of synergies, since it allows to distribute **parametrized pedagogical resources**, that is content that can be personalized based on the specific context and even on the single screen or group of students that is requesting it.

Finally, we should also notice that the channels, being potentially represented by any interactive software resource, could include highly complex tools: we could even think of integrating, if needed for a particular instructional design, an additional orchestration tool, e.g. a set of boards to take notes such as GroupScribbles (see Section 2.2.5), to complement RoomCast in the orchestration process.

- **Models/theories**: RoomCast takes inspiration from the existing principles and models of orchestration that we have seen in Chapter 2. Even if the researchers in this field are still searching for a universally shared and comprehensive framework for orchestration, RoomCast represents a small **paradigm** for the specific context of orchestrating multi-display-based classroom settings where screen-based software resources with pedagogical

value are distributed to students and teachers. This paradigm is centered around the **content provider metaphor**, which is supported both in terms of basic notions such as ‘channel’ and ‘package’, but also with the help of **user interfaces** that are designed to make users build a specific **conceptual model**, for example with the card used throughout the system to graphically represent a channel. Overall, indeed, this pervasiveness of the metaphor helps to shape “the mental models that different actors have about how the scenario should be orchestrated” (69): we argue that this helps both teachers and students understand their roles and capabilities while using the orchestration technology, thus, finally, improving the usability at classroom level.

As we can notice, the ‘5+3 aspects’ framework by Prieto represents a useful tool to analyze a system for orchestration in its main functional components. However, as we have described in Section 2.1.3.2, there are some aspects that might be relevant for orchestration, but which have not been included in this framing yet. Let us highlight a few significant aspects of orchestration that we think are particularly relevant in RoomCast:

- **Motivational/emotional aspects:** we argue that the emotional states of the actors in the classroom setting, and, in particular, their motivation, can have an impact on the effectiveness of the orchestration process. In RoomCast, the motivation could be increased by factors such as a particular tuning of the level of self-regulation of the students or by leveraging the metaphor of the content provider, for example by distributing particular channels as special tools that groups of students have to be rewarded with their work.

- **Minimalism and modest computing:** as Dillenbourg states, “the technology push leads us to offer always more, but I am convinced that, from an orchestration viewpoint, less is more”. [...] The main guideline is minimalism, i.e. to avoid adding functionalities in TEL environments that are not strictly necessary, since they might increase the **global orchestration load**” (7). This need for minimalism in the design for orchestration is what Dillenbourg calls *modest computing*, while the *global orchestration load* is defined as “the total increase/decrease of teacher’s effort required by a technology for orchestrating integrated classroom activities”.

If we now look at RoomCast, the search for minimalism should be clear: it is embedded in the paradigm itself followed by the system, that is the idea of having a separation between core pedagogical design and orchestration system, where the latter cares mostly about what happens around that core instructional design (that is the **kernel**, represented by the channels). This means having a **light and flexible orchestration technology** which reduces the cognitive load for the teachers, thus being easier for them to manage. Finally, the simplicity and usability of the different interfaces, together with the help provided by the content provider metaphor, contribute to this search for minimalism.

Given the lack of studies and research on these topics, these last two aspects will be particularly interesting to be evaluated as a result of a pilot study.

Let us conclude this detailed analysis by providing a summary of RoomCast’s vision and paradigm for orchestration. A graphical representation of this is presented in Figure 26: the

reader might recall that a similar schema was used in Figure 2 (Section 2.1.3.6) to depict the role of an orchestration technology with respect to orchestrable technology.

We can notice that RoomCast, being a technology dedicated to supporting the activity of orchestrating, has to be considered an **orchestration technology**. On the other hand, what Tchounikine called **orchestrable technology** is represented by the channels managed by RoomCast: they represent core pedagogical resources (or, with Dillenbourg's words, the **kernel**) created outside of the orchestration technology, but coordinated, parametrized and distributed by RoomCast.

In particular, we argue that a few characteristics make the channels an orchestrable technology:

- they represent **modular** units of pedagogical content that can be independently and selectively distributed to the devices in the class;
- they can be **parametrized** by the orchestration system by receiving parameters from RoomCast at runtime;
- they are **reusable**, since once we have developed a channel, that is a web interface or a mobile application, we can distribute it in multiple classes or send it to other research groups just by **sharing the channel/card**.

RoomCast, moreover, represents an **orchestration layer**, that is a structure that acts like a middleware between the orchestrable technology used by all the actors and the teacher who orchestrates the scenario. In other words, we could also say that while the channels represent the

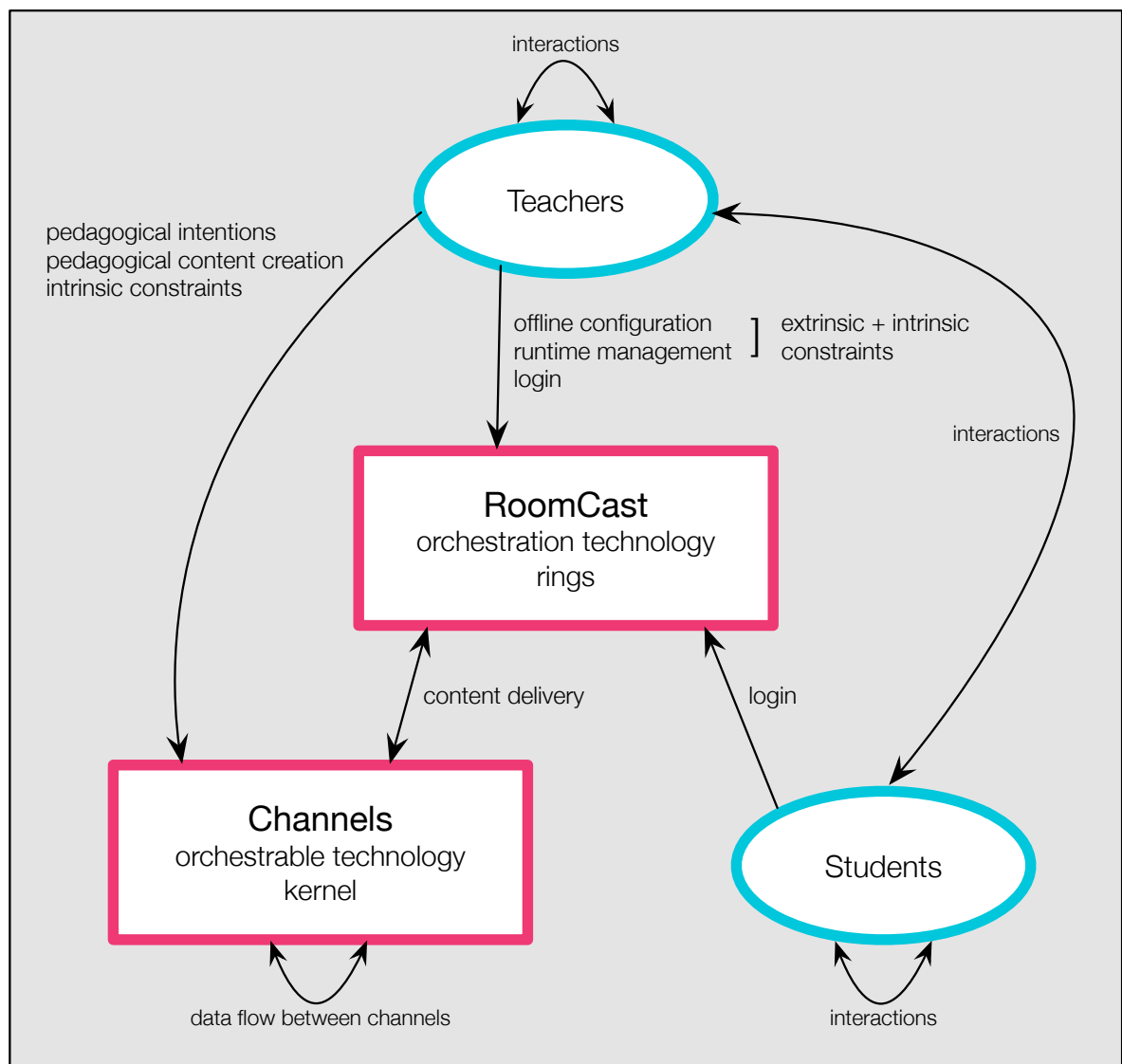


Figure 26: Graphical representation of the orchestration analysis of RoomCast.

kernel by expressing the pedagogical intentions for the class and setting intrinsic constraints, RoomCast **cares mostly about the rings**, that is the activities that happen, at different “distances”, around the kernel.

However, RoomCast has also a scripting component: the ability to plan the flow of activities and to rearrange the distribution of resources to different actors over time represent design choices that certainly have a pedagogical value. If we consider the distinction that was presented in Section 2.1.4, we can identify two scripting phases in RoomCast:

- **Primo-scripting**: even if the core pedagogical content is inside the channels, the use of the package creator as an authoring environment to design flows of activities and distribution of resources represents a light form of scripting that includes setting both intrinsic and extrinsic constraints.
- **Runtime scripting**: the use of RoomCast during enactment can include operations that have a pedagogical impact. This happens at different levels and it might include the choice of the teacher of reordering or skipping activities by means of the teacher controls, but also interventions such as modifications of the assignment of channels to specific students or to the whole class, e.g. to respond to **emergent occurrences** that require deploying configurations of RoomCast that were not predictable at design phase. As we have seen for the “kernel and rings model”, this might involve managing emergent activities, that are the nearest to the kernel.

Any other operation supported by RoomCast can be classified as **conduction**, including coordination at runtime such as starting scheduled activities at the right time or launching

channels dedicated to managing **envelope activities** needed to reinforce the learning process. Finally, the login phase and capabilities of RoomCast are useful to easily manage the common **infra activity** needed to set up the right context, necessary to support the enactment.

This section has hopefully clarified the unique traits and characteristics programmatically stated in Chapter 1, which represent the contribution that this work wants to bring.

The next chapter will be dedicated to showing the results from the field study that was carried out to test the system and to evaluate its design in an actual classroom setting.

CHAPTER 4

PILOT STUDY

This chapter describes a pilot study of RoomCast that was carried out as part of a multi-week instructional unit, called “Wallcology”, enacted in a primary school by our Learning Technologies research group. We will first present a description of the context and pedagogical structure of the unit; then, we will show in details the method that was used to gather research data and evidence to be finally able to analyze the results and provide a first evaluation of RoomCast’s support for orchestration.

4.1 Wallcology

In this section we will describe the context of the school in which the study took place, the pedagogical structure and content of Wallcology and finally the initial setup of RoomCast, in terms of channels available for this specific instructional unit.

4.1.1 The context

The Wallcology unit was enacted in a private **primary school** in Chicago. This school has a **high level of ICT resources and support**: each student has a personal laptop and tablet, classrooms are equipped with desktop computers and projectors and they support the screen-casting of the devices of both teachers and students. For these reasons, both learners and teachers are quite familiar with a wide range of technologies, they frequently use online shared documents to work collaboratively; students, in particular, are used to preparing presentations

to be projected on the walls and discussed with the whole class.

In addition, the school has a highly technological science lab, which is the main setting that was used for our study.

The Wallcology unit was enacted over a **two months** long period and for three different **6th grade** classes, each one taught by one teacher. As we will see, the opportunity of running the same experiment with three separate learning groups, composed by students of the same age but with different characteristics, both individual and as a class, revealed to be particularly useful to gather evidence for our studies.

4.1.2 Description

Wallcology was first introduced by Moher et al. in 2008 (6) and it was revised and technologically enhanced by our research group for the present study.

The unit represents an *embedded phenomena* application which consists of a multi-week simulation of **population ecology**. In WallCology, students are required to act as investigators within a complex simulated ecosystem coinciding with the classroom walls. Inside the classroom, there are four public displays (desktop computers): each one of them shows a simulation of the part of the ecosystem corresponding to the wall in front of which the display is positioned, thus pretending that the computer is actually showing what is happening inside that wall. For this reason, we call these four computers “*wallscopes*”.

Each wallscope shows an environment (that from now on we will call “*ecosystem*” followed by a sequential number) containing mold, vegetation and various species of animated creatures

crawling over brick and pipes. The characteristics of an ecosystem are different and diverse for each single wallscope.

In this scenario, students act as ecologists whose goal is to keep the ecosystem alive and thriving. In order to do so, they are required to conduct investigations to identify and classify the species and the relationships between them and to understand the dynamics of the population over time. Several topics are tackled, such as life cycle phases, food chains, predator-prey relationships, habitat selection, response to environmental change and adaptation.

4.1.3 RoomCast setup

To better understand how Wallcology works, let us start by showing the set of applications, embedded as channels inside RoomCast, that have been developed by our team for the enactment of the unit.

Throughout the unit we used 5 kinds of interfaces:

- **Ecosystem:** 4 channels, one for each wallscope, showing a 3D animated simulation of the environment.
- **History:** 1 channel, showing histograms representing the number of organisms for each species inside a specific ecosystem. The interface allows to select the time range (dates) for the observation and the ecosystem that the student wants to study.
- **Screen controls:** 4 channels, each one to apply changes to the correspondent ecosystem.

The interface allows to select a species and then to execute one among four operations:

- insert a species into the ecosystem

- remove a species from the ecosystem
 - increase the population of a species (x 4)
 - decrease the population of a species (x 0.25)
- **Tangible controls:** 4 channels, each one to apply changes to the correspondent ecosystem. They are exactly identical to the screen controls, but instead of being control buttons on a web interface, an action is activated by using physical objects (small plastic tokens called *tangibles*): a student needs to put a combination of tangibles on an iPad in front of the computer to perform an action. A combination is made of a tangible to select one of the four ecosystems, another tangible to select the species (we had actual 3D printed colored creatures) and one more tangible to select the desired operation among the four available. The communication between tangibles and iPad happens through iBeacon transmitters and the whole proximity tracking is managed by a nutella component called **RoomPlaces**.
 - **RoomCast teacher controls:** 1 channel, dedicated to the teacher. This is the RoomCast interface for runtime orchestration that was presented in Section 3.5.2.5.

Figure 27 shows part of the catalog of channels added for the setup of RoomCast to run Wallcology. We have to notice that this is the complete lineup of channels that were developed for the unit, but since each one of them embeds part of the pedagogical activities and content that have to be gradually deployed into the classroom, the channels will be distributed incrementally over time and based on the pace of learning of each class.

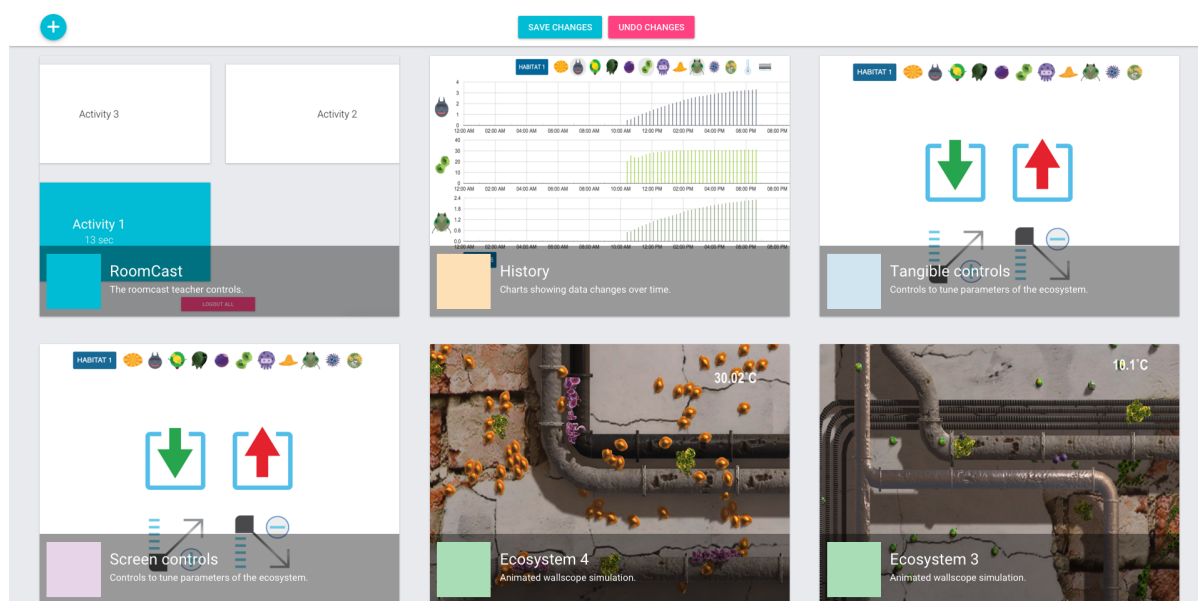


Figure 27: Wallcology: screen of RoomCast channel creator.

Finally, the students also had access, on their iPads, to a tool developed by the learning technologies group of research from the University of Toronto which allowed them to take notes and pictures about their findings during the unit. These notes were accessible by both researchers and teachers to gather research evidence and evaluate and grade their results.

In order to support the three parallel enactments of the unit, we configured three separate *nutella runs* (one for each class) within the Wallcology *nutella application*. During enactment, we then leveraged the login phase provided by RoomCast to setup the correct context for each session. Figure 28 shows the runs for Wallcology: ‘default’ is a run used by the researchers for testing purposes, the three classes in Chicago (whose real name has been hidden here) are those on which we performed the pilot study, the remaining two classes were instead running a

Wallcology unit enacted by colleagues from the University of Toronto. Even if our study and evaluation do not include results from the unit in Toronto, we will present a few significant cases in which RoomCast turned out to be particularly useful to setup configurations very different from the ones needed on our side.

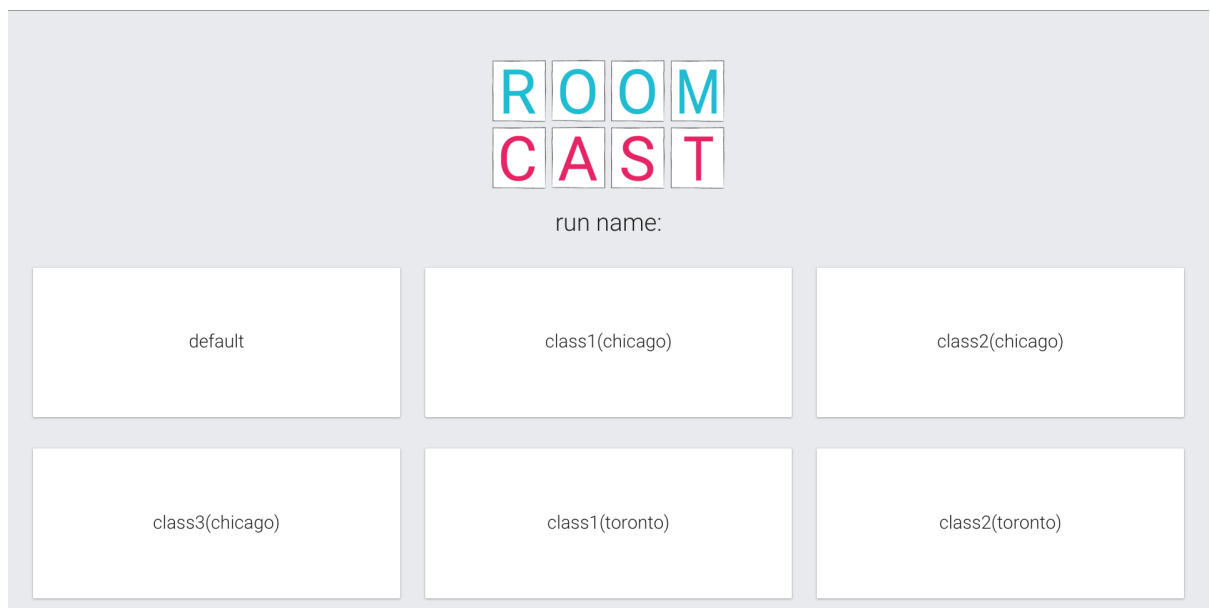


Figure 28: Wallcology: RoomCast login screen.

4.2 Method

In Section 1.3 we stated our main research question for this work as “Can RoomCast serve as an effective technology for classroom orchestration of an ecology of screen-based software resources?”. In order to investigate and answer this question, we set up an exploratory study in

the context of the Wallcology unit. We use the term “**pilot study**” to describe this experience with RoomCast because it was the very first time that this new orchestration technology was deployed and tested in a real classroom setting. For this reason, since we are still in the **early stages** of the development of the system, our main goal for this part of the overall work is to **provide evidence that RoomCast has value** with respect to the support for orchestration and, more in general, to the contributions that this work aims to bring. In this sense, we will talk about ‘*evaluation*’ of RoomCast with reference to the specific meaning of performing an exploration of this novel system.

In order to achieve this, we based our approach on a **qualitative research study** (102; 103), a methodology commonly used in works dedicated to evaluating technologies for orchestration (32; 86; 92). As Prieto points out, “One of the main characteristics of qualitative research studies is its focus, not on the statistical significance and generalizability of the data collected, but rather on maximizing the understanding of the complex factors and relationships that shape the case under study” (32). This is particularly relevant for preliminary studies (or pilots) where we want to **explore how a new tool is perceived and used by the users**. As an example, a similar approach was followed by Bakker et al. (86), who used a long-term qualitative study to explore the use of an innovative classroom technology, “Fireflies”, which presented an open-ended design with many possible and different uses allowed to the teachers.

More specifically, as many works have shown (51; 89; 92), when studying the deployment of technologies for orchestration there’s the “need for evaluations that address orchestration’s

multiple facets” (89). For our study, first of all we decided to carry out the evaluation of RoomCast by following two main and complementary perspectives:

- **Workshops with the teachers:** these sessions took place once a week during the whole period of the unit and they were meant to assess the **teacher apprenticeship** with respect to the orchestration technology. We wanted to study if and how the teachers involved in using RoomCast to orchestrate the Wallcology unit in their classes were able to learn and become comfortable with the technology. To do this, they started from a situation in which they had never dealt with RoomCast or similar technologies before and they had to tackle a learning curve while, in parallel, starting to actually use the system during the enactment of the unit.

At this point, the reader might wonder why we chose to have private weekly sessions specifically with the teachers. The reason is strictly related to the notion of **teacher-centrism**: while describing the theoretical foundations of the concept of orchestration we talked about the view of the teacher as the conductor and guide of the orchestrated learning scenario. If this is true, to reach a proper understanding of the effectiveness of our orchestration technology we need to give particular attention to the role of the teachers by studying two major aspects:

- The **evolution of their appropriation of the technology**: as Dimitriadis et al. observe, “we can see a new trend raising, which we believe lies in the heart of the use of the ‘orchestration’ term: that of how teachers appropriate and integrate in their practice the different technologies at their disposal (either digital or paper-

based, generic or intended for orchestration)” (104). This view of the problem of orchestration as “integration of technologies in the classroom” leads us to consider the teacher apprenticeship as fundamental to study their understanding and proficiency first of all with the orchestration tool and then with all the orchestrable technologies in the scenario, in our case represented by applications (channels) that were quite simple and intuitive. This is about solving the core aspect of orchestration, that is improving the management of the “classroom as a complex ecosystem” (105).

This appropriation by the teachers was tested in terms of understanding of the system and usability studies during the workshops, where they were asked to design the plan for the following classes by making critical choices about the flow of activities and distribution of resources to the students. Together with this, also their performances when using RoomCast during enactment were analyzed.

- The **continuous feedback** that they were able to provide over the period of enactment of the unit: during every workshop, after a week of enactment of the unit from the previous workshop, we dedicated some time to discuss with the teachers their performances using RoomCast during the class period. The data gathered from this sessions will be critical when evaluating the results of the study for what concerns both considerations about the current implementation and vision of the system and also possible future work to improve it.

The use of workshops as the modality to assess the aspects mentioned above had already been used in previous works, such as by Prieto et al. in (32), where teachers were presented

with new orchestration technologies that they had never used before. More in general, many studies have analyzed the teachers' appropriation of technology: as an example, Angeli analytically studied the increasing technology competency of preservice teachers, starting from the problem that "as the student/computer ratio in schools is getting smaller and smaller, more concerns are raised about the preparedness of teachers to appropriately integrate technology in teaching and learning" (106). An approach for teacher apprenticeship similar to the one we employed for our pilot study was presented by Glazier et al., who introduced the concept of **Collaborative Apprenticeship**, arguing that "effective technology integration requires teachers to obtain learning experiences within the context of their teaching so they can practice, reflect, and modify their practices" (107). To achieve this, the method provides that "teachers receive on-site support, in-time training, and continuing training" (107) by both "peer-teachers" and "teachers-leaders"; in a similar fashion we used this approach by having us, the researchers, giving support and training to the teachers throughout the entire duration of the unit.

- **Classroom enactment:** this was the actual deployment of RoomCast to orchestrate the whole multi-week instructional unit for the three classes that participated in this learning experience. During enactment, each teacher had an iPad with the RoomCast mobile application configured to access the RoomCast teacher controls; in addition, a laptop with the RoomCast package creator for runtime management of emergent occurrences was available. Most of the times, and especially during the first weeks of enactment,

a group of researchers was present to support the teachers with any technical difficulty including support to leverage RoomCast in the best way.

Let us now describe more in detail the evaluation process that was set up and performed for this study. A graphical representation of this process is shown in Figure 30. If we first consider the evidence that was collected to perform the evaluation, we can notice from the schema that **several data gathering techniques** were used:

- **Observation** (OBS): observations were made by collecting video recordings with cameras and notes; this data was gathered during both workshops and enactment.
- **Interview** (INT): qualitative and **semi-structured** interviews that were recorded and transcribed. These were held during weekly workshops, either individually or with the 3 teachers together, and also during the final individual feedback session.
- **Screen recording** (SCREEN): recording taken during the weekly workshop with the teachers to keep detailed track of their interactions with the user interface while deploying the design into RoomCast.
- **Log of user interactions** (LOG): this log was created by RoomCast log bot (see Section 3.5.4 during enactment to keep track of significant user interactions made by both teachers and students.
- **Questionnaire** (QUEST): qualitative and exploratory web-based questionnaire to be completed individually by each teacher at the end of the RoomCast evaluation phase. The questionnaire, which includes twelve questions related to the different aspects identified

by the “5+3 aspects [rk]” by Prieto, can be viewed in Appendix B. After each question, the teacher was also invited to insert an additional note (text) to support or precise her answer.

The use of multiple heterogeneous data sources and techniques is particularly important in qualitative research approaches like the one we are using, since we want to increase the value of the findings by supporting them with evidence coming from different point of views and origins. In addition, we also leveraged more **quantitative data** coming from the logs of user interactions recorded by RoomCast.

If now we consider the main steps of the evaluation process we have:

- **Initial workshop:** this first meeting with the teachers represented a training session to show them for the first time RoomCast, make them understand the basic ideas and concepts behind the system and to let them try very simple interactions with the interface. In order to introduce the basic mechanisms of assignment of channels and creation of packages and activities, we first provided the teachers with a paper version of a simple RoomCast configuration (Figure 29). After having guided them through this, we were able to show them how the same procedure could be implemented in the authoring environment and finally to ask them to apply some changes by testing the interface themselves.
- **Weekly workshops:** these workshops, either individual or with the 3 teachers together, were divided into two main phases. In the first, the teachers discussed the experience with Wallcology and RoomCast during the previous week, providing feedback on the system,

Activity 1	
Package name	Channels
student	Ecosystem
teacher	RoomCast

Activity 2		
Package name	Channels	
student	Ecosystem	History
teacher	RoomCast	History

Activity 3			
Package name	Channels		
student	Ecosystem	History	Screen controls
teacher	RoomCast	History	Pressures

Figure 29: First example of simple RoomCast configuration used in the training session.

and then decided the instructional design for the coming week of enactment. This design was then deployed in RoomCast during the second phase of the workshop, where we assisted them and gathered data about their interactions and observations about the use of the tool.

- **Enactment:** this was the actual execution of Wallcology, supported by RoomCast, inside the classroom. It consisted of 2 weekly sessions for each of the three classes over a period of 2 months.
- **Final feedback session:** this session was held during the final phases of the enactment of Wallcology, in order to collect observations and feedback from the teachers about the overall use of RoomCast throughout the unit.

Figure 30 shows the specific type of data that was gathered in each phase of the evaluation process.

4.3 Wallcology unit: design and enactment

Before providing the results obtained from the evaluation process, it is useful to show how the Wallcology unit was structured (or designed) by the teachers together with the researchers. This plan, discussed during the weekly meetings and then deployed during enactment, roughly followed the following steps (which we called *days*) from the point of view of the use of RoomCast:

- Day 1: the four public displays are only allowed to show to the students the ecosystem that they, as wallscopes, are supposed to monitor. This means that there is only one channel

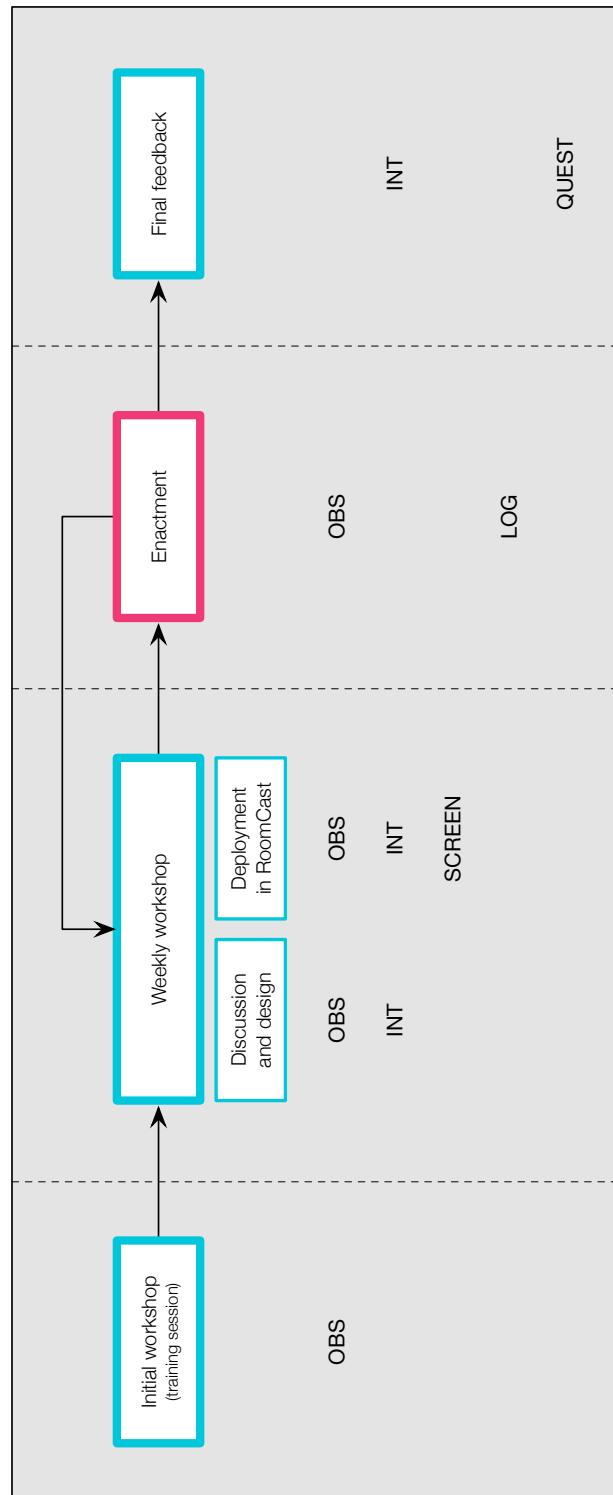


Figure 30: Evaluation process and data gathering techniques for the pilot study.

available and the students in this phase don't know yet about the role of RoomCast: the groups are required to study their habitat e.g. by building food webs.

- Day 2: a new tool, the 'History' channel, is added to the lineup on each display. Now the groups can leverage the help of graphs that show the historical growth of the populations of organisms.
- Day 3: once the groups have reached a sufficient understanding of the dynamics affecting the populations inside their ecosystems, it is time to give them access to the 'Screen Controls' channel. The students are now allowed to make modifications on their habitats. The number of available actions inside the channel is increased over time by modifying the RoomCast card associated to the channel. As an example, Figure 31 shows the activities available from the teacher controls at an early stage of use of the controls: 'Activity 3' gives access to the 'Controls' channel, but this interface had been modified to allow only actions of increase or decrease of one species of organisms. Then, in the following classes and when the groups proved to have reached a good level of understanding of the dynamics and effects of these controls, they were given also the insert/remove operations on their ecosystems.
- Day 4: when the students are comfortable with the use of the web controls, we can introduce the 'Tangible Controls' channel, which has the same capabilities but lets them apply the changes by just putting physical objects (access tokens) in the proximity of a display. Figure 32 shows the configuration of an activity to implement this design.

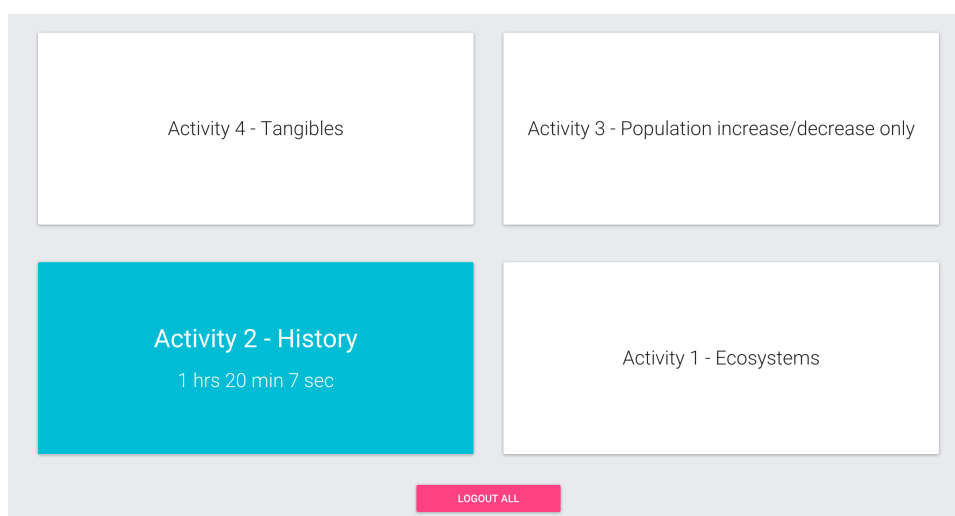


Figure 31: Wallcology design: RoomCast teacher controls.

In the next section, while providing the results of the pilot study we will also describe how the transitions between days took place, including the unpredictable and unexpected occurrences that required adaptations of the design at runtime.

4.4 Results

In this section, we will present the results obtained from the pilot study. Before delving into the analysis of the support for orchestration, we want to highlight that RoomCast fully satisfied the critical non-functional requirement of **reliability**: the system remained stable for the whole duration of the Wallcology unit, that is 45 days of enactment over a period of 8 weeks. We didn't experience any failures, which is particularly important for a tool deployed in a classroom environment. We were able to apply changes to the configurations of RoomCast even manually by moving data from one class to another (e.g. from one in Chicago to one

Activity:
Activity 4 - Tangibles

Public
+
-

Ecosystem 1
Ecosystem 1
Tangible controls 1
History
+

Ecosystem 2
Ecosystem 2
Tangible controls 2
History
+

Ecosystem 3
Ecosystem 3
Tangible controls 3
History
+

Ecosystem 4
Ecosystem 4
Tangible controls 4
History
-

Add Row

Personal
+
-

Teacher
History
RoomCast
+

Add Row

Figure 32: Wallcology design: configuration of ‘Activity 4 - Tangibles’ in RoomCast package creator.

in Toronto), keeping the system working without downtimes and in a transparent way for the users. We can also say that the checks made by RoomCast both at back-end and front-end level for bad or unexpected inputs have worked properly to prevent errors from manifesting to the users.

4.4.1 Orchestration analysis

For analyzing the results of the study in order to answer our research question, it looks convenient to evaluate RoomCast, as an orchestration technology (that is from the point of view of **classroom usability**), by looking separately at the evidence related to the different facets identified by the ‘**5+3 aspects**’ framework by Prieto. In particular, this evidence will be represented by the different data that we collected by means of different techniques and we will see, in practice from our experience, the importance of having a data gathering process able to find supporting evidence for all of orchestration’s multiple facets: for some aspects of orchestration one type of data might be much more relevant and informative with respect to others.

This approach for studying the results of the evaluation process is useful first of all because we have already provided a detailed orchestration analysis of RoomCast in Section 3.6 following the same framework, thus we will be able to easily make comparisons and draw conclusions by looking at the data that we gathered from the pilot study. Moreover, this type of analysis has already been used in a number of works in this field (51; 89; 92): Muñoz-Cristóbal et al., as an example, recently evaluated the support for orchestration of GLUEPS-AR, the system we described in Section 2.2.6; to do this, they identified some ‘topics’ for which they were able to

gather data by means of multiple different techniques and then justified their approach stating “since such topics correspond to the different aspects of the ‘5+3’ orchestration framework, they enable us to explore how GLUEPS-AR provides orchestration support” (92).

Finally, the same structure was also used to structure the questions for the final questionnaire for the teachers.

Let us look at the evidence that we were able to find for each aspect of the framework; the reader can find, for each result, references to the specific data source(s) that can be used to support that statement.

Design/planning

- The evaluation showed that RoomCast enabled the teachers to deploy the learning design for Wallcology: they were able, in each weekly workshop, to first think and discuss about the design for the successive week of enactment and then to deploy this plan into RoomCast using the package creator interface [OBS, INT, SCREEN]. There was a certain learning curve to fully appropriate this workflow, which will be presented in the section dedicated to the teachers apprenticeship.
- Regarding the package creator, overall the teachers agreed that “the interface and interactions required look ‘quite intuitive’ ” [INT] and that “the authoring environment looks simple enough, it’s a matter of practice using it” [final INT].
- All the teachers agreed that the operation of deployment of the instructional design into RoomCast (creation of channel packages and activities) is easy to perform, after an appropriate initial training [QUEST]. Two of them pointed out that their learning process

had been slowed down, with some initial confusion, by their busy schedule and number of classes taught, but that they “can see this being very facile in years to come” [QUEST], that is when they will use RoomCast to support future educational units similar to Wallcology.

- They strongly agreed that the system has allowed them, by selectively assigning resources (channels) to the different displays in the classroom, to plan a design according to the pedagogical approaches that they want to use [QUEST]. They also agreed on the fact that RoomCast has never forced them to organize student work in a different way than what they are used to [QUEST].
- During the first workshop, one of the teachers, after having deployed a design for one of the classes, asked: “Did I do this configuration for all the classes or I need to do it for the others?” [OBS]. This highlighted an interesting point, that is the need for **reusable configurations** across multiple classes. If in the current implementation the duplication of configurations had to be manually done by the researchers by accessing the raw data structures on the server, future work will address this need by adding this capability to the user interface, making it available to the teachers.
- During the final interview, one of the teachers looked confused when asked about the mapping of an ‘Ecosystem1’ channel assigned to a package with name ‘Ecosystem1’. In this case, we had decided during the very first workshop to use package names corresponding to the four different ecosystems, since each public display in the classroom had to keep that identity throughout the whole unit. The teacher, however, after weeks of enactment using that fixed role for the displays showed some confusion about the notion of ‘package’.

This leads us to think that this problem was due to the early experience with the tool, which probably would be clarified in future units that will leverage RoomCast and its capabilities in different ways; at the same time, however, this confirms that the concept of ‘mapping resources (channels) on virtual identities (packages)’ is the most challenging to understand and it requires time and practice to be fully appropriated.

Regulation/management

- **Resources management:** all the teachers strongly agreed that the idea of RoomCast of selectively distributing channels to specific displays has effectively helped them regulate and manage their classes [QUEST]. This included regulating the available resources for each activity according to a plan, but also the runtime modification (insertion/removal) of channels based on unpredictable occurrences (more on this in the adaptation aspect). As we will see, this was helped by the use of the metaphor of RoomCast as a content provider, both semantically and with the visual help of the interfaces.
- **Workflow management:** the sequence of activities was discussed, designed and deployed into RoomCast during the weekly workshops; then, during enactment, the teachers launched the planned activities from their personal RoomCast controls. Since this was the first experience of the teachers with the tool, it was particularly useful to meet them weekly and guide them towards the identification of the critical activities that had to be set up. Moreover, we found that these regular workshops were useful for them to revise the plans from the previous days based on the achievements and level of understanding of the different classes: RoomCast allowed them to **iterate on the design** of the flow of

activities, for example by incrementally adding new activities containing more resources as the unit progressed. During this process, they all showed a good understanding of the idea of ‘staging activities/configurations’ and became increasingly autonomous while doing it. [OBS, INT]

- **Group management:** this aspect was tested during our pilot only with respect to a fixed configuration of groups that included four packages, one for each public display corresponding to the four wallscopes, and an additional package of channels delivered exclusively to the teacher. Even if this worked effectively as expected [OBS], it represented only a basic usage of RoomCast to manage the same group formation over the whole period of the unit: in the future we will need to further study the different possibilities of regrouping allowed by the system with applications that will have pedagogical structures more complicated than Wallcology’s one.
- **Time management:** the time variable was successfully implicitly managed by the teachers when switching between successive activities [OBS]. Considerations about the information provided about the time elapsed for an activity will be provided among the results for the awareness aspect.
- **Management of instructional units over multiple classes and login phase:** the login phase provided by RoomCast was effectively used by the teachers when setting up the public displays and also their personal devices [OBS]. When asked about this feature, they agreed that the procedure is ‘simple and quick enough, without asking for complicated and time-consuming password-based authentication steps’ [final INT].

One of the unexpected occurrences that we noticed was that in a few situations students had to reload the RoomCast app on a public display because the channel they were using had stopped working; once presented with the login screen (and without anyone helping them) they were able to log in correctly and to get back to their previous activity [OBS]. In previous units where we were not using orchestration systems such as RoomCast, these students would have had to ask for help from the teacher to restore a lost session, including accessing an explicit URL with possibly additional parameters: RoomCast solved this problem, **making any technical detail completely transparent** to the users.

Adaptation/flexibility/intervention

- When asked if the operation of launch of new activities from the iPad application (teacher controls) for the teacher during the class period is sufficiently fast and simple to be effectively used at any time when needed, the teachers strongly agreed about the effectiveness of this way to apply changes at runtime [QUEST, INT].
- On the other hand, when asked if the assignment of channels that are available to the students, through the **package creator**, is sufficiently fast and simple to be effectively used at anytime **during enactment** (e.g. when we needed to assign a new tool just to one group of kids) the teachers expressed some doubts. They did think that this capability is very useful (“It was really helpful to be able to differentiate for various groups and initiate one group’s change if they were ready, even if another group wasn’t” [QUEST]), but they had to admit that to leverage it at runtime they had to ask for additional help from the researchers.

Indeed, to change an assignment of channel the current implementation of the interface requires 4 clicks, which is a burden, especially if the operation is done at runtime by a teacher who is not yet so confident with the system. To solve this issue, we will need to **iterate again** on the design of the interface.

One of the teachers suggested that a solution for this could be to add some controls, equivalent to those in the package creator, directly in the RoomCast app running on the screens in the classroom: this would possibly make it easier to go there and enable the needed channels. This is certainly an interesting perspective that might be explored as future work even if it comes with some drawbacks, such as the need for authentication of the teacher on the screens before showing the controls. [QUEST, INT]

- Concerning the adaptation at runtime, one of the teachers noticed that ‘in a **perfect educational setting** we want to give to groups that advance quicker in their activities the tools (channels) to move on. This is what RoomCast allows to do with the package creator at runtime, but that requires some practice. For this reason the teacher added that “in possible non-perfect scenarios, we could say ‘we will move on to Activity 4 only when everyone is ready’, that is if a group finishes before time it can still do some additional work with the channels it already has or use notebooks or other non-ICT tools, but we enforce that everybody moves on when the teacher enables this from the RoomCast teacher controls. This represents an interesting **simplification of the orchestration** needed at runtime which **reduces the orchestration load for the teacher** to the use of just one, simple interface, but at the cost of reducing the effectiveness of the educational

method. This might still represent a good choice for teachers that cannot afford the flexibility allowed by a more complex orchestration level and it is a solution that is perfectly supported by RoomCast. [final INT]

On the other hand, another teacher explicitly confirmed that for this unit most of the usefulness of the system came from the possibility of ‘managing groups that were out of sync with each other’ with respect to the activities/channels they were allowed to work on [QUEST].

- The teachers agreed that overall RoomCast was useful to adapt to emergent occurrences, such as when a group was late with the work and had to wait before getting access to a new tool. [QUEST]
- The need of dealing with emergent occurrences at runtime was clearly evident also from the observation during the enactment of the unit. As an example, during an early session a teacher asked for help from the researchers saying: “Only those two groups are ready to get the controls [they had agreed on which action they wanted to perform on their ecosystem and explained evidence and motivation in their notes], can we give the new channel only to them?”. This request expressed a clear need of the teacher to selectively advance the unit for a subset of students. The teacher, however, in this case was not yet comfortable with making the changes in RoomCast by herself to deal with this **improvisation**. Moreover, in successive sessions, even if she would have been able to do it herself, help from the researchers was again needed because of the extremely limited amount of time that the teacher had to apply these orchestration changes while having also to contemporarily

support other groups of students.

These observations highlight once again that further efforts will be needed to keep **minimizing the cognitive/orchestration load for the teacher** at runtime, while still providing these useful functionalities. [OBS]

- In a similar way, not only the insertion of a new channel, but also the **removal of a previously assigned channel** was sometimes needed at runtime. As an example, when we first introduced the ‘Controls’ channel to manipulate an ecosystem, after a group had performed the desired action the teacher had the need of preventing the students from using that affordance again. This was solved by means of the package creator, but again further efforts will have to be put for improving the **usability at runtime** for the teacher, which resulted to be much more constrained by the real setting with respect to the ‘offline’ use during the design phase.

Awareness/assessment

- While the teachers agreed that in general RoomCast, during enactment, can potentially help increase their awareness of the status of the class (e.g. availability of tools (channels) and tasks, group formation, time progress of the activity), they provided also some interesting remarks. They all noticed that the usefulness of this kind of capability is **strongly related to the way in which a specific educational unit is enacted**: if the unit doesn’t require the teacher to go around the class to monitor what the students are actually doing and interact with them, then information on the time elapsed for an activity and current distribution of channels could be useful. However, this is often not the case

for participatory simulations of this kind: as one of the teachers observed, “We cannot let them [students] work and stay behind a computer to check what is going on...”. [QUEST, INT]

In this sense, all the teachers agreed that during the unit their primary way to be aware of the status of the advancement of the class was by directly interacting with the students to assess their understanding.

- On the other hand, when asked about the possible use of channels dedicated to improving the “**private awareness of the teacher**”, they found it potentially very useful: some of them expressed the need to have, for example, a channel which provides an easy to read summary of the notes written by the students on their tablets (e.g. in a spreadsheet-like format), an operation that during this unit instead had to be done after class by reading all the notes from a graphical user interface which slowed down the assessment phase. In this sense, another suggested channel would provide some simple **statistics** about the use of Wallcology and of RoomCast itself (e.g. frequency of switching of channels for each group, which would be a good indicator of the effort that the teams are putting during enactment). As one of the teachers suggested, “It would be useful, for example, to know how many times the students have done certain operations”.

These requests highlighted that the needs of the teachers during the enactment phase go towards tools and information that are **specific to the context of each different instructional unit** (i.e. application, such as Wallcology). Thus, general purpose utilities such as the timing of activities are not as useful as having **personalized private channels**

for the teachers: this capability is well supported by RoomCast, but it requires to develop ad-hoc channels based on the teachers' feedback and requests.

Roles of the teacher and other actors

- In our study, the teachers have always been in charge of the orchestration process and the importance of their role has been highlighted by the attention dedicated to their progress and apprenticeship throughout the dedicated workshops. In this sense, after this first experience we can argue that the **role of the teachers** and their understanding and evaluation of the orchestration system is critical for the final outcome of the instructional unit. The teachers' feedback on the advantages and drawbacks of the system is the most reliable source for future improvements, since most of these have to take into account several constraints imposed by the real setting and whose perception would be extremely hard and altered if they were not coming from those who are in charge of the orchestration itself.
- With respect to the possible tuning of the level of guidance of the class, it was interesting to observe the differences along the evolution of the Wallcology unit: if at the beginning the students were just allowed to watch and study their ecosystems (only one channel per screen), later on they were given increasingly more tools to finally reach a situation in which each group was allowed to use all the channels implemented for the unit. Looking at this process, we were able to appreciate an interesting evolution with respect to the **role of the students**: at the beginning, without even being aware of the existence of RoomCast, they used a single application, just like they would have done in any other

non-orchestrated scenario; in contrast, in later phases of the unit when they were given more **choice** and freedom, an **increasing debate** internal to each group arose, where kids had different opinions about which tools should have been used to accomplish a certain task. We frequently assisted at discussions in which one student said to the other group members: “To do this, we should use the History channel, go to the History!” [OBS]. From this data, we can argue that from the point of view of the students, RoomCast looks to be perceived as a “**toolbox**” that contains useful instruments to support their investigation. As we will see, this aspect will have also consequences from the point of view of the students’ own conceptual model of the experience with the unit leveraging RoomCast.

Pragmatism/practice

- This aspect was evaluated through the **dedicated sessions and workshops with the teachers**, where we studied their process of apprenticeship of the technology, but at the same time gathered feedback and discussed their practical experience with the system during enactment. This was useful to understand to what extent the variety of pragmatic constraints that might influence the ability of the teacher to coordinate the scenario were actually satisfyingly managed by RoomCast and, on the other hand, what were the drawbacks or lacks that should be addressed in future iterations. Considerations about these topics are reported throughout this section, as part of the other aspects of the framework, and also in the results from the workshops that will also be provided.

- A further side note about the pragmatic use of RoomCast involves the successful deployment of the system to orchestrate a Wallcology unit that was enacted, in parallel to our Chicago pilot, in a school in Canada by the learning technologies group at the University of Toronto. In this context, RoomCast proved to be leveraged with easiness and flexibility: even if the design of the activities and resources was very similar to the one from our pilot, several times they needed small variations of configurations or contents of the channels. To manage this, RoomCast allowed them to start from our design and apply just the small changes to fit their requirements. In other occasions, **unexpected uses** of RoomCast were required to deploy new types of designs: Figure 33 and Figure 34 shows two alternative configurations that were deployed in Toronto, based on the need of having, at runtime, either distributed or centralized controls for the ecosystems.

Alignment/synergy

- As we saw in the orchestration analysis of RoomCast, the main way to integrate scaffoldings in a synergistic way relies on the distributions of channels to the screens across the classroom. In this sense, it was interesting to study, during enactment, the actual use of the resources by the students. Since the observations with the camera can cover only partially what is happening at runtime, this is one of the cases in which we could rely on RoomCast's **logging system**: this provided a log of all the user interactions to be filtered with queries and analyzed after enactment. As an example, a simple query like the following:

```
db.getCollection('6TV/roomcast-log').find({"time.month":11, "time.day":3,
```

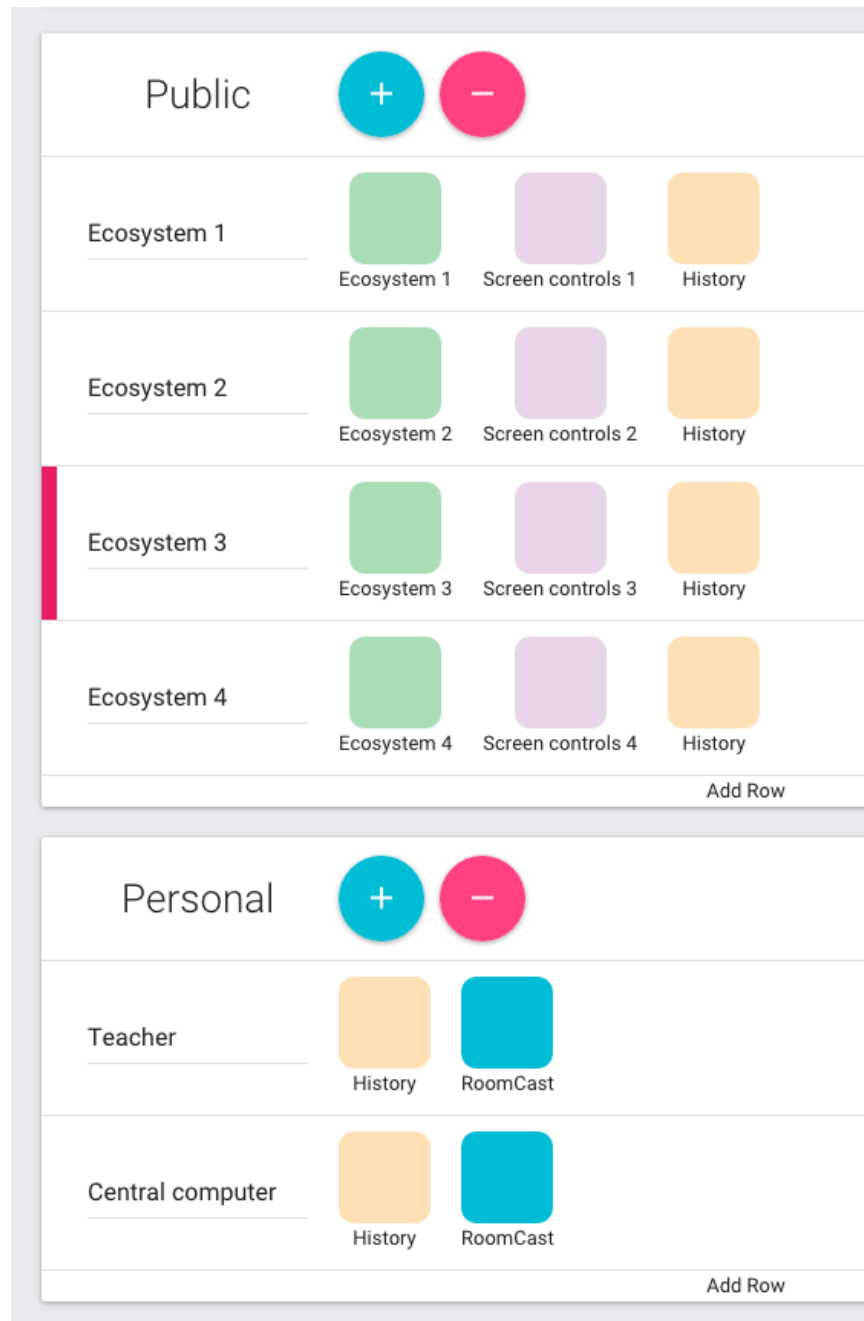



Figure 33: RoomCast design in Toronto: activity with distributed controls.

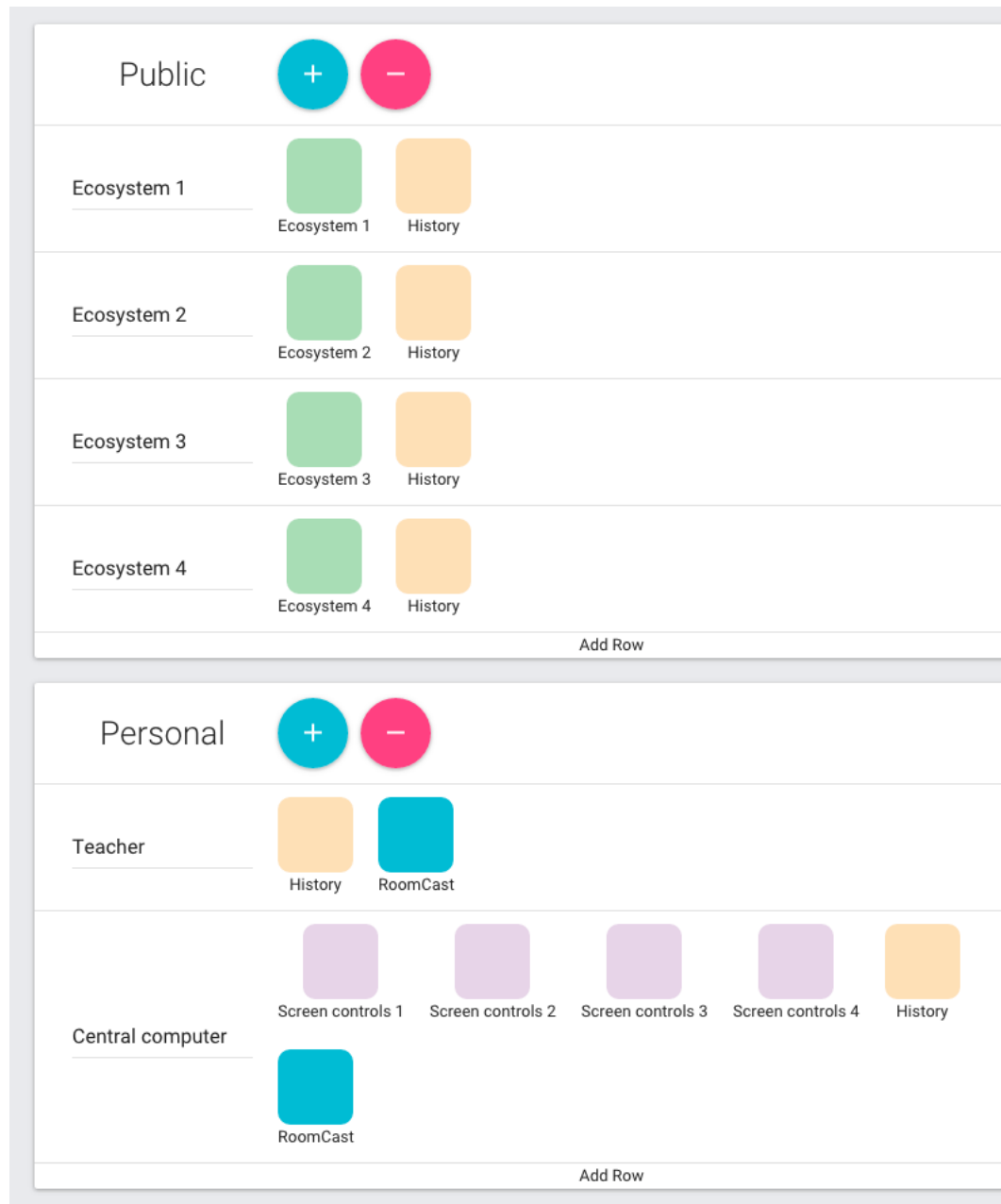


Figure 34: RoomCast design in Toronto: activity with centralized controls.

```
"device_id":"6elgnysf.o53"}).sort({"time.timestamp":1})
```

permitted to filter the log data to obtain the interactions that occurred on a specific day, on a specific device.

Leveraging queries like this to find evidence about the synergistic use of different resources, we discovered that the frequency of launch of a channel, together with that of switching between different channels, is significant to understand how a group is progressing throughout an activity. An example for this occurred when we analyzed the log data for a group that had not performed well, being the only one not to be able to come up with a shared decision on which manipulation to apply to their ecosystem: when we looked at their interactions, we found out that they had not switched channels and they had not used as a support for their decision the History channel at all (in contrast with the groups that obtained the best results, which had used all the tools at their disposal and switched between them very frequently, for a total of 8 channels requested, to make analyses and comparisons). Thus, we can argue that there is a **correlation between the use of different resources (channels) and the success in completing the assigned tasks**. In other words, the groups that made progress faster were the ones that had been able to leverage the **synergies between tools provided by a wise use of the orchestration technology**. In this sense, the teachers could also assess the students' ability to manage the margin of **self-regulation** that was increasingly left to them. [LOG, OBS]

Models/theories

- With respect to the use of the **metaphor of the television provider** that distributes channels, all the teachers strongly agreed that it was effective for themselves to understand how the system works and then to be able to use it. At the same time, they also thought that the metaphor was useful and simple to understand for the students; one of them pointed out that the students are particularly familiar with “on-demand services” such as YouTube and Netflix, based on the model of providing a set of available channels-resources, that is something very similar to what the students experienced with RoomCast. [QUEST, INT]
- During enactment, when a teacher first presented RoomCast to the students she said: “Imagine now that you can choose the channel, just like when you pick channels on the TV, you can choose to watch channels... and they are actually tools!” and later on she added: “The new tool that I am going to give you today is the History channel!” [OBS]. This introduction of RoomCast, by strongly relying on the metaphor, proved to be very effective for the quick understanding of what RoomCast is about. Then, when the teacher added: “Now, if you look up in the top-left corner of the screen, and you push that little pink arrow, it shows you the available channels that you can select. You can see that now you have also the History tool!” [OBS], the students showed to be able to start using the system and switching between channels with great confidence from the very first moment [OBS].

Additional aspects

- All the teachers confirmed that they noticed that RoomCast helped **increase the motivation of the students** during specific phases, such as when they were told that a new tool (channel) was coming [QUEST, INT].

It is interesting to notice that this aspect was foreseen with a good intuition by a teacher on the very first workshop: “I am thinking that now that we have this option for differentiation, there are maybe some groups that only at the very end will play with the controls themselves, they will first need to show mastery with simpler tools: that will become an **intrinsic motivator** since kids will say: ‘they [the other group] completed that level, that’s why they’re on that new level!’ ” [INT].

The same teacher, during the final interview, added that “if it was too hard for the students to get a new channel, they would give up, just like in a **videogame**”. But for this unit she confirmed that “there was a right balance to get to the ‘next level’ ”. Another teacher underlined that using the term “**new tool**” while presenting a new channel to the students is a “very effective expression, it also conveys a sense of **added responsibility**” [final INT].

- If we look at the same **motivational aspects** from the point of view of the students, the results are in accordance with what perceived by the teachers: the video recordings show that the students looked very excited every time they were told that a new channel was about to be added. This happened, for example, when the teachers presented for the first time the introduction of the tangible controls (3D-printed reproductions of the species

inside the ecosystems): in the first group that was able to finish the previous task and get access to the ‘Controls’ channel, a student exclaimed: “Look, look, we got the new tool!” just before switching to that channel and starting using it. [OBS]

- If we consider the evidence that we got and the feedback from the teachers, it is clear that the need for **minimalism** expressed in the literature by Dillenbourg is critical to reducing the ‘global orchestration load’ that they have to deal with. There is, indeed, a **tradeoff** between the effort required from the teachers for orchestrating the scenario (thus potentially resulting in more instructional effectiveness) and the need for minimalism and high usability, especially at runtime, of the orchestration system [INT, OBS].

This tradeoff was clearly visible when comparing the ease of use that the teachers experienced with the RoomCast teacher controls with respect to the higher effort required by the RoomCast package creator during enactment.

Finally, let us look at some interesting **statistics** derived from the data stored by the RoomCast logger of user interactions:

- Number of weeks of enactment: 8
- Number of 6th-grade classes: 3 in Chicago + 2 in Toronto (not considered in the next statistics)
- Number of class meetings: 45
- Number of activities launched by the teachers: 62 (45 at the beginning of class + 17 during class)

- Number of changes to a running activity during enactment: 53
- Number of channels requested: 429
- Average number of channels requested for each class meeting: 10
- Largest difference of number of channels requested between two groups (public displays) in the same activity: 2 vs 8.

It is interesting to notice that the number of interventions of the teachers at runtime (53, often with support from the researchers) is significantly higher than the number of activities launched by the teachers excluded the ones at the very beginning of each class (17) to let all the groups advance with the same pace. This confirms the need to maximize the support for runtime adaptations, but keeping a low orchestration load for the teacher.

4.4.2 Teacher apprenticeship

In this section, we want to specify additional results obtained during the workshops with the teachers and specifically with respect to their process of apprenticeship of the technology. As we have already underlined, given the central role of the teachers, studying their **appropriation** of the orchestration system is critical to understand how the technology is practically perceived and used in a real setting.

In particular, from the first training session we were able to assess their **quick appropriation of some key concepts** such as the meaning of the cable television metaphor and the idea of having a sort of parental control to tune the distribution of channels. In this sense, after the first presentation of the example of simple configuration on paper one of the teachers said:

“Looks good, it makes sense to me on paper, it’s just a matter of seeing how it looks like on the real system”, that is showing understanding and willingness to start trying RoomCast. When asked to start trying to interact with RoomCast, the first doubts arose, with questions such as: “How do you set up the distinction between students and teachers, say for activity 1?” or “For the ecosystem, do we have to take care of designating which wall is which computer?”. These were important questions for most of which explanations had not been provided yet, thus showing a good level of understanding of the general ideas behind the orchestration system, together with **desire and curiosity** about going more in detail and understanding all the mechanisms and possibilities offered by the system. This second step, that is the appropriation of how to use the system and to become familiar with the user interfaces, overall required, as it is natural, a **longer learning curve**, including the actual practice during the enactment of Wallcology.

During the following weekly workshops we experienced a **growing confidence** of the teachers with the use of the authoring part of the tool, even if with some differences between the three teachers in terms of time needed to become familiar with the tool: as an example, on one of the early workshops one teacher expressed concerns about the possibility of forgetting some aspects or mechanisms of RoomCast before the successive session; on the other hand, during the same workshop, another teacher seemed much more confident and she even made us notice that in one configuration of RoomCast the package dedicated to the teacher was missing and she suggested to add it from the package creator. These **differences of learning curves** are common and normal, so we will need to pay particular attention to the results and feedback

from the last and more advanced sessions, when the appropriation of the system should be at a good level for all the teachers.

While the description of the improvement of the teachers' appropriation of RoomCast during enactment has already been presented while describing the results from the viewpoint of the orchestration analysis, we want to introduce here an additional significant remark: what we increasingly experienced throughout the enactment of the unit was that there were **specific recurrent operations and design patterns** that the teachers, with the support of the researchers, decided to use. Among these “**routines**” we can find, as an example, an activity in which one or more channels are removed from all the devices, useful to manage periods in which the teacher wants to signal that the students have to stop doing their work and participate to a class-wide discussion (**regulation of attention**). Another example is the common situation in which the teacher, as happened with the deployment of the ‘Controls’ channel in Wallcology, wants to selectively assign a new channel to the groups as soon as they successfully complete a task. These recurrent situations have been called “**orchestration patterns**” in literature: as many researches have noticed, “uses and combinations of tools by teachers in the observed classrooms are highly routinized” (32; 104). Dimitriadis et al., for example, proposed “a method for the elicitation, classification, combination and application of small-scale design and enactment patterns for orchestration” (104). As Prieto suggests regarding the work of identifying these recurrent routines, “the ultimate goal of this research effort is to support, to help teachers in their teaching practice with ICT” (32). In this sense, we can argue that the identification of

patterns of use of RoomCast to solve recurrent problems was a useful way to facilitate, with the help of the researchers, the teachers' appropriation of the technology.

Let us finally consider the feedback that we got from the teachers during the final questionnaire and interviews. All the teachers agreed that “the interfaces of RoomCast look simple enough, it's a matter of practice using it” (final INT). One of them declared that the initial confusion with RoomCast was due to the use of many different terms, sometimes very similar between them, throughout not only RoomCast but all the applications in the Wallcology unit: as an example, having to remember what ‘Activity 2 - Perturbation’ (name of a RoomCast activity chosen during one of the first workshops) was about, while at the same time calling ‘manipulations’ the actions performed on the ecosystem, was leading to doubts when using RoomCast at runtime. This proves that a wise and coherent choice of the terms used to set up the orchestration system, together with proper images and descriptions for the channels, is critical to guarantee the effectiveness of the system when actually using it. They also all think that RoomCast is easy to use for non-technical teachers, after an appropriate initial training, and that their confidence in using RoomCast has increased over time throughout the different workshops for the teachers.

All the teachers appreciated the effectiveness of the visual metaphor of the channel/card, including the use of an image and colored icon for it, and in general all the coherent visual aspect of the tool. A significantly positive feedback was expressed by a teacher in the final questionnaire: ‘RoomCast is a fast and effective system, no matter who is using/administering it. I love the clean UI and the vibrant visuals’.

Finally, regarding the overall experience and future expectations, one of them said: ‘I am excited about the possibility of repeating Wallcology again next year. So much of this year has just been getting our feet wet, wading in and making sense of each new step as we reached it, and making lots of adjustments’. In this sense also for us, researchers, once overcome the initial pilot study, it will be useful to perform a full evaluation of RoomCast during the enactment of future units.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

To conclude this work, it is useful to provide a summary of the contributions that have been presented.

From the pilot study of RoomCast in the context of the multi-week enactment of the Wall-cology unit, we have provided evidence to support four main characteristics of RoomCast:

- **Reliability:** the system proved to be reliable in terms of continuous support for orchestration over a multi-week period of enactment and meetings with the teachers;
- **Classroom usability:** the system was able to implement many different traits of classroom orchestration, as described in Section 4.4.1, including additional aspects such as motivation and minimalism;
- **Learner usability:** by means of the orchestration analysis we have provided evidence of the very good reception of the technology by the learners. With the support of additional quantitative data from the logging system, we have shown how the students used the pedagogical resources with the help of RoomCast and the correlations between the patterns of use of the system and their learning progress;
- **Teacher usability:** by means of the analysis of the evidence from both classroom enactment and workshops we have shown the progress of the teachers' apprenticeship of

RoomCast, including both ‘good parts’ and features and interfaces that will require future improvements.

These aspects include both **pedagogical contributions**, expressed within the novel framing for supporting the orchestration of multi-display based classrooms, and **technological contributions**, that is the ones introduced by RoomCast as a system built from scratch to solve a specific problem and with particular attention to the usability of interfaces and user interactions. Furthermore, considering the four characteristics presented by the system, we can answer our research question saying that RoomCast has proved to be able to serve as an effective technology for classroom orchestration of an ecology of screen-based platforms, showing its ability to continuously support a multi-week instructional unit enacted in parallel for different classes with same resources, but different teachers and different pace of advancement of the activities.

5.2 Future work

Once having provided evidence and results for the first experience of deployment of the new technology, we can describe the possible future improvements that were suggested and derived from the pilot study. Future work to improve the usability of RoomCast both at individual and classroom level include:

- **Enhancements for orchestration during enactment:** as we have seen with the pilot study, while the RoomCast teacher controls have shown to be easily used at runtime because of the minimalism of the interface, the use of the package creator for adaptation of the design to unexpected or unpredictable occurrences is much more difficult to be

managed, due to the multiple constraints of the classroom setting. Future work will need to iterate on the design of this interface: on one hand we could think of moving critical affordances to the teacher controls, but this would only decrease the usability of the interface and duplicate the core functionalities that we already have in the package creator. For these reasons, a possible solution would be to revise the RoomCast package creator interface itself by adding the possibility to switch, whenever needed, from the normal behavior (the one described in this thesis, that is the authoring environment) to an “**enactment mode**”, that is a second mode dedicated to enhancing the interface to deal with the need for faster interactions and simpler operations at runtime. This could mean drastically reducing the number of clicks needed to perform an action, such as clicking directly on the button to add a channel and have it directly affect the classroom setting without having to save and confirm the changes. By keeping the very same interface, but requiring 1 click instead of 3 to change the design, we could improve the frequency of operations of the teachers while reducing the cognitive load due to remembering that every time they need to save the changes. This would become optimized to be used, together with the teacher controls, on the personal iPad of the teacher, by just tapping once on the package to which we want to add/remove a channel.

- **Reusable configurations:** as the results of the evaluation suggested, we will need to simplify a common need of the teachers, that is redeploying the same design across multiple classes. To do this, we will give the possibility of using previously saved configurations as **design templates** to be redeployed in the same or another class. This would represent

a powerful tool that leverages the helpful use of **design patterns/routines** as we saw from the pilot study.

- **Identification of individual actors:** in the current implementation, RoomCast supports the creation of packages as ‘virtual identities’ that can be assumed, through the login phase, by any device inside the classroom. While this is what we need for public displays (used by groups of students that change over time), for the case of personal devices it might be useful to have the possibility of logging them in based on the actual identity of their owner, such as a student or teacher. As an example, we would be able to design RoomCast packages (seen as roles) in the usual way, but then also to specify the actors (teachers and students) in the scenario and to map each of them (or his personal devices) to the package/role that he assumes at a specific time. In this sense, from the package creator we would simply design **activities as mappings of people on roles**, e.g. saying that “in Activity1 Tom has to be a predator, while Matt will be a prey”. In other words, we would create an **additional layer of abstraction** to identify the roles of individual actors inside the classrooms. The great advantage of this is that a student would be required to log in with his name just once and from that point the channels assigned to him would be managed by RoomCast considering the role which that specific student has to assume throughout each activity.
- **Integration of new types of channels/resources:** to enrich the instructional units, we could think of adding to RoomCast the support for technologies that will allow to create new kinds of channels, in addition to the current web and iOS-based ones. The modularity

of RoomCast would permit to have the same lineup of cards representing channels, but then to show one each device only the subset of channels supported by it.

- **Integration with RoomPlaces:** RoomPlaces is a service provided, like RoomCast, inside the nutella framework. It allows to track the location of people and devices inside the classroom by means of sensors powered by the iBeacon technology. An integration of RoomPlaces into RoomCast would give to the orchestration system new capabilities that leverage the physical position of the actors inside the setting. As an example, a specific token own by the teacher could give instant access to a channel just by having the teacher reach the proximity of a public display, thus with interesting possibilities in terms of runtime adaptation and flexibility.
- **Agent-based selection of channels:** what if we enhanced RoomCast to act as an agent that is able to decide by itself which channel to play at any point in time? It could, as an example, start the execution of a specific channel based on some rules, such as when the kids in front of a public display are all using the same tool on their iPads and we want the system to give them a channel/resource on the display that complements their activity. This would reduce the need for teacher intervention and would open new interesting scenarios.

APPENDICES

Appendix A

DATA STRUCTURES

The following code represents the JSON Schema (json-schema.org) specification for the two main data structures inside RoomCast:

channels.json

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "http://roomcast.org",
  "type": "object",
  "title": "RoomCast's channels schema.",
  "description": "Data structure for RoomCast's channels catalogue.",
  "name": "/",
  "properties": {
    "channels": {
      "id": "http://roomcast.org/channels",
      "type": "object",
      "description": "Map of channels accessible via their ids.",
      "name": "channels",
      "patternProperties": {
        "^(\\[0-9\\]+)*$": {
          "type": "object",
          "name": "channel",
          "description": "A RoomCast channel.",
          "properties": {
            "name": {
              "type": "string",
              "name": "name",
              "description": "Name of the channel."
            }
          }
        }
      }
    }
  }
}
```

Appendix A (continued)

```

},
"icon": {
  "type": "string",
  "name": "icon",
  "description": "Hex color for the icon representing the channel.",
  "pattern": "^#(?:[0-9a-fA-F]{3}){1,2}$"
},
"screenshot": {
  "type": "string",
  "name": "screenshot",
  "description": "Url to online file or path to local file."
},
"description": {
  "type": "string",
  "name": "description",
  "description": "Short text to describe the content of the channel."
},
"url": {
  "type": "string",
  "name": "url",
  "description": "Pointer to the actual content of the channel: either
                  a URL or a CustomURL to launch mobile apps."
},
"type": {
  "type": "string",
  "name": "type",
  "description": "The type of channel: either 'web' or 'iOS' are
                  currently supported.",
  "pattern": "^(web|iOS)$"
}
},
"required": [
  "name",

```

Appendix A (continued)

```

        "icon",
        "screenshot",
        "description",
        "url",
        "type"
    ]
}
},
"additionalProperties": false
}
},
"required": [
    "channels"
]
}

```

configs.json

```

{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "id": "http://roomcast.org",
    "type": "object",
    "title": "RoomCast's configurations schema.",
    "description": "Data structure for RoomCast's configurations.",
    "name": "/",
    "properties": {
        "currentConfig": {
            "id": "http://roomcast.org/currentConfig",
            "type": "integer",
            "name": "currentConfig",
            "description": "Id of the current running configuration."
        },
        "launchTime": {
            "id": "http://roomcast.org/launchTime",

```

Appendix A (continued)

```

    "type": "number",
    "name": "launchTime",
    "description": "The time (seconds elapsed since the Epoch) when the
                    current configuration was launched."
  },
  "configs": {
    "id": "http://roomcast.org/configs",
    "type": "object",
    "name": "configs",
    "description": "Map of configurations, accessible via their ids.",
    "patternProperties": {
      "^([0-9]+)*$": {
        "type": "object",
        "name": "configuration",
        "description": "A RoomCast configuration.",
        "properties": {
          "name": {
            "type": "string",
            "name": "name",
            "description": "Name of the configuration."
          },
        },
        "mapping": {
          "type": "array",
          "name": "mapping",
          "description": "Array containing families of devices, each one
                        with a set of packages and associated channels.",
          "items": {
            "type": "object",
            "description": "A mapping of channels for a specific family of
                          devices.",
            "properties": {
              "family": {
                "type": "string",

```

Appendix A (continued)

```

    "name": "family",
    "description": "The name of a family of devices, e.g. 'public
                    display', 'private' or 'tablet'."
  },
  "items": {
    "type": "array",
    "name": "items",
    "description": "List of packages (identities) available,
                    associated to their channels.",
    "items": {
      "type": "object",
      "description": "Object representing a package and its
                      channels.",
      "properties": {
        "name": {
          "type": "string",
          "name": "name",
          "description": "Name of the package."
        },
        "channels": {
          "type": "array",
          "name": "channels",
          "description": "An explanation about the purpose of
                          this instance described by this schema.",
          "items": {
            "type": "string",
            "description": "Id of the channel."
          }
        }
      }
    }
  }
}

```

Appendix A (continued)

```
        }
      }
    }
  }
},
  "additionalProperties": false
}
},
"required": [
  "currentConfig",
  "launchTime",
  "configs"
]
}
```

Appendix B

FINAL QUESTIONNAIRE FOR TEACHERS

The following questionnaire was proposed to the teachers at an advanced phase of the enactment of the Wallcology unit. It is structured following the themes identified by the “5+3 aspects” framework by Prieto (see Section 2.1.3.2): this made it easier to perform an evaluation of the results by comparing them to the orchestration analysis of RoomCast described in Section 3.6.

For each question we asked to answer with a number on a scale from 1 to 5, where 1 meant ‘strongly disagree’ and 5 meant ‘strongly agree’.

Design/Planning

- The operation of deployment of the instructional design into RoomCast (creation of channel packages and activities) is easy to perform, after an appropriate initial training.
- The system has allowed me, by selectively assigning resources (channels) to the different displays in the classroom, to plan a design according to the pedagogical approaches that I want to use.
- RoomCast has asked me to organize student work in a different way than what I’m used to.

Appendix B (continued)

Regulation/management

- The idea of RoomCast of selectively distributing channels to specific displays has effectively helped me regulate and manage the class, e.g. by enforcing the formation of groups in front of the displays and the flow of activities that have to be performed by the students.

Adaptation/Flexibility/Intervention

- The operation of launch of new activities from the iPad application for the teacher during the class period is sufficiently fast and simple to be effectively used at anytime when needed.
- The assignment of channels that are available to the students is sufficiently fast and simple to be effectively used at anytime during the class period (e.g. when we needed to assign a new tool just to one group of kids).
- Overall, I think that RoomCast was useful to adapt to emergent occurrences, such as when a group was late with the work and had to wait before getting access to a new tool.

Awareness/Assessment

- During the class period RoomCast helped increase my awareness of the status of the class (e.g. availability of tools (channels) and tasks, group formation, time progress of the activity).

Models/theories

- I think that the use of the metaphor of the television provider that distributes channels is effective for both teachers and students to understand how the system works.

Appendix B (continued)

Additional aspects

- I noticed that RoomCast helped increase the motivation of the students during specific phases, such as when they were told that a new tool (channel) was coming.
- Overall, I think that RoomCast is easy to use for non-technical teachers, after an appropriate initial training.
- My confidence in using RoomCast has increased over time throughout the different workshops for the teachers with Matteo.

CITED LITERATURE

1. Brown, A. L. and Campione, J. C.: Fostering a community of learners. www.unco.edu/cebs/psychology/kevinpugh/5-7320/ITcomponents/FCL.html. (Visited on 11/12/2015).
2. Fischer, F. and Dillenbourg, P.: Challenges of orchestrating computer-supported collaborative learning. In 87th annual meeting of the American Educational Research Association (AERA), 2006.
3. Dillenbourg, P. and Jermann, P.: Technology for classroom orchestration. In New science of learning, pages 525–552. Springer, 2010.
4. Prieto, L.: An exploration of teacher enactment of cscl activities in computer-integrated classrooms. Master's thesis, Escuela Técnica Superior de Ingenieros en Telecomunicaciones, Universidad de Valladolid, available online at http://gsic.uva.es/%7Elprisan/Prieto2009_ExplorationEnactmentCSCLCiC.pdf Last visit, 12:2010, 2009.
5. Prieto, L. P., Holenko Dlab, M., Gutiérrez, I., Abdulwahed, M., and Balid, W.: Orchestrating technology enhanced learning: a literature review and a conceptual framework. International Journal of Technology Enhanced Learning, 3(6):583–598, 2011.
6. Moher, T., Uphoff, B., Bhatt, D., López Silva, B., and Malcolm, P.: Wallcology: Designing interaction affordances for learner engagement in authentic science inquiry. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 163–172. ACM, 2008.
7. Dillenbourg, P.: Trends in orchestration. second research & technology scouting report. 2011.
8. Dillenbourg, P., Järvelä, S., and Fischer, F.: The evolution of research on computer-supported collaborative learning. In Technology-enhanced learning, pages 3–19. Springer, 2009.
9. Muñoz-Cristóbal, J. A., Prieto, L. P., Asensio-Pérez, J. I., Jorrín-Abellán, I. M., and Dimitriadis, Y.: Orchestrating tel situations across spaces using augmented re-

CITED LITERATURE (continued)

- ality through glue!-ps ar. Bulletin of the IEEE Technical Committee on Learning Technology, 14(4):14, 2012.
10. Dillenbourg, P. and Fischer, F.: Computer-supported collaborative learning: The basics. Zeitschrift fur Berufsund wirtschaftspadagogik, 21:111–130, 2007.
 11. Roschelle, J., Dimitriadis, Y., and Hoppe, U.: Classroom orchestration: synthesis. Computers & Education, 69:523–526, 2013.
 12. Forman, E. A. and Ansell, E.: Orchestrating the multiple voices and inscriptions of a mathematics classroom. Journal of the Learning Sciences, 11(2-3):251–274, 2002.
 13. Kovalainen, M., Kumpulainen, K., and Vasama, S.: Orchestrating classroom interaction in a community of inquiry: Modes of teacher participation. The Journal of Classroom Interaction, pages 17–28, 2001.
 14. Jurow, A. S. and Creighton, L.: Improvisational science discourse: Teaching science in two k-1 classrooms. Linguistics and Education, 16(3):275–297, 2005.
 15. Moon, J.: Short courses & workshops: Improving the impact of learning, training & professional development. Psychology Press, 2001.
 16. Sawyer, R. K.: Creative teaching: Collaborative discussion as disciplined improvisation. Educational researcher, 33(2):12–20, 2004.
 17. Piaget, J. and Inhelder, B.: The psychology of the child. Basic Books, 1969.
 18. Lave, J. and Wenger, E.: Situated learning: Legitimate peripheral participation. Cambridge university press, 1991.
 19. Kimble, C., Hildreth, P. M., and Bourdon, I.: Communities of practice: Creating learning environments for educators, volume 1. IAP, 2008.
 20. Wenger, E.: Communities of practice: Learning, meaning, and identity. Cambridge university press, 1999.
 21. Duffy, T. M. and Jonassen, D. H.: Constructivism and the technology of instruction: A conversation. Routledge, 2013.
 22. Weiser, M.: The computer for the 21st century. Scientific american, 265(3):94–104, 1991.

CITED LITERATURE (continued)

23. Kreitmayer, S., Rogers, Y., Laney, R., and Peake, S.: 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, pages 801–810. ACM, 2013.
24. Moher, T.: Embedded phenomena: supporting science learning with classroom-sized distributed simulations. In Proceedings of the SIGCHI conference on human factors in computing systems, pages 691–700. ACM, 2006.
25. Cober, R., Fong, C., Gnoli, A., Silva, B. L., Lui, M., Madeira, C., McCann, C., Moher, T., Slotta, J., and Tissenbaum, M.: Embedded phenomena for knowledge communities: Supporting complex practices and interactions within a community of inquiry in the elementary science classroom. In Proceedings of the Tenth International Conference of the Learning Sciences, volume 2, pages 64–71, 2012.
26. Horwitz, P., Neumann, E., and Schwartz, J.: Teaching science at multiple space time scales. Communications of the ACM, 39(8):100–102, 1996.
27. Roschelle, J. and Kaput, J. J.: Simcalc mathworlds for the mathematics of change. Communications of the ACM, 39(8):97–99, 1996.
28. Moher, T., Hussain, S., Halter, T., and Kilb, D.: Roomquake: embedding dynamic phenomena within the physical space of an elementary school classroom. In CHI'05 Extended Abstracts on Human Factors in Computing Systems, pages 1665–1668. ACM, 2005.
29. Colella, V., Borovoy, R., and Resnick, M.: Participatory simulations: Using computational objects to learn about dynamic systems. In CHI 98 Cconference Summary on Human Factors in Computing Systems, pages 9–10. ACM, 1998.
30. Colella, V.: Participatory simulations: Building collaborative understanding through immersive dynamic modeling. The journal of the Learning Sciences, 9(4):471–500, 2000.
31. Gnoli, A., Perritano, A., Guerra, P., Lopez, B., Brown, J., and Moher, T.: Back to the future: embodied classroom simulations of animal foraging. In Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction, pages 275–282. ACM, 2014.

CITED LITERATURE (continued)

32. Prieto, L. P., Villagr -Sobrino, S., Jorr n-Abell n, I. M., Mart nez-Mon s, A., and Dimitriadis, Y.: Recurrent routines: Analyzing and supporting orchestration in technology-enhanced primary classrooms. Computers & Education, 57(1):1214–1227, 2011.
33. Dillenbourg, P.: Design for classroom orchestration. Computers & Education, 69:485–492, 2013.
34. Ishii, H. and Ullmer, B.: Tangible bits: towards seamless interfaces between people, bits and atoms. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, pages 234–241. ACM, 1997.
35. Dourish, P.: Where the action is: the foundations of embodied interaction. MIT press, 2004.
36. Hawkey, K., Kellar, M., Reilly, D., Whalen, T., and Inkpen, K. M.: The proximity factor: impact of distance on co-located collaboration. In Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work, pages 31–40. ACM, 2005.
37. Bachl, S., Tomitsch, M., Kappel, K., and Grechenig, T.: The effects of personal displays and transfer techniques on collaboration strategies in multi-touch based multi-display environments. In Human-Computer Interaction–INTERACT 2011, pages 373–390. Springer, 2011.
38. Bligh, B.: On multi-display classroom systems: the affordances and constraints of simultaneous display and non-linear presentation for students and tutors. EDULEARN09 Proceedings, pages 283–292, 2009.
39. David, B., Chalon, R., Delotte, O., Masserey, G., and Imbert, M.: Orchestra: formalism to express mobile cooperative applications. In Groupware: design, implementation, and use, pages 163–178. Springer, 2006.
40. David, B. and Chalon, R.: Orchestration modeling of interactive systems. In Human-Computer Interaction. New Trends, pages 796–805. Springer, 2009.
41. Brown, A. L.: Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. The journal of the learning sciences, 2(2):141–178, 1992.

CITED LITERATURE (continued)

42. Collins, A.: Toward a design science of education. Springer, 1992.
43. Wang, F. and Hannafin, M. J.: Design-based research and technology-enhanced learning environments. Educational technology research and development, 53(4):5–23, 2005.
44. Peltz, C.: Web services orchestration and choreography. IEEE Computer, (10):46–52, 2003.
45. McCutchen, D., Covill, A., Hoyne, S. H., and Mildes, K.: Individual differences in writing: Implications of translating fluency. Journal of educational psychology, 86(2):256, 1994.
46. Gage, N. L.: The scientific basis of the art of teaching. Teachers Coll Press, 1978.
47. Delamont, S.: Teachers as artists. International encyclopedia of teaching and teacher education, 2:6–8, 1995.
48. O'Donnell, A. M. and Dansereau, D. F.: Scripted cooperation in student dyads: A method for analyzing and enhancing academic learning and performance. Interaction in cooperative groups: The theoretical anatomy of group learning, pages 120–141, 1992.
49. Dillenbourg, P. and Hong, F.: The mechanics of cscl macro scripts. International Journal of Computer-Supported Collaborative Learning, 3(1):5–23, 2008.
50. Weinberger, A., Fischer, F., and Mandl, H.: Fostering computer supported collaborative learning with cooperation scripts and scaffolds. In Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community, pages 573–574. International Society of the Learning Sciences, 2002.
51. Santos, L. P. P., Damoulis, I. D., and Pérez, J. I. A.: Supporting orchestration of blended CSCL scenarios in distributed learning environments. Doctoral dissertation, Universidad de Valladolid, Escuela Técnica Superior de Ingenieros de Telecomunicación, Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática, 2012.

CITED LITERATURE (continued)

52. Dillenbourg, P.: Over-scripting cscl: The risks of blending collaborative learning with instructional design. Three worlds of CSCL. Can we support CSCL?, pages 61–91, 2002.
53. Kollar, I., Fischer, F., and Hesse, F. W.: Collaboration scripts—a conceptual analysis. Educational Psychology Review, 18(2):159–185, 2006.
54. Dillenbourg, P. and Jermann, P.: Designing integrative scripts. In Scripting computer-supported collaborative learning, pages 275–301. Springer, 2007.
55. Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämmäläinen, R., Häkkinen, P., and Fischer, F.: Specifying computer-supported collaboration scripts. International Journal of Computer-Supported Collaborative Learning, 2(2-3):211–224, 2007.
56. Haake, J. M. and Pfister, H.-R.: Flexible scripting in net-based learning groups. In Scripting computer-supported collaborative learning, pages 155–175. Springer, 2007.
57. Consortium, I. G. L. et al.: Ims learning design specification, 2003.
58. Koper, R. and Tattersall, C.: Learning design - a handbook on modelling and delivering networked education and training. 2005.
59. Sawyer, R. K.: Creating conversations: Improvisation in everyday discourse. Hampton Press (NJ), 2001.
60. Humphreys, M. and Hyland, T.: Theory, practice and performance in teaching: professionalism, intuition, and jazz. Educational Studies, 28(1):5–15, 2002.
61. Hudak, P. and Berger, J.: A model of performance, interaction, and improvisation. In Proceedings of International Computer Music Conference. Int’l Computer Music Association, pages 1–8. Citeseer, 1995.
62. Brown, M. and Edelson, D.: Teaching by design: Curriculum design as a lens on instructional practice. In Annual meeting of the American Educational Research Association, Seattle, WA, 2001.
63. Dimitriadis, Y., Asensio-Pérez, J. I., Hernáez-Leo, D., Roschelle, J., Brecht, J., Tatar, D., Chaudhury, S. R., DiGiano, C., and Patton, C. M.: From socially-mediated to

CITED LITERATURE (continued)

- technology-mediated coordination: A study of design tensions using group scribes. In Proceedings of the 8th international conference on Computer supported collaborative learning, pages 184–186. International Society of the Learning Sciences, 2007.
64. Dillenbourg, P. and Tchounikine, P.: Flexibility in macro-scripts for computer-supported collaborative learning. Journal of computer assisted learning, 23(1):1–13, 2007.
 65. Tabak, I.: Synergy: A complement to emerging patterns of distributed scaffolding. The journal of the Learning Sciences, 13(3):305–335, 2004.
 66. Hickey, D. T., Moore, A. L., and Pellegrino, J. W.: The motivational and academic consequences of elementary mathematics environments: Do constructivist innovations and reforms make a difference? American Educational Research Journal, 38(3):611–652, 2001.
 67. Järvenoja, H. and Järvelä, S.: How students describe the sources of their emotional and motivational experiences during the learning process: A qualitative approach. Learning and instruction, 15(5):465–480, 2005.
 68. Dillenbourg, P.: Integrating technologies into educational ecosystems. Distance Education, 29(2):127–140, 2008.
 69. Prieto, L. P., Dimitriadis, Y., Asensio-Pérez, J. I., and Looi, C.-K.: Orchestration in learning technology research: evaluation of a conceptual framework. Research in Learning Technology, 23, 2015.
 70. Alavi, H. S., Dillenbourg, P., and Kaplan, F.: Distributed awareness for class orchestration. In Learning in the synergy of multiple disciplines, pages 211–225. Springer, 2009.
 71. Natriello, G.: Imagining, seeking, inventing: the future of learning and the emerging discovery networks. Learning Inquiry, 1(1):7–18, 2007.
 72. Wecker, C. and Fischer, F.: Fading scripts in computer-supported collaborative learning: The role of distributed monitoring. In Proceedings of the 8th international conference on Computer supported collaborative learning, pages 764–772. International Society of the Learning Sciences, 2007.

CITED LITERATURE (continued)

73. Kollar, I., Fischer, F., and Slotta, J. D.: Internal and external scripts in computer-supported collaborative inquiry learning. Learning and Instruction, 17(6):708–721, 2007.
74. Dillenbourg, P., Zufferey, G., Alavi, H., Jermann, P., Do-Lenh, S., Bonnard, Q., Cuenet, S., and Kaplan, F.: Classroom orchestration: The third circle of usability. CSCL2011 Proceedings, 1:510–517, 2011.
75. Tchounikine, P.: Clarifying design for orchestration: orchestration and orchestrable technology, scripting and conducting. Computers & Education, 69:500–503, 2013.
76. Sharples, M.: Shared orchestration within and beyond the classroom. Computers & Education, 69:504–506, 2013.
77. Zufferey, G., Jermann, P., Lucchi, A., and Dillenbourg, P.: Tinkersheets: using paper forms to control and visualize tangible simulations. In Proceedings of the 3rd international Conference on Tangible and Embedded interaction, pages 377–384. ACM, 2009.
78. Ronen, M., Kohen-Vacs, D., and Raz-Fogel, N.: Adopt & adapt: structuring, sharing and reusing asynchronous collaborative pedagogy. In Proceedings of the 7th international conference on Learning sciences, pages 599–605. International Society of the Learning Sciences, 2006.
79. Demetriadis, S. and Karakostas, A.: Adaptive collaboration scripting: A conceptual framework and a design case study. In Complex, Intelligent and Software Intensive Systems, 2008. CISIS 2008. International Conference on, pages 487–492. IEEE, 2008.
80. Honegger, B. D. and Notari, M. P.: Over-computing cscl macro scripts? Gaining flexibility by using WikiPlus instead of specialized tools for authoring macro scripts. In Proceedings of the 9th international conference on Computer supported collaborative learning-Volume 1, pages 482–486. International Society of the Learning Sciences, 2009.
81. Slotta, J. D., Tissenbaum, M., and Lui, M.: Orchestrating of complex inquiry: three roles for learning analytics in a smart classroom infrastructure. In Proceedings of the Third International Conference on Learning Analytics and Knowledge, pages 270–274. ACM, 2013.

CITED LITERATURE (continued)

82. Slotta, J. and Peters, V.: A blended model for knowledge communities: Embedding scaffolded inquiry. In Proceedings of the 8th international conference on International conference for the learning sciences-Volume 2, pages 343–350. International Society of the Learning Sciences, 2008.
83. Slotta, J., Tissenbaum, M., and Lui, M.: Sail smart space: Orchestrating collective inquiry for knowledge communities. In Proceedings of the Ninth International Computer-Supported Collaborative Learning Conference, pages 1082–1083, 2011.
84. Slotta, J.: Evolving the classrooms of the future: The interplay of pedagogy, technology and community. Classroom of the future: Orchestrating collaborative spaces, pages 215–242, 2010.
85. Roschelle, J. and Pea, R.: A walk on the wild side: How wireless handhelds may change computer-supported collaborative learning. International Journal of Cognition and Technology, 1(1):145–168, 2002.
86. Bakker, S., van den Hoven, E., and Eggen, B.: Fireflies: supporting primary school teachers through open-ended interaction design. In Proceedings of the 24th Australian Computer-Human Interaction Conference, pages 26–29. ACM, 2012.
87. Muñoz-Cristóbal, J. A., Prieto, L. P., Asensio-Pérez, J. I., Jorrín-Abellán, I. M., Martínez-Monés, A., and Dimitriadis, Y.: Glueps-ar: A system for the orchestration of learning situations across spaces using augmented reality. In Scaling up Learning for Sustained Impact, pages 565–568. Springer, 2013.
88. DiGiano, C., Tatar, D., and Kireyev, K.: Learning from the post-it: Building collective intelligence through lightweight, flexible technology. In Conference on Computer Supported Cooperative Work Companion, Banff. [http://Group Scribbles. sri. com/publications/index. html](http://GroupScribbles.sri.com/publications/index.html), 2006.
89. Prieto, L. P., Asensio-Pérez, J. I., Muñoz-Cristóbal, J. A., Jorrín-Abellán, I. M., Dimitriadis, Y., and Gómez-Sánchez, E.: Supporting orchestration of cscl scenarios in web-based distributed learning environments. Computers & Education, 73:9–25, 2014.
90. MacNeill, S. and Kraan, W.: Distributed learning environments: A briefing paper. London, JISC CETIS, 2010.

CITED LITERATURE (continued)

91. Muñoz-Cristóbal, J. A., Prieto, L. P., Asensio-Pérez, J. I., Martínez-Monés, A., Jorrín-Abellán, I. M., and Dimitriadis, Y.: Deploying learning designs across physical and web spaces: Making pervasive learning affordable for teachers. Pervasive and Mobile Computing, 14:31–46, 2014.
92. Munoz-Cristobal, J., Jorrin-Abellan, I. M., Asensio-Perez, J., Martinez-Mones, A., Prieto, L. P., Dimitriadis, Y., et al.: Supporting teacher orchestration in ubiquitous learning environments: A study in primary education. Learning Technologies, IEEE Transactions on, 8(1):83–97, 2015.
93. Rick, J.: Towards a classroom ecology of devices: Interfaces for collaborative scripts. 2009.
94. Martinez Maldonado, R., Dimitriadis, Y., Kay, J., Yacef, K., and Edbauer, M.-T.: 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, pages 119–128. ACM, 2012.
95. Gnoli, A.: nutella: the construction and enactment of simulated macroworlds. Doctoral dissertation, University of Illinois at Chicago, to appear.
96. Gibson, J. J.: The theory of affordances. Hoboken, NJ, 1977.
97. Norman, D. A.: The Design of Everyday Things. Basic Books, 2002.
98. Kirschner, P., Strijbos, J.-W., Kreijns, K., and Beers, P.: Designing electronic collaborative learning environments. Educational Technology Research and Development, 52(3):47–66, 2004.
99. Kreijns, K., Kirschner, P. A., and Jochems, W.: The sociability of computer-supported collaborative learning environments. Educational Technology & Society, 5(1):8–25, 2002.
100. Kirschner, P. A.: Can we support CSCL? Educational, social and technological affordances for learning. In Three worlds of CSCL: Can we support CSCL, ed. P. K. (Ed.). Heerlen: Open University of the Netherlands, 2002.
101. Rothstein-Fisch, C. and Trumbull, E.: Managing Diverse Classrooms: How to Build on Students? Cultural Strengths. ASCD, 2008.
102. Cohen, L.: Manion. l. and morrison, k.,(2007). research methods in education.

CITED LITERATURE (continued)

103. Denzin, N. K. and Lincoln, Y. S.: The SAGE handbook of qualitative research. Sage, 2011.
104. Dimitriadis, Y., Prieto, L. P., and Asensio-Pérez, J. I.: The role of design and enactment patterns in orchestration: Helping to integrate technology in blended classroom ecosystems. Computers & Education, 69:496–499, 2013.
105. Luckin, R.: The learner centric ecology of resources: A framework for using technology to scaffold learning. Computers & Education, 50(2):449–462, 2008.
106. Angeli, C.: Transforming a teacher education method course through technology: Effects on preservice teachers' technology competency. Computers & Education, 45(4):383–398, 2005.
107. Glazer, E., Hannafin, M. J., and Song, L.: Promoting technology integration through collaborative apprenticeship. Educational Technology Research and Development, 53(4):57–67, 2005.

VITA

NAME	Matteo Palvarini
EDUCATION	<p>University of Illinois, Chicago, IL - M.S. Computer Science, December 2015</p> <p>Politecnico di Milano, Milan, Italy - M.S. Computer Engineering, December 2015</p> <p>Politecnico di Milano, Milan, Italy - B.S. Computer Engineering, September 2013</p>
TECHNICAL SKILLS	<p>Front End skills HTML5, CSS3, JavaScript, d3.js, React.js, AngularJS, Node.js</p> <p>Other languages C, Java, Java EE, JSF, PHP, Ruby, Python, R, Swift (iOS and OSX)</p> <p>DB and cloud MySQL, MongoDB, Amazon AWS, EC2, S3 SAP Business ByDesign</p> <p>Tools and SW Git, SVN, Eclipse, WebStorm, Xcode Photoshop, Illustrator, Sketch, RapidMiner, Weka, RStudio</p>
LANGUAGES	<p>Italian Native speaker</p> <p>English Full working proficiency</p>
WORK EXPERIENCE	<p>Jan 2015 - Dec 2015 Research Assistant - University of Illinois at Chicago</p> <ul style="list-style-type: none">• worked as RA in Human-Computer Interaction applied to the Learning Sciences domain built, with a design-based research approach, interactive applications for primary school classrooms• built, with a design-based research approach, interactive applications for primary school classrooms <p>Jan 2015 - May 2015 Teaching Assistant - University of Illinois at Chicago</p> <ul style="list-style-type: none">• worked as TA for the course CS 422 - User Interface Design and Programming• graded assignments, projects and exams, coordinated in-class discussions, managed online activities

VITA (continued)

SELECTED PROJECTS

Aug 2015	Dropverse - Dropbox Hack Week 2015 <ul style="list-style-type: none"> • designed and implemented, in a team of 6, an interactive, web-based data visualization for a large display, showing the level of cross-functionality of the meetings happening between employees inside Dropbox • implemented a dedicated iPad interface to control the time for the data displayed by the main app • presented the work to the CEO and the VP of Engineering and got a special mention at the closing ceremony
Nov 2014 - Dec 2014	Right Here, Right Now - CS 424 Visualization and Visual Analytics <ul style="list-style-type: none"> • implemented, in a team of 4, an interactive data visualization tool which helps users make decisions about their travel plans in Chicago by enabling several different layers of real-time data, using d3 and custom SVG graphics • focused on UI and interactions design, including a real-time feed of relevant tweets (1500 sloc)
Jun 2014	MLSP 2014 - Schizophrenia Classification Challenge <ul style="list-style-type: none"> • implemented, in a team of 4, a Rapid Miner application, parallelized on EC2 to perform efficient feature selection • wrote an R script to apply visualization techniques on the training set to get insights and meaning from the data • reached the 5th position of the Leaderboard on kaggle.com
Oct 2013 - Dec 2013	Travel Dream - Software Engineering 2 <ul style="list-style-type: none"> • created an e-commerce website for travels, using JavaEE (client-tier logic) and JSF (front-end), 4000 sloc • produced detailed documentation: RASD, Design Document, Installation Guide, User Acceptance Testing

SCHOLARSHIPS

Fall 2014	UIC full tuition waiver - scholarship for the student who obtained the best results in the first three classes of the Master, among all the enrolled Italian students
April 2010	Winner of philosophy contest - starting from an extract from "Sociology of Religion" by Max Weber, with an essay entitled "Happiness as an individual's need for relative self-assertiveness" (translated from Italian)
