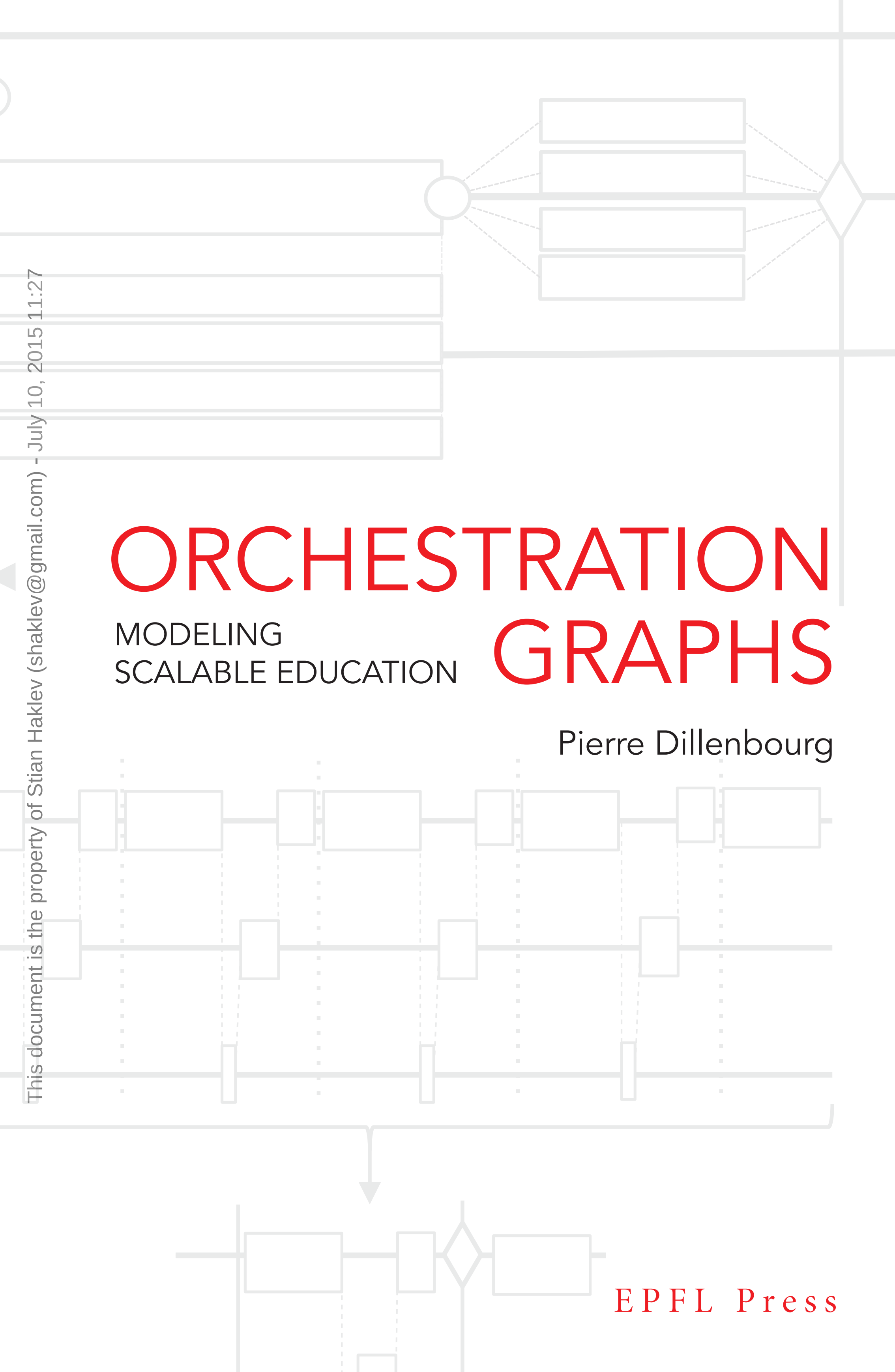


ORCHESTRATION GRAPHS

MODELING SCALABLE EDUCATION

Pierre Dillenbourg

EPFL Press

ORCHESTRATION GRAPHS

MODELING SCALABLE EDUCATION

ORCHESTRATION GRAPHS

MODELING SCALABLE EDUCATION

Pierre Dillenbourg

EPFL Press

The author and publisher express their thanks to the Ecole polytechnique fédérale de Lausanne for its generous support towards the publication of this book.

EPFL Press

is an imprint owned by the Presses polytechniques et universitaires romandes, a Swiss academic publishing company whose main purpose is to publish the teaching and research works of the Ecole polytechnique fédérale de Lausanne (EPFL).

Presses polytechniques et univeristaires romandes
EPFL – Rolex Learning Center
Post office box 119
CH-1015 Lausanne, Switzerland
E-mail: ppur@epfl.ch

www.epflpress.org

ISBN (version papier): 978-2-940222-84-1
ISBN (version numérique): 978-2-88914-378-8
Printed in Switzerland

A Christine, Dinesh, Savita, et Naël

Contents

Summary

A sequence of learning activities can be modeled as a graph with specific properties. The vertices or nodes of the graph are the learning activities. Learners perform some of these activities individually, some in teams, and others with the whole class. The graph has a geometric nature, time being represented horizontally and social organization (individual, team, class) vertically. These activities can be inspired by heterogeneous learning theories; a graph models the integration of heterogeneous activities into a coherent pedagogical scenario. The edges of the graph connect activities. They represent the two-fold relationship between activities—how they relate to each other from a pedagogical and from an operational viewpoint.

From the operational viewpoint, edges are associated with operators that transform the data structures produced during a learning activity into the data structures needed to run the next activity. In this book I will present 26 operators classified into 5 categories. A sequence of operators constitutes a workflow.

From the pedagogical viewpoint, an edge describes why an activity is necessary for the next activity; it can, for instance, be a cognitive prerequisite, a motivational trick, an advanced organizer, or an organizational constraint. The edges are classified in a library of 28 pedagogical ideas. The extent to which one activity is necessary for the next one is encompassed in the weight of an edge. The transition between two activities is stored as a matrix; the cell (m,n) of a transition matrix stores the probability that a learner in cognitive state m will evolve to state n in the next activity. I propose a list of 28 states that have a specific meaning in education. The transition matrix can be summarized by a parameter that constitutes the edge weight; an edge between two activities has a heavy weight if learner performance in one activity is highly predictive of success in a connected activity. The graph also constitutes a probabilistic network that allows predicting the future state of a learner.

When the pedagogical scenario is running, the actual state of the learner can be inferred not only from his past activities but also from his current behavior and from the activities of others. Learner modeling combines these 3 sources of information and is hence represented as a cube.

This book does not propose a learning theory. It describes how rich learning activities, often designed for small classes, can be scaled up for use with thousands of participants, as in MOOCs. It also describes how a pedagogical

scenario can be adapted to the level of participants or repaired on the fly when problems occur. The graph describes how the scenario can be modified, stretched, cut, and extended. Orchestration refers to the real-time management of pedagogical scenarios to ensure the maximum the satisfaction of many constraints, listed in this book.

Introduction

This book proposes a language for modeling the design and the orchestration of sequences of learning activities. In a nutshell, orchestration refers to the real-time management of pedagogical scenarios and their permanent adaptation to the many constraints that have to be satisfied for a lesson to “work well”, as would a teacher say. This modeling language relies on graphs that describe educational scenarios from four different viewpoints. In Chapter 1, the graph describes the structure of activities; who does what and when. It models the **visible** part of the educational activities. Chapter 2 proposes a library of **pedagogical ideas** that underlie the graph structure. In Chapter 3, the graph describes the **workflow** underneath rich scenarios, every edge of the graph being potentially associated with data transformation operators. In Chapter 4 and Chapter 5, the graph describes the learner path as a **stochastic** process, every edge being associated with a probability matrix. In the last chapter, I explore how analytics contribute to the real-time management of scenarios and the improvement in their effectiveness over time.

A modeling language is not a theory that predicts how people will learn but a tool for designers of pedagogical scenarios, as well as scholars working on learning analytics. A theory can be proven or rejected with empirical data. A modeling language is not true or false but rather may or may not be useful for those who attempt to express themselves with this vocabulary and this syntax. My motivation for proposing a formal language is described in the next paragraphs.

First, modeling lies at the heart of any scientific field. It constitutes a legitimate activity for any scholar and does not require any justification in terms of outcome or application. Throughout the book, I try to find a balance between the lack of formalism in learning sciences and the risk of writing down meaningless formulae; quantifying the rich semantics of educational dialogues, along with the subtle social cues of social situations, could lead to a caricature of science. Of course, education is not about numbers; it is about inspiring humans, about passion, and about creating a warm social atmosphere. Nonetheless, I am utterly convinced that there is something deeply rational, almost algorithmic, in the art of teaching. This rational layer constitutes the slice of the educational realm that defines the perimeter of this book. This modeling ambition is not restricted to online education, but concerns any classroom lesson. I will return to this point several times in the book.

Beyond the intrinsic legitimacy of modeling, a second motivation is to improve the dialogue between learning sciences and computer science.

Computer scientists often complain that the learning sciences fail to provide them with formal models of education. This book results from an attempt to make a modest step in that direction. John Self (1992) named this effort "computational mathetics," which could have been the title of this book. The proposed modeling language translates educational scenarios into computational structures that are familiar to computer scientists, such as graphs or Markov chains. This book does not describe a platform where language, graphs, and operators are implemented. Such a platform does not exist, and this book does not aim to provide the specifications of such a platform. Instead, the proposed modeling language could be used upstream of the implementation process and influence more than one educational platform.

On the more applied side, the proposed modeling language could enhance learning technologies in several ways.

- This work has been inspired by the wish to enrich the pedagogy of Massive Open Online Courses (MOOCs). Many rich learning activities have been empirically validated with small classes, but, at first glance, it seems difficult to scale them up to thousands of learners. Are MOOCs condemned to rely on simple learning activities such as watching videos, answering quizzes, and doing exercises? This book hypothesizes that a formal description of pedagogical scenarios will allow rich activities to be run on a large scale.
- A modeling language could enhance the power of learning analytics; that is, the processing of traces left by learners during their activities. When 500 sensors are distributed over a mountain, the 3D model of this mountain and the precise location of each sensor are necessary to integrate the data collected from the sensors into a consistent dataset. The graphs presented in this book constitute the 3D model of a lesson. The three dimensions are time, social structures, and diagnosis inference levels. I postulate that learning analytics will exploit the traces of learners more accurately once these traces have been mapped onto a model of the learning activities.
- If the modeling language was used by different platforms, data from different MOOCs could be compared, since it would be possible to identify what is comparable or not between two MOOCs. I hope that one day data from MOOCs will be made available to the whole learning community, and not only to the institution that owns the MOOC. Other scientific fields made a quantum leap forward when they succeeded in efficiently sharing their data. This major achievement would, however, remain unproductive unless there was a description of MOOCs precise enough to support comparisons.
- This modeling language could provide the programming abstractions required to make the code of learning environments more explicit and less ad-hoc.

This book was written for researchers in education and learning technologies, as well as for computer scientists developing new educational environments. The proposed language is suited to describing learning technologies, especially online education—namely MOOCs. However, the scope of this book is not limited to MOOCs. I believe that the proposed modeling language is relevant for any formal educational context. At a first glance, a lesson on subtraction for 20 pupils in an elementary school classroom does not have much in common with an academic MOOC on digital signal processing, with 20,000 students. Under the surface though, both can be modeled as learning activity graphs. In other words, this is not a book restricted to MOOCs, but the rise of MOOCs has prompted the need to formalize rich learning activities in order to bring them to scale.

Orchestration graphs describe the space of pedagogical scenarios that can be elaborated. A scenario is a sequence of activities devoted to a set of learning objectives. This book is neither a practical guide for creating a scenario or a MOOC, nor a theory that prescribes how scenarios or MOOCs should be designed. Learning theories have to be “proven” in some way. A modeling language is neither true nor false; it may or may not be useful. Therefore, this model will be validated if future platforms are implemented on the bases presented in this book and if these implementations actually enable rich pedagogical scenarios on a large scale. Another way in which it could be useful is in providing a more specific description of MOOC activities, thereby enhancing learning analytics.

This book contains some mathematical notations. Although they are not necessary for understanding the modeling language, I hope that learning scientists will nonetheless find them useful. These mathematical elements are not intended to be a contribution to either mathematics or machine learning; more modestly, I have used them as anchors for articulating the educational processes to existing computational models.

The presentation of the modeling language is arbitrarily segmented into 42 points. Other than the nod to marathon friends, this didactic decomposition has no deeper meaning than building the model incrementally.

Any modeling activity implies some simplification of the reality being modeled, acting as a lens that reveals some aspects but ignores others. While this book only looks at the social and cognitive dimensions of learning, this is not to deny the importance of other dimensions, such as emotional and cultural ones. I will come back to the limitations at the end of the book. Simplification is the price to pay for elaborating a new modeling language. Later on, this language will have to be enriched in order to cover more dimensions of the education process.

About Scale

MOOCs have attracted millions of students, and yet, their pedagogy is often less sophisticated than state of the art pedagogy in learning sciences. This book explores the possibility of extending rich learning activities to large audiences. The scale is the ratio between the number of teachers and the number of students in the same class. It ranges from 1:1 (e.g., a private ski lesson), to 1:25 in European elementary schools, to 1:80 or 1:400 in lecture theatres, and up to 1:150,000 in MOOCs. Actually, a more realistic ratio would be 5:150,000, as there are probably 4 teaching assistants who would second the teacher in such a MOOC. When I write "the teacher" in this book, I am referring to the whole teaching team.[1] For teacher and learners, I use "he" as a gender-neutral pronoun that could be read as "he or she."

What is the relationship between scale and pedagogy? Scaling up learning activities can be viewed as a loss of pedagogical richness, made acceptable by the benefits of giving broad access to education for a marginal increase in cost. Though for simple learning activities, scaling up actually implies only a negligible loss of quality; for instance, the amount that John learns from watching a video or from answering quizzes will not vary greatly if there are 10,000 other students doing the same activity. However, some learning activities that are manageable with small classes do not scale up; for instance, guided discovery (e.g., learning from a simulation) is only effective with guidance at an appropriate level (De Jong & van Joolingen, 1998), which can partly be automated, but which does not scale as easily, as illustrated by Figure 0.1. Team problem solving is also limited in scale, since the core mechanisms of shared meaning making are limited to small groups. Of course, a class of 5,000 students could be divided into 1,000 teams of 5 students. However, since self-regulation skills are often below what teams require in order to be effective, it cannot be trusted that these teams would in fact self-regulate, and the management of 1,000 teams by a teacher would prove intractable.

Some activities scale well, some don't. Does it matter? Would 1:10,000 activities be more effective if they reproduced the pedagogical scenarios used in 1:1 situations? The ratio 1:1 is often presented as the optimal condition for learning. The idea behind "individualized instruction" is to keep large class activities as effective as 1:1 activities. A seminal paper from Bloom (1984) showed that a 1:30 lesson can get close to 1:1 efficiency if the learners' prerequisites are systematically consolidated before the lesson. But is 1:1 really the Holy Grail in education? A Chinese tradition was for a prince not to learn alone with a tutor but for another child to be brought to the palace to learn with him (Chan & Baskin, 1988). The scale 1:2 was considered as better for the prince than the 1:1 scale. In the Swiss vocational education system, an apprentice works 4 days per week in a company, understanding one specific context. On his weekly day at school, he can share the experience of the 19 other apprentices working in other companies. He is able to discover the variety of processes that exist for the same tasks across various workplaces and thereby disentangle what is specific to each context from what is true in any context.

---

[1] A new phenomenon has emerged with MOOCs—in some cases, so many people volunteered to be teaching assistants that the scale became 25:10,000. We can't simplify it to 1:400, since 25:10,000 requires a much more complex organization than 1:400.

In this context, 1:20 is pedagogically richer than 1:1. What if activities could be more effective with 1,000 students than with 20? The philosophy of this book is to view scale as a pedagogical opportunity instead of as a phenomenon that filters out rich learning interactions.

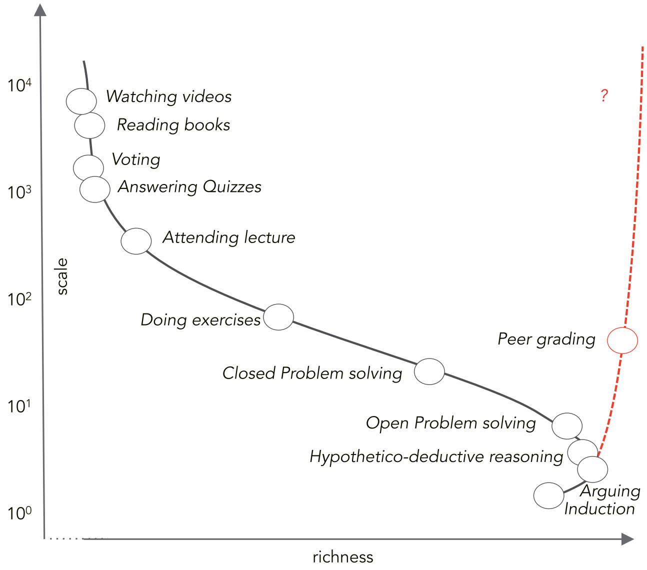


Figure 0.1 The relationship between the interaction richness of a pedagogical scenario and its scale. This graph is not built from empirical data but is based on common sense or experience in educational practices. The red line corresponds to the aims of orchestration graphs—to be able to scale up rich learning activities without sacrificing their interaction richness. On the red line, we have to invent activities that take advantage of scale. Peer grading constitutes a first example on this path.

In learning technologies, the balance between individual adaptation and social interaction has evolved over decades. Since the 1960s, individual adaptation has been the leitmotiv of computer-based education. If the key is to adapt instruction to individual needs, learners should work individually on computers. However, since in the past there were more students than computers, two or more learners were often assigned to one machine. Surprisingly, this situation turned out to be more effective than individual work (Dickson & Vereen, 1983); the “loss” due to a lower accuracy in individualization seemed to be somehow overcompensated for by the gains generated from verbal interactions in front of or through a computer. This gave birth to the field of computer-supported collaborative learning (CSCL). This evolution in learning technologies is represented in Figure 0.2 as a left movement of the pendulum. The evolution from individualized instruction to CSCL is depicted by the red arrow in the in the 1990 pendulum.

In early years of 2000, researchers in CSCL understood that collaborative activities had to be more structured in order to be effective. The mere fact that teams work together on a task does not guarantee that they engage in rich interactions such as explanation, argumentation, or mutual regulation. Along with some colleagues, my team therefore developed so-called collaboration scripts (Dillenbourg, 2002; Weinberger & Fischer, 2006), where a script is a pedagogical scenario that structures team interactions by defining roles, phases, differences in viewpoints, and so on. For instance, a well-known script, called “Jigsaw” (Aronson et al., 1978), provides each team member with a subset of the information required to carry out the task. To achieve its goal, the team therefore needs to integrate the contribution of each individual

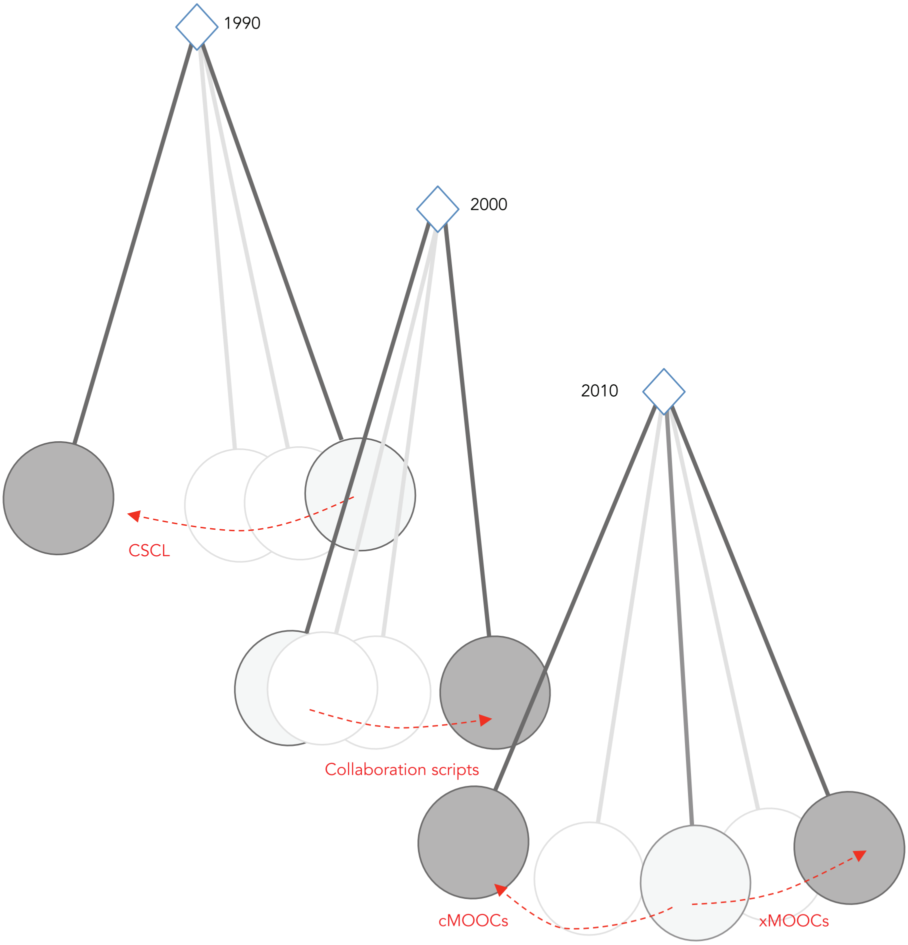


Figure 0.2 The individual-social pendulum in learning technologies—the individual force pulls the ball to the right; the social force pushes it to the left.

member. Another example of script is ArgueGraph (Dillenbourg & Jermann, 1999; Dillenbourg & Hong, 2008); this scenario fosters argumentation by forming pairs of individuals with contrasting opinions. In the 2000 pendulum in Figure 0.2, the development of script is depicted as a pendulum movement to the right, back to the center of the social-individual continuum, because scripts prescribe individual processes within team processes. Team cognition is not viewed as an emergent property but is "engineered" by fine-tuning individual processes within teamwork. These scripts can be viewed as the ancestors of the graphs presented in this book, the latter being a generalization of the former. The ArgueGraph script will namely be used in this book to explain the functional role of a workflow in a pedagogical scenario.

The 2010 pendulum in Figure 0.2 depicts recent movements of the pendulum in two opposite ways—to the left side of the continuum, which corresponds to cMOOCs, based on connectivism (Siemens, 2005), and to the right side, the academic teaching style reflected in xMOOCs. The large scale MOOCs distributed by actors such as Coursera or EdX have been xMOOCs. This distinction has somehow become obsolete, but, in a nutshell, cMOOCs build upon the social nature of learning, while xMOOCs focus on individual learning. Actually, this x/c dichotomy is more conceptual or ideological than a real conflict in practice. In a pedagogical scenario, educational practices are not exclusive; some activities can be close to x and others closer to c. For instance, we asked students to watch xMOOCs in teams of 4, and they reported this as being a very positive experience (Li et al., 2014). The complementarity of learning activities inspired by divergent learning theories is central to this book. The need to integrate activities into a consistent learning scenario, despite the heterogeneity of these activities, is very much reflected in the modeling language.

Personal note

The style of this book is sometimes schizophrenic. I propose a formal language that emphasizes the rational side of education, but, from time to time, I add comments based on personal values. This dualism is intrinsic to education—even when scientists argue rationally and rigorously about data, methods, or theories, their reasoning is shaped by their values and ideas of what education should be. Any educational debate mixes rigor and emotion. A scientist should never "believe" in his theory, despite the evidence, and an educationalist should always "believe" in the learner's success, despite the obstacles. This is why an educational scientist is schizophrenic. For me, an effective lesson (when all the students are "with" the teacher) is as admirable as a snake-like ski trace in powder snow. Effectiveness is beauty. This will appear in the text as I make occasional digressions from the modeling language.

Chapter 1

Orchestration Graphs

The modeling language proposed in this book aims to describe every kind of pedagogical scenario. By "pedagogical scenario," I mean any sequence of learning activities, also referred to as a lesson plan, a didactic sequence, or a script. A pedagogical scenario will be modeled as a graph, with specific properties that I will present throughout the 42 points of this book. Describing a lesson plan as a graph may seem pointlessly complex as most lessons, as well as most existing MOOCs, are built as a linear sequence of activities. However, pedagogical scenarios based on a richer structure also exist, and, with orchestration graphs, I aim to capture these richer scenarios. Therefore, I will extract the workflow that underlies these scenarios; that is, the (often invisible) sequence of data transformations between activities. I hypothesize that formalizing the workflow of a pedagogical scenario will enable our community to scale up learning activities that may initially appear as non-scalable.

A model is like a lens; it constitutes a specific way of looking at reality, emphasizing some elements, variables, or phenomena. The proposed modeling language pays particular attention to the way educational scenarios concretely unfold with time, what happens practically during the course, and what needs to be repaired. The term "orchestration graph" embeds this pragmatic viewpoint: "orchestration" refers to the real-time management of learning activities (Dillenbourg, 2013) such as launching activities, managing time, circulating data, and adapting the scenario on the fly. The term "orchestration" is not the optimal metaphor, because orchestration differs from conducting an orchestra. I will come back to this controversy in Point 41. But, intuitively, it is true that there is a touch of the maestro in the performance of a talented teacher enthusiastically conducting rich class activities, running across multiple planes (individual, team, and class activities—see Point 2), with or without computers. How does an elementary school teacher adapt the current activity if two students missed the previous activity or if a crane comes into view in front of the classroom window and starts to operate? How does a university lecturer react when he notices that he is losing the attention of his audience? How does a MOOC teacher cope with thousands of complaints that the last video was incomprehensible? Of course, the adaptation of instruction is not a new topic, but this term usually refers to the process of adapting instruction to individual learner needs. The notion of orchestration is broader. It includes the selection of the most appropriate activities for the learners, but it also extends to very practical aspects of classroom management such as managing time ("*I have 5 minutes left—it's too late to start a new chapter*"), managing space ("*My classroom is too small to move chairs while forming teams*"), assessment constraints ("*I have to issue individual grades because the school does not accept team grades*"), energy constraints ("*I will use peer grading because the number of assignments to grade is too high*"), and safety constraints ("*The students cannot explore the city on their own, so each team will investigate architectural aspects of the same street*"). Managing a lesson or a MOOC requires continual regulation—monitoring the learners' activity, adapting some activities, or even modifying the scenario. For this reason, this book follows a kind of systemic viewpoint on the educational ecosystem, in which the scenario is a species that constantly needs to adapt to the constraints of its environment in order to "work well." As a consequence, the proposed modeling language has to capture the flexibility of a pedagogical scenario, that is, the features that make the orchestration process easy or difficult.

There is a contradiction between these two first paragraphs: the first one claims that scale requires automated workflows, while the second explains that orchestration requires flexibility. This tension will be present throughout this book—a "flexible workflow" is an oxymoron, like a deafening silence. This tension constitutes an interesting challenge for computer science: how to modify, skip, and reorder the various steps of data processing without breaking data consistency?

A distinctive feature of orchestration graphs is the integration of heterogeneous activities; individual learning activities, teamwork, and lectures are integrated into a single pedagogical scenario—hopefully one that is consistent. The proposed language breaks down the walls of didactic churches by articulating activities inspired by different learning theories (e.g., behaviorism, mastery learning, constructivism, and socio-cultural theories) within the same scenario. Of course, orchestration graphs can also be used to model homogeneous pedagogical scenarios, that is, scenarios in which all activities are inspired by the same learning theory. The point is that the design of heterogeneous scenarios requires paying special attention to how diverse learning activities are integrated into a consistent whole—both pedagogically consistent and operationally consistent. Orchestration graphs have been designed to express this consistency.

In summary, the language I propose in this book describes pedagogical scenarios that (1) integrate heterogeneous activities (2) into a workflow that (3) remains flexible. The language will therefore provide several viewpoints on a pedagogical scenario.

- In Chapter 1, a graph describes the structure of activities: who does what and when. It models the **visible** part of the educational activities. The eight points of this chapter describe each constituent of an orchestration graph.

- In Chapter 2, a set of graph edges is described: edges embed the pedagogical ideas or the **rationale** underlying the scenario.
- In Chapter 3, a graph describes the **workflow** underneath rich scenarios, with some edges of the graph being associated with a data transformation operator. This chapter describes several categories of operators and proposes design patterns that can be built with these operators.
- In Chapter 4, a graph describes the learner path as a **probabilistic** network: every edge is associated with a transition matrix. The eight points of this chapter articulate stochastic models with graph components and lay the groundwork for the development of learning analytics in Chapter 5.
- Finally, Chapter 6 analyses the concept of orchestration with the concepts and the modeling language developed along the previous chapters.

Point 1 Horizontal axis

Orchestration graphs are based on sequences[1] of activities placed in a two-dimensional space. I will start with the horizontal dimension, which determines the position of an activity in time. This first point begins with the basics; more substance will come in the next points. To illustrate this axis, let's consider a scenario for a geometry course in an elementary school. The classroom holds 25 children who are about 9 years old.

10:00 a.m. Activity 1	The teacher summarizes the lesson from the previous week where pupils learned to calculate the surface of a rectangle, first by counting square units, and then by measuring its dimensions. He then informs the students that the goal of the current lesson is to use similar methods to measure the surface of a triangle.
10:10 a.m. Activity 2	The teacher forms pairs, giving 2 paper rectangles of the same size, but of different shapes to each pair. He asks the students to measure the surface of these rectangles, with each member of the pair using one of the two methods from the previous week, and then to compare the results.
10:15 a.m. Activity 3	The teacher asks the pairs to cut the rectangles in such a way that one rectangle becomes 2 triangles, to calculate the surface of these triangles, and to compare it to the surface of the rectangle.
10:30 a.m. Morning break	
10:45 a.m. Activity 4	The teacher distributes non-rectangular triangles and asks teams to search for a method for measuring surfaces. After a while, he invites the teams to use scissors to find a solution.

---

[1] Later on, graphs will include richer structures than sequences and therefore justify the term "graph."

11:00 a.m. Activity 5	The teacher asks the pupils to show their solutions to the class. Then, the teacher summarizes the solution to the class and demonstrates how to find the height of any triangle.
11:10 a.m. Activity 6	Each student receives a sheet with 5 triangles and is asked to draw the altitude of the triangles and calculate their surface.
11:25 a.m. Activity 7	The pupils have to copy to their notebook the summary written on the blackboard by the teacher.

This scenario can be modeled as a simple sequence of activities: doing an exercise, writing a summary, listening to the teacher, and copying the blackboard contents. Some of them (e.g., activity 7) are not "learning" activities as such, but are activities performed by the learners (see Point 4). These activities actually populate a significant part of classroom life, but have rarely been taken into account by educational theories. In Figure 1.1, each activity is modeled as a rectangle whose length is proportional to the activity duration and that is placed from left to right in chronological order. The 10:30 a.m. pause appears as a gap on this timeline.

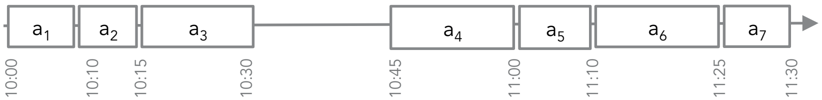


Figure 1.1 A simple sequence of learning activities.

An activity will be defined by several more parameters that will be incrementally introduced in this chapter. Let's start with a simple definition:

A pedagogical scenario is a sequence (a_1, a_2, … , a_i, … , a_n) where a_i is the i^{th} learning activity.

Let's now examine what can be considered as a radically different context—an online course at university level. This is a fictitious example.

Week 1	In a MOOC on urbanism, online students follow video lectures 1, 2, and 3.
Weeks 2–3	Students have to write a case study about a city, selected from a list of 10 cities proposed by the teacher. They upload their report as a PDF file.
Week 3	The system creates a forum for each of the 10 cities. In the forums, students are asked to discuss the main urbanism challenges in the city they have selected. They draw up a list of challenges and vote for the 3 main ones.
Week 4	Each student has to analyse 2 case studies from another city, with respect to the 3 challenges selected on Week 3.

Week 5	Students follow video lectures 4 to 6.
Week 6	Students have to annotate a map of the city they selected and point out 3 places that illustrate the concepts taught in lecture 6.
Week 7	The system automatically forms groups of students who annotated the same map area for the same city. It creates an online conferencing space for them and asks them to produce a common map.
Week 8	The teacher presents a real-time video lecture where he discusses the most interesting and least interesting maps produced by students.

This scenario can be represented by the same kind of timeline as the geometry lesson:

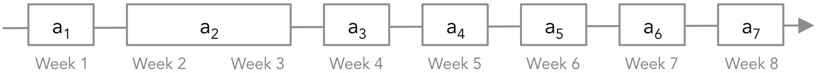


Figure 1.2 A simple sequence of activities in a MOOC.

As emphasized in the introduction to this chapter, a distinctive feature of these two examples is that they are composed of heterogeneous activities. Some activities are performed individually, others are done in teams, and some are conducted with the whole class. Some activities are computer-based and others don't rely on digital technology.[2] For too long, pedagogical practices were clustered together by unnecessary orthodoxies such as behaviorism, constructivism, project-based learning, and problem-based learning. However, in daily practice, teachers tend to integrate various models. For instance, elementary school teachers blend global and analytic methods for learning to read. At university level, lecturing is combined with guided-discovery lab sessions. At the other end of the scale, researchers tend to identify themselves with a theoretical perspective, probably because excellence in research requires a clear theoretical framework. I am convinced that the transfer from research to practice would be enhanced if researchers would integrate multiple perspectives into consistent pedagogical scenarios. I refer to these scenarios as **integrated learning scenarios**, since they integrate activities borrowed from multiple theories. These activities are not merely juxtaposed; instructional design is not about assembling a bit of everything. They are expected to form a consistent whole. The need for integration comes from the heterogeneity of activities. There would be no need for integrating

---

[2] With technologies such as tangible interfaces and paper-based interfaces, the frontier between what is computer-based or not is actually vanishing. Lecturing on a blackboard is not called "chalk-based teaching": the use of technology should not define the nature of the learning activity.

activities if they were similar or grounded in the same learning theory. The proposed modeling language describes how heterogeneous activities form a coherent structure.

Orchestration graphs describe pedagogical scenarios that can integrate heterogeneous activities. They specify the pedagogical and functional relationships between activities.

The integration of activities will be modeled by making the relationship between activities explicit. This relationship is two-fold. First, it is a pedagogical relationship—one activity activates cognitive and/or motivational processes that can be exploited by the next activity. Second, it is also an operational relationship—it describes how the data required to run the current activity are generated from the data that produced in the previous activity.

Orchestration graphs model the structure of the activity sequence, not the details of each activity. The reason for this structural focus is that the language is intended to model the orchestration process, not the learning mechanisms; this book does not propose a learning theory.

Let me now formalize the horizontal dimension of orchestration graphs, that is, the time dimension.

An activity a_i runs from starting time $t^s{}_i$ to the ending time $t^e{}_i$.
The duration of a_i is $d_i = t^e{}_i - t^s{}_i$

Time is counted as the number of time units (e.g., seconds, weeks) from the start of the scenario and is denoted by t_0, $t_0 = 0$

The time lag between the end of activity a_i and the beginning of a_j is denoted by $l_{ij} = t^s{}_j - t^e{}_i$

The scale of the horizontal axis is left to the designer: in the first example the scale unit is minutes, while in the second example the scale unit is weeks.

It may sound like overkill to formalize simple things such as duration or lag. The role of these parameters will become clearer later on, but time is clearly not a simple implementation issue—it's the main constraint in any educational or training institution, and maybe the most coercive constraint. Any formal educational system relies on an implicit or explicit contract that some skills will be acquired within a time budget. In the European Credit Transfer System (ECTS), the time budget is counted in credits (28–30 hours of work for one student). Efficient teachers permanently monitor time and balance what should be done with how much time is left. Underestimating the time budget (we rarely overestimate it), can lead to stress, dropping out, and to incomplete skills. We will also see that the lag between two activities has an influence on the effectiveness of the scenario: what a learner learned today might still be available on the same day, but 6 months later, it will need to be refreshed.

Point 2 Vertical axis

In the geometry example presented in Point 1, the activities are based on 3 different organizations of work. In a_1 and a_5, the teacher is speaking to the whole class and collecting feedback. In a_2, a_3, and a_4, learners work in groups, namely pairs. Finally, activities a_6 and a_7 are carried out individually. These changes in the social organization of activities constitute another salient aspect of pedagogical practices. Therefore, the social structure of the pedagogical scenario will be represented on the vertical axis of the model.

This dimension is treated as discrete; it is an ordered set of levels. Actually, since the word "level" is used for describing many things, I prefer the word "plane." This term comes from Vygostky (1994), who differentiated between the intra-individual plane (the space of cognitive processes), the inter-individual plane (where intersubjectivity occurs), and the social plane (shaped by its culture). The two examples presented in Point 1 stretch over three social planes.

- On the **individual** plane, students work on a task by themselves (e.g., read a text, write a summary, do an exercise).
- On the **team** plane, students work in small groups, typically made up of 2 or 3 students for problem solving, 4 to 6 students for projects, and 8 or more students for brainstorming or "problem-based learning" (PBL).[3] They are assigned a joint task to achieve; for example, to build a piece of software, to create a document, to invent an advertisement slogan, or to conduct an experiment. Within the team, individuals may be assigned different roles (see Point 8), but, at the end, they need to converge on a joint product.
- On the **class** plane, the activity involves all the students[4] in the class: they do activities such as listening to lectures, participating in discussions, presenting posters, or visiting a museum together. Class activities do not exclude individual interactions among learners (e.g., asking questions to the neighboring student or whispering), but the intended interactions are those that occur between the class and the teacher. The concept of class is used in a broad sense: it refers to the set of participants

---

[3] There is no maximal size for groups; one group can be made up of two groups of 50 people who collaborate on the same project. On the other hand, a group of 10 students that listens to a lecture independently from one another while the rest of the class does another activity, is more like a class, and hence called a "subclass" (see Point 7).

[4] Or a subset of the students, if the class is split into subclasses (see Point 7).

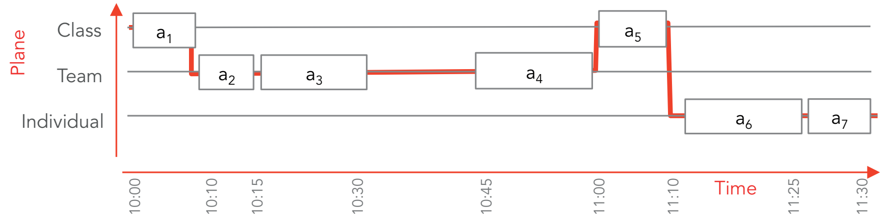


Figure 1.3 Adding the vertical structure to the model presented in Figure 1.2.

to a session where the scenario is run, independently from the physical location of participants.

This second dimension is independent from the time dimension (horizontal axis) and hence placed vertically on orchestration graphs, as illustrated in Figure 1.3.

In diverse educational practices, pedagogical scenarios may involve broader circles; namely actors who are not in the classroom or who are not MOOC participants. Therefore, orchestration graphs are extended with three more planes. To illustrate them, let's consider the scenario modeled by Figure 1.4. A biology teacher presents a lesson (a_1) and then sets up a session of individual exercises (a_2). After the second lecture (a_3), students go to the science lab where they run experiments in teams (a_4). With the data they collected, they write a report individually (a_5). On the following week, the class goes out to study the same phenomenon on a field trip (a_6). By using the pictures taken in the field and the data collected in the lab, students create a poster to be presented during open days (a_7), when parents visit the school. This poster will also be published on the school web site (a_8).

This example illustrates three higher social planes.

- On the **periphery** of the class, activities involve actors who do not "belong" to the class (e.g., students, teachers, assistants), but who nonetheless have

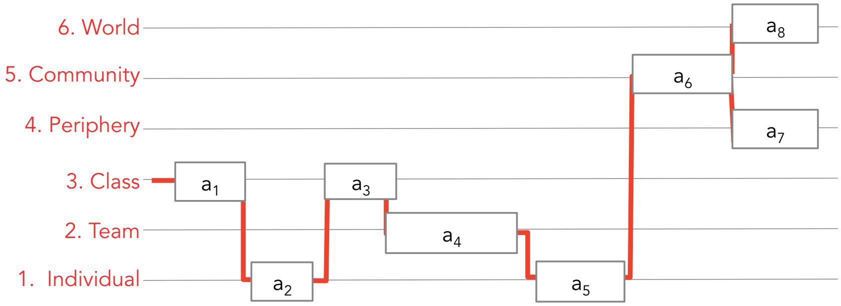


Figure 1.4 The biology scenario that expands to 6 planes.

a stable and explicit educational relationship with the class, such as school actors (the director, other teachers, other classes from the same school or classes in another school who have a partnership with that class), parents, and workplace supervisors. If the scenario is based on a computational environment, these people typically have a log-in to the online platform used by the class.

- On the **community** plane, activities engage temporary actors from the local community, such as a museum guide or a butcher invited to explain his profession, or from the broader community, such as an expert in astronomy who agrees to answer participants' questions. The "community" around a class is the set of people who have occasional interactions with the class, as they would have with any other class in the community, but do not have specific ties with that class.
- On the **world** plane, activities include disseminating information via the Internet, radio, publications, exhibitions, and forums. On this plane, interactions include feedback on online objects (e.g., "likes" or forum postings), but there is no intention to build a personal relationship between the class members and online anonymous actors.

Activities beyond the class circle did not wait for the invention of the Web, however. They existed many years before; for instance, in the pedagogical approach of Célestin Freinet, (1966) which included gardening activities (periphery plane), inter-school correspondence (community plane), and printing a school journal (world plane).

The vertical dimension of the model is a set of 6 social planes:

$\{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6\}$

A class is a set of students engaged in the same session of a pedagogical scenario.

The class of students is denoted by S. It corresponds to π_3. A student is denoted by s.

By integrating different social organizations into a common structure, orchestration graphs instantiate the didactic ecumenism I preached in the introduction. I explained that the individual-social dialectic has been central to educational debate and has led to practices perceived as mutually exclusive. This war is over: the brain is both a social machine from the software viewpoint and an individual device from the hardware viewpoint. Educational scenarios should therefore not hesitate to cross the boundaries between planes on a regular basis.

It is important to emphasize that **a plane does not describe the individual cognitive processes, but the social structure of activities**—which tasks are assigned to whom and who is supposed to interact with whom. During a lecture (π_3), students process the teacher's discourse on their own, which is an individual cognitive activity, but the main space of interaction is the class. Conversely, when a learner solves a problem alone, his cognition is shaped by language, therefore the cognition is partly social, but the plane of this activity is π_1 and not π_3. So, the social plane is an orchestration concept; it describes the organization of learning, not the cognitive processes.

Another important clarification is that **a plane does not correspond to a physical space or to a virtual space**. It is true that some spaces are more appropriate for some activities—a classroom is appropriate for π_3 and a library for π_1, a forum for π_3 versus a shared online document for π_2 or an email for π_1. In on-campus courses, the spatial constraints influence the vertical axis of the graph, that is, the social structure of the scenario. Educational practices have indeed been fossilized in formats such as a "2-hour lecture + 1-hour exercise," in which the former occurs in a lecture theatre (π_3) and the latter in the student's dorm room (π_2). As these room constraints are often associated with scheduling constraints, they may actually shape the scenario more than pedagogical considerations. To summarize, in daily practice, social planes are often associated with physical spaces. But orchestration graphs do not encapsulate this constraint. If activities from various planes could be conducted in the same space, the scenario would be more flexible (e.g., inserting a 15-minute exercise between two short lectures of 15 minutes).

Finally, I have to acknowledge that the proposed segmentation into 6 planes is a simplification and is arbitrary. First, graphs may adopt a more intricate social organization, for instance merging groups from two different classes for special interclass activities. The proposed modeling language simplifies this by associating a learning activity with a single plane. An activity that stretches over more planes, for instance, a team project in which each individual has specific subactivities, can be modeled (Point 7), but it has to be broken down into multiple activities for the sake of consistency with the rest of the model. The second simplification is that educational practices vary within a plane; for instance, although classes of 300 students have to be handled differently from classes of 30 students, orchestration graphs do not differentiate between them. **The notion of plane does not correspond to the notion of scale**: 1,000 students may do exercises individually (π_1), while 10 students may listen a lecture (π_3).

Nonetheless, this segmentation of the social space into 6 planes, despite being arbitrary, seems to match a broad range of practices. I have been using these levels for a decade, and they fitted well with the educational scenarios that I encountered during that period. The modeling language could probably be adapted to a different set of planes, as long as they are hierarchized. For the clarity of this book, henceforth I will stick to the six planes and, in particular, to the three lower planes.

Point 3 Topology

Since each activity can be placed horizontally (Point 1) and vertically (Point 2), orchestration graphs are laid out on a two-dimensional space. The horizontal dimension is time, represented from left to right, and the vertical axis is the set of social planes.

Each activity a_i: $(t^s{}_i, t^e{}_i, \pi_i)$ is represented by a rectangle in the space Time × Plane

The vertical position of the rectangle a_i is the plane π_i , $\pi_i \in \{ \pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6\}$

The height of the rectangle is fixed for full classes. Half height will be used for subclasses (Point 7).

Horizontally, the left side of the rectangle a_i is placed at $t^s{}_i$ and the rectangle length is d_i

The simplest example is the graph of a lecture that includes 2 activities: at π_3, the professor speaks for 45 minutes and at π_1, when students are invited to ask questions for the last 5 minutes. In the peer instruction method (Courch & Mazur, 2001), the lecturer interrupts his lecture (a_1) on a regular basis and presents a multiple-choice question (Figure 1.5). Each student answers individually with a clicker (a_2) and then has a few minutes to convince his neighbor (a_3). The pair answers again and all answers are collected on the teacher slides (through an “aggregation” operator—see Point 14). The teacher provides feedback and additional explanations (a_4); for instance, addressing the misconceptions behind the incorrect answers that he deliberately integrated into the quizzes. This sequence is repeated several times in a lesson.

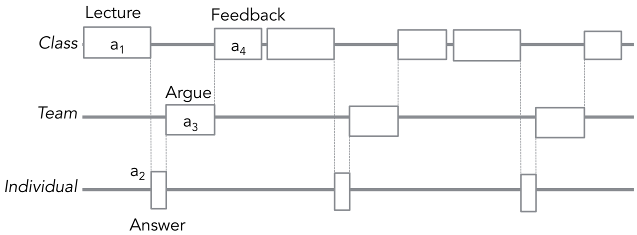


Figure 1.5 Graph showing the peer instruction method.

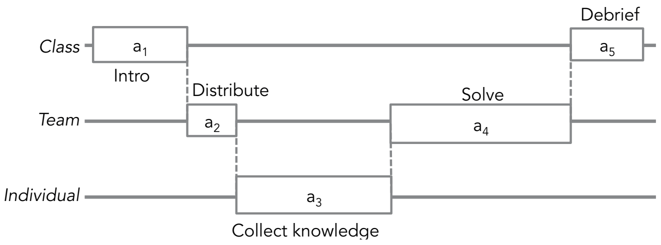


Figure 1.6 Graph showing problem-based learning.

In problem-based learning (Figure 1.6), the teacher starts by presenting a problem to the class (a_1); for instance, a medical case. Teams of 8 students meet and decide what they have to learn (a_2) in order to solve the problem. Each student addresses different aspects of the problem, collects knowledge (a_3) from the library or the web, and reports to the team, which then collectively elaborates a solution with a tutor (a_4). Ideally, the solutions feed the teacher's next lecture (a_5), which will elaborate on what students have learned while solving the problem.

An xMOOC (Figure 1.7) is generally a weekly sequence in which students watch a video and answer quizzes individually. After a few videos, they have to work on their weekly assignments. The forum maintains an interaction space

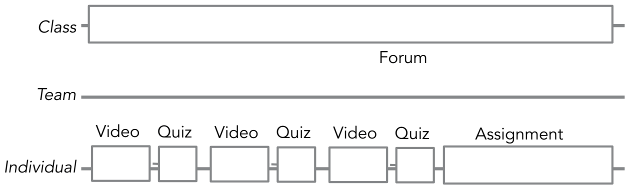


Figure 1.7 Graph showing a standard xMOOC.

at π_3 while conducting mostly individual activities at π_1. The forum activity is represented as being parallel to the main activity, since students may go back and forth between the forum and other activities. Parallel activities are described in Point 8.

Using a MOOC with on-campus students (Figure 1.8) leads to pedagogical scenarios that have been labeled "flipped class": students watch the lecture (videos) at home and meet the professor for sessions that allow richer interactions. The off-campus individual activities (video watching and quiz answering), may, for instance, be followed by a practice session[5] on campus, where

---

[5] Also referred to as 'recitation session'

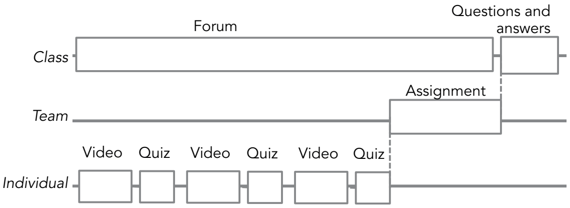


Figure 1.8 Graph of a flipped class based on an xMOOC.

students do their exercises under the supervision of teaching assistants. Later on, the teacher sets up a debriefing session whose contents are elaborated on the basis of the most common mistakes found in the assignments, as well as from the questions asked during the practice sections.

These examples describe standard practices. The goal is, of course, to go beyond these basic scenarios and to model richer scenarios. I will progressively enrich the modeling language throughout the book in order to represent richer pedagogical scenarios.

Let me now formalize the definition of an orchestration graph. Formally denoted by the letter G, a graph is defined by the pair (A,E) where A is the set of vertices, in this case activities, and E the set of edges[6] that connects two activities:

$G = (A,E)$ where $A = \{a_i\}$ for $i = 1, \ldots, n$ and $E = \{e_{ij} | a_i, a_j$ are the vertices of e_{ij}, for $i=1, \ldots, n-1, j=2, \ldots, n, i \neq j\}$

The graph vertices are the learning activities.

The edges that connect two activities represent the pedagogical and computational dependencies between these activities. The edge between a_i and a_j is denoted by e_{ij}.

An activity a_j is pedagogically dependent on previous activity a_i if a_i has a high probability of bringing the learner to a state that is necessary for a_j to have a high probability of success.

An activity a_j is computationally dependent on previous activity a_i if a_i generates data that are necessary for conducting a_j.

Graphically, an orchestration graph can be represented as in Figure 1.9.

We will see in Chapter 4 that pedagogical dependency actually relates two *states* rather than two activities—the state of the learner at the end of a_i and the state of the learner at the end of a_j. The graph of activity somehow doubles as a graph of learner states, one per activity. Let's keep things simple until then, though, and consider a simple graph with activities as edges.

---

[6] I will often consider cases where $j = i+1$.

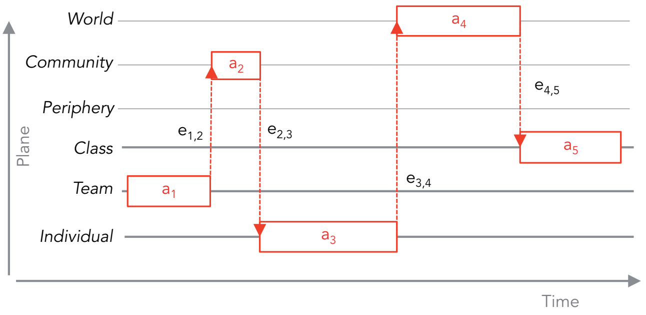


Figure 1.9 General structure of an orchestration graph.

If a_i is conditional to a_{i+1}, which is conditional to a_{i+2}, a dependency therefore exists between a_i and a_{i+2}. The edge between a_i and a_{i+2} is, however, not represented on the graph, because this would clutter the graph. I only represent edges between successive activities (plus looping edges—see Point 7). Other dependencies can be inferred from the transitivity of conditions.

An orchestration graph has several properties that will be examined throughout this book. Graph properties are usually captured by adjectives—an orchestration graph is a **weighted directed geometric graph**.

- It is a **weighted** graph because there is a probability factor associated with each edge. This weight estimates the dependency between a_i and a_j. A weight close to 1 indicates that completing a_i leads to a greater chance of success in a_j. A strength value of 0 would mean that it is not useful to complete a_i in order to complete a_j. We'll see in Chapter 4 how this weight is calculated. Through experience, a teacher knows that some learning activities can be shortened or skipped because they are not located on the critical path to learning. Conversely, for other activities, he knows that, even though it might take a long time, there is no point in moving to the next activity as long as students have not completely mastered the current activity. The edge weight will determine if an activity can be easily skipped or shortened.
- It is a **directed** graph because edge e_{ij} implies that a_i is conducted "before" a_j, as explained in Point 1. This "before" relationship will be interpreted as "completing a_i is a condition for beginning a_j."
- It is a **geometric** graph because the vertices are associated with a geometric configuration—a specific position in the space $\pi \times t$. This topology is further explained below.

If the reader is familiar with graphs, he may directly jump to the next point. Interpreting a graph consists in giving meaning to the edges between the graph vertices. I illustrate the role of the graph topology with a variety of graphs that are not orchestration graphs. The first example graph represents a simple ontology. In such a graph, the relationship between vertex V1 and vertex V2 often means something like "V1 is a subclass of V2." Hence, in Figure 1.10, the ontology represented is identical in both graphs; what matters is that V1 is connected to V2, but the relative position and space of V1 and V2 is irrelevant.

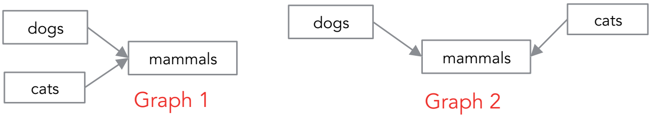


Figure 1.10 Graph with "is-a-subclass-of" links.

In other graphs, for instance, those resulting from social network analysis, the length of edges may convey a quantitative meaning such as the frequency of messages between two persons respectively represented by vertices V1 and V2. In a recommender system, the edge length could be the similarity of the objects purchased by these two individuals. For instance, in Figure 1.11, Graph 3 means that Savita and Dinesh like similar music, while Näel has divergent musical tastes. Conversely, Graph 4 and Graph 5 mean that Savita's musical choices are closer to Näel's. In these graphs, the relative position of objects conveys information. Therefore, Graphs 4 and 5 are equivalent as the distances are identical, with the absolute position of vertices being meaningless.

Finally, in other graphs (Figure 1.12), including orchestration graphs, the absolute position of vertices conveys information. For instance, the relative distance between key European capitals is identical in the symmetrical Graphs 6 and 7, but the former better represents the geographical reality of Europe. It does not mean that Graph 6 is correct and Graph 7 is incorrect. Both constitute a form of data visualization that is to be interpreted based on the semantics of its constituents, namely the existence of edges, their direction, their length (hence the relative position of the vertices), and the absolute position of vertices. In fact, any grammar rule for visualization can be inverted to convey another meaning. In Figure 1.12, Graph 8 illustrates an interesting

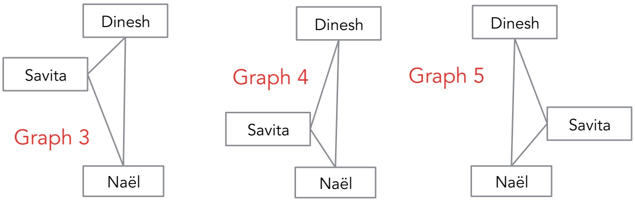


Figure 1.11 Graphs where distance represents quantitative relationships.

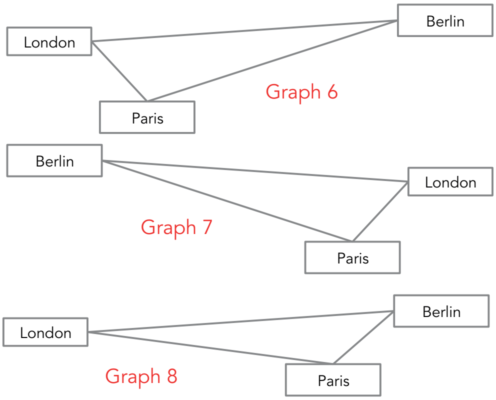


Figure 1.12 Graphs where absolute and relative positions matter.

visualization, by breaking the geographical mapping rule. It plays with the contrast between geographical locations and distances; if the length of edges in Graph 8 is inversely proportional to the number of bilateral visits among the 3 governments, this graph would nicely visualize what was called the "German-French couple" in the development of European Union policies.

This short detour aims to clarify that orchestration graphs have a specific geometry, in which the relative horizontal position indicates the time ordering and the absolute vertical position indicates the social structure of activities. There is a North and a South, an East and a West, a below and a between, a long and a short, and so on. Orchestration graphs are hence described as **geometric graphs**. An orchestration graph has a visual signature, a kind of *gestalt*. For instance, a glance at Figure 1.13 tells the reader that the scenario modeled in Graph 9 has been flipped for Graph 10—this is the famous flipped classroom often referred to in describing the use of MOOCs with on-campus students. The comparison between Graphs 10 and 11, if they have the same horizontal scale (i.e., the same time unit) intuitively reveals a difference in the level of integration between learning activities.

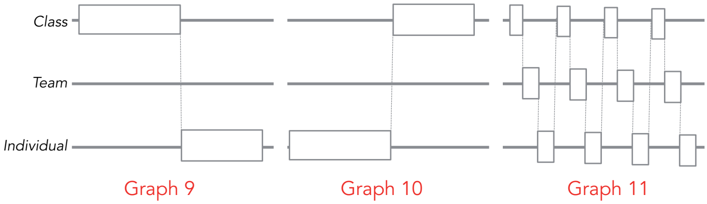


Figure 1.13 A *gestalt* reading of orchestration graphs.

Point 4 Activities

While points 1 to 3 of this chapter described the structure of orchestration graphs, points 4 to 6 will describe the elements placed on that structure.

The vertices or the nodes of orchestration graphs, that is, the atoms of the scenario, are learning activities. So far, I have used "learning activity" in its everyday meaning,[7] which is to perform a task such as listening to a lecture, watching a video, calculating a sum, or writing a summary. An activity describes what students are supposed to do, according to the scenario design, for instance: "*The students have to calculate the standard deviation of the data set collected in the previous activity and enter the value into the tool.*" This being a broad definition, I have refined it to the following five elements:

- An activity can last 2 seconds or 2 months. Is "*Writing a summary*" a single activity or should it be decomposed into subactivities such as "*Read the text*," "*Identify key ideas*," and "*Order key ideas*"? One way to answer this question is to define levels of granularity; for example, activity, subactivity, task, subtask, and action. However, any activity can be further decomposed into subactivities, recursively down to neuronal activation. The question therefore is: how far should an activity be decomposed in the modeling of a pedagogical scenario? What is worth modeling? Given the nature of orchestration graphs, it is not worth decomposing a task further if the teacher will not orchestrate its subtasks. For instance, if the activity is to write a summary, without any intervention regarding how the students formulate their summary, it can be modeled as a single activity. If, however, the teacher aims to separate it into two subactivities—the phase of extracting ideas from the text and the phase of arranging them into a new text—then it should be modeled as two activities.
- The way a specific student actually performs the activity (e.g., the values he introduced, the time taken, or the number of trials) is not included in the term "activity," but will be referred to as his behavior. Nor are the cognitive processes engaged by the learner to perform the predefined activity included in the concept of "activity." The term "learner activity" describes what students are asked to do. I could call it the "prescribed activity."
- Some activities performed by learners cannot strictly be qualified as "learning activities"; that is, students are not exactly learning anything, but they have to perform an activity required for continuing the learning scenario. This could be finding 2 peers to form a team, installing a piece of software on their computer, or passing a prerequisite test. Since these activities cannot be neglected from the viewpoint of orchestration, I have chosen to represent them explicitly in the graphs, in the same way as any learning

---

[7] I do not refer to the way this term is used in "activity theory."

activity. The term "learning activity" can therefore be considered as synonymous with "learner activity."

- Orchestration graphs describe the activities of the learners and not the activities of the teachers. This may sound like a contradiction to my focus on orchestration, but it is somehow similar to a musical score that describes what musicians have to play, but nonetheless determines what the conductor has to do. Teacher activities do somehow mirror the learners' activities and could be represented as another dimension (Prieto et al., 2011). This would however make the graph representation more complex, while I am trying to keep it simple.
- Some activities are more important than others. As we'll see in Chapter 2, certain activities are only there to prepare for a later activity or to further exploit what learners have done in a previous activity. In a pedagogical scenario, there are often one or a few activities that constitute the key element(s) of the scenario—its cornerstone or pivot point. The importance of such an activity will be expressed by the weight of the edges that connects the two activities (Point 5). In graph representations, this importance could be expressed by any graphical mark such as a thicker box line or a different color,[8] as illustrated in Figure 1.14.

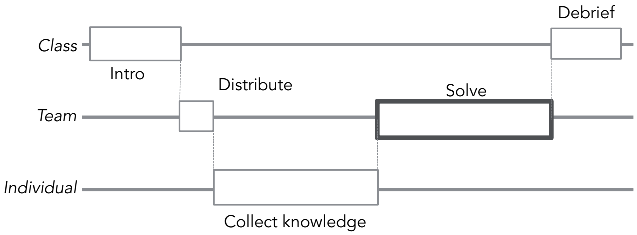


Figure 1.14 The same PBL graph as in Figure 1.6, with a visual marker for the cornerstone activity "*team problem solving.*"

Since activities are the core elements of orchestration graphs, I will now describe them in detail. Such a formal description may again seem useless at this stage, but it includes eight parameters whose usefulness will appear in the following chapters.

$$\boxed{V = \{a_i\} \mid a_i\text{: } t^s, t^e, \pi, \text{object, product, } \{c\}, \text{traces, } \{\text{metadata}\}}$$

---

[8] I will leave the choice of a graphical grammar for implementation time.

The first three parameters, t^s, t^e, and π, have already been described, so I will now describe the five other parameters.

Object(s): The object is the input to the activity—what students receive as part of their instructions and have to process during the activity. I denote objects by square brackets, as in the following examples: "*Watch video [v35.avi]*," "*Invent 5 sentences that includes the structure [despite of]*," and "*Calculate the standard deviation from the data in [file.ex32.txt]*." The object can be generated by an operator; for example, "*Compare the air pressure of* [X] *and [Y]*," where the system selects locations *X* and *Y* in real time as two points on the planet that have the same altitude, but different weather conditions. An activity can have several objects as an input.

Product(s): Some learning activities produce a result, an answer, or a solution—the output of the activity. The term "product" does not refer to the cognitive traces left by the activity (i.e., the learning effects), but to the answer or solution. An activity such as "*Memorize these 5 city names*" has no product, since there is no response or solution to be produced. In computer-based education, as well as in classrooms, the products of activities are not just volatile objects for evaluating an activity. They become persistent (data) objects that can be stored and sometimes reused in activities that occur later in the graph. The pedagogical exploitation of the "emerging objects," as Ulrich Hoppe called them (Wichmann et al., 2010), is a distinctive feature of modern learning technologies.

Competencies {ci}: Some learning activities target specific competencies, sometimes a single one, and often several ones. Some of them are just logistical activities and are not related to any skill. This parameter is developed in Point 6.

Traces: Some activities such as reading a document do not generate a product, as defined earlier, but do nonetheless leave traces of the learner's activity; for example, navigation acts during a video (the sequence of clicks on the "pause," "play," or "forward" buttons), the number of zoom in/out actions while reading a map, or the gaze patterns of learners recorded with an eye tracker. Keeping track of traces is an added value of digital tools in education—it allows teachers to have a closer look at what is happening to learners. Let me emphasize how these learning technologies have evolved—they have developed from analyzing only activity products (the answers) to processing the many tiny digital footprints left while producing the answers. The same evolution (basically from deductive to inductive science) characterizes many disciplines and is now influencing education.

Metadata: In digital documents, metadata refer to information pieces—mostly keywords—that describe the file: who created it, in which language, with which tool, and so on. In orchestration graphs, metadata document the activities. The simplest example is to give a name to an activity, such as, "*Presenting Kolmogorov-Smirnov*" or "*Part II: Causal Adverbs*." Other metadata can indicate the copyright status of a document, its time validity, its level of difficulty, or the target audience. The function of educational metadata is to provide teachers/systems with information useful for deciding if an educational resource (e.g., a document or an activity) is relevant for the learning goals of the scenario to be designed, and if it is, how it can be exploited. In orchestration graphs, metadata should also include the **design rationale** of the activity, that is, a few lines of text that explain why an activity has been created, referring to intrinsic or extrinsic constraints that justify design choices (Point 35). If the teacher who is conducting the pedagogical scenario is not its author, he will need this information for deciding which modifications he can carry out on the scenario. Metadata are the key elements in sharing educational resources among teachers, or for allowing learners to identify the resources they need. Educational metadata have been the object of attention for two decades. Some standards have been developed in the learning technologies community for effectively **sharing** educational resources through repositories of "open educational resources" (OER). Allow me a short parenthesis about OERs.

> *The idea of OER is close to the MOOC idea, but with a lower grain size; what is shared is a set of examples, questions, or exercises, rather than courses. OER repositories are actually more "open" than MOOCs, in the sense of "open source" (everyone may contribute). Intensive and clever work has been done on these standards; we should not reinvent the wheel and reuse them to enrich the description of activities. Unfortunately, OERs haven't reached a very large audience so far. Several factors may explain this low development of OERs. Giving access to one's own teaching material is feasible. Even if the complexity of some metadata schemes can be a black hole for teachers' energy, there are ways to automatically extract some of the metadata. Indeed, the difficulty is less in letting others use one's material than reusing the material developed by others. Borrowing educational material requires trust and the ability to address the contents in the same way the resource creator saw them, with the same level of granularity, the same approach, and maybe the same scientific notation. The work on educational metadata has perhaps come too early in the development of learning technologies. The current rise of online education could lead to its renaissance.*

The activity parameters relate the activity descriptions to various areas of learning technologies—*metadata* connect activities with repositories on educational resources, *traces* connect activities with learning analytics (Chapter 5), while *objects*, *products*, and *competencies* will be explained hereafter in relation to mastery learning. Describing a learning activity in terms of input and output sounds a bit like good old cybernetics. I don't do it because of a sense of nostalgia, but for the two following reasons. First, the structure of data before and after an activity is an essential element of workflow, which in turn is the backbone of orchestration graphs (see Chapter 3). We'll see that the same activity a_i may trigger different cognitive processes depending on the way data have been processed before a_i; for instance, finding a solution to a_2 could either lead to consensus or conflict, depending on whether team members received the same or conflicting evidence. The second reason for specifying the input and output of an activity is in order to elaborate a description of activities that is precise enough to situate an activity into existing classifications of cognitive activities. For instance, if the object is a concept definition and the product is a classified instance, the cognitive process is deductive. Conversely, if the object is a set of examples and the product is the concept definition, the activity is inductive. Bloom et al. (1956) and Guilford (1965) have developed domain-independent classifications of mental operations. D'Hainaut (1985) defined 6 categories of cognitive activities based on the relationship between the object and the product. These taxonomies are more than theoretical instruments; they play a key role in instructional design—it is "hygienic" to analyze the level of the designed activities within a taxonomy. By distinguishing between low-level and high-level activities, cognitive taxonomies reduce the risk of a disease that is endemic to education—low-level activities, especially reproduction/memory tasks and simple application tasks, being overrepresented in educational practices (probably because they are easier to design and evaluate than higher-level activities).

These taxonomies originate from an effort to make instructional design more rational, and more effective overall. The activity parameters connect orchestration graphs with an educational theory developed in the 1960s under the name "mastery learning" (Bloom and Carroll, 1971). This theory is not so popular anymore, maybe due to its behaviorist roots or to its exaggerated fragmentation of learning activities into small steps. It would, however, be a mistake for today's learning technologies to neglect a theory that has developed highly effective methods. This effectiveness is principally related to a close control of individual progression—checking that each step has been mastered before moving to the next one. In this theory, pedagogical scenarios are enriched with 4 types of tests—they are not, properly speaking, "learning activities," but "learner activities."

- At the very beginning of a graph, a **prerequisite test** verifies if the learner possesses the skills necessary to succeed in the graph activities with reasonable probability. What can be negatively perceived as an entry filter is actually respectful of the learner's time; it would be a wasted effort to let someone invest hours in a pedagogical scenario if he is certain to fail.
- At the beginning, a **pretest** also verifies if, by any chance, the learner already possesses some of the skills he is supposed to acquire in the graph. This increases efficiency—namely, minimizes learning time—by allowing the learner to skip parts of the graph that correspond to the skills he already masters. Moreover, this is useful for calculating the learning gain, which establishes how much knowledge the learner gained between the

pretest and the post-test. Additionally, it allows the use of the pretest as a covariate when analyzing post-test scores.

- In mastery learning, a graph is often structured into modules, with a module being simply a segment of the graph. In this case, **intermediate tests** measure the skills after each module. They serve both as a post-test for that module and as a test of prerequisites for the next module (if there is a prerequisite relationship between them). It does not have to look like a test; it can simply be one of the activities of the graph.
- At the end of a graph, a **post-test** or final assessment verifies if the learner has acquired the skills he was supposed to acquire. This is the exam.

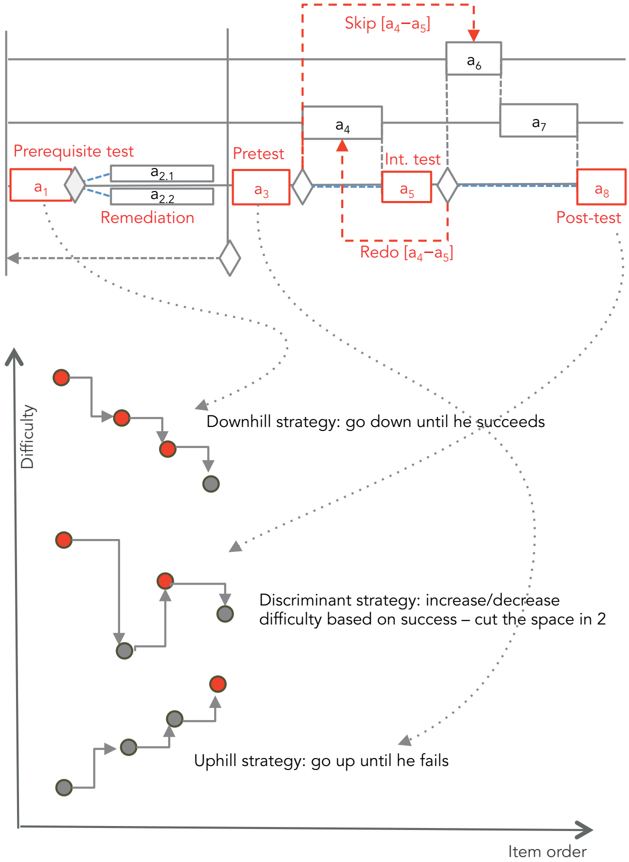

Skip [a_4–a_5]
a_6
a_4
a_7
Prerequisite test
Pretest
Int. test
a_1
$a_{2.1}$
$a_{2.2}$
a_3
a_5
a_8
Remediation
Post-test
Redo [a_4–a_5]
Difficulty
Downhill strategy: go down until he succeeds
Discriminant strategy: increase/decrease difficulty based on success – cut the space in 2
Uphill strategy: go up until he fails
Item order

Figure 1.15 Tests shown in the graph (top) and testing strategies (bottom). After the prerequisite test (a_1), learners may get a specific remediation activities test ($a_{2.1}$ or $a_{2.2.}$) and then take the test again. After the pretest, they may skip some activities. In the strategy graph, red dots are items that the student failed.

One could wonder when students learn, if they spend most of their time sitting tests. Fortunately, adaptive testing strategies reduce the time spent in testing. They are illustrated in Figure 1.15. In a prerequisite test, the learner is supposed to have already acquired the measured skills. Therefore, complex items are tested first; if the learner succeeds, the test may stop, but if he fails, subskills have to be tested, recursively through the map of skills (Point 6). Conversely, the pretest measures skills that the learner is not supposed to have, since he joined in order to learn them. Therefore, basic skills are measured first. If the learner does not master skill X, there is no point in testing skill Y, if X is a prerequisite of Y. If there is no hypothesis about the student's level, the optimal testing strategy follows a sorting algorithm: first select an item in the middle of the difficulty scale; if he succeeds/fails, select an item at the middle of the upper/lower half of the scale; and so forth.

Point 5 Edges

In orchestration graphs, activities are connected by links or edges. An edge between two activities represents a double dependency between these two activities.

- The conditional dependency between two activities is a measure of how much the former is a condition for the latter. If the learner failed an activity, does he have a chance of succeeding at the next one? If time is too short, can an activity be skipped without endangering the whole scenario? The set of conditional dependencies between activities constitutes a probability network and is the core of the stochastic approach that will be developed in Chapter 4. This dependency will be modeled as a transition matrix (Point 22).
- The data dependency between two activities describes how data are transferred between them. For instance, in a peer-critiquing scenario, a student writes a text (a_1) that another student comments on (a_2), and then the original author revises his text (a_3). To operate this scenario, the product of a_i (the text) becomes the object of a_2, and the product of a_2 (comments) becomes the object of a_3. The transformation of data between activities constitutes a workflow, described in Chapter 3 as a sequence of operators. Workflows enable a scaling up of learning activities, since they automatically perform the manipulations that a teacher would have performed manually.

Two activities can be pedagogically dependent without being functionally dependent, for instance, if learners acquire a skill that is a prerequisite for the next activity, but there is no data transfer between these two activities. Inversely, if the first activity involves entering the references of the location where students will collect soil measurements, these data are necessary for running the next activity (producing a map with all measurements), but not cognitively required for reading the map. These two dependencies are conceptually so different that I could actually represent them as double edges, but the graphs would then become unreadable. Instead, these two dependencies are stored as different parameters associated with an edge, as explained hereafter. Allow me to stress this duality, which could be confusing for the reader; a graph is a structure that plays two different, but complementary, functions— an operational function as a workflow and a descriptive function as a stochastic model. The superposition of these two functions is essential for the remainder of this book.

The edges have a further, third function; in some cases, an activity can connect to two subsequent activities, and the decision criteria to branch to one or the other is also stored in the edge, as control structures (Point 7). These 3 functions of edges are summarized in Figure 1.16.

An edge is defined by its 2 connected vertices, plus the 5 parameters described below. I will outline all 5 parameters now, despite the fact that some of them will only become relevant in later points.

We define the set of edges as $E = \{ e_{ij} \}$ where e_{ij} is a tuple (a_i, a_j, {operators}, {controls}, label, weight, elasticity).

Operators: The parameter {operators} is a set of operators that describes the data dependencies—how data structures generated by learners when performing a_i are transformed into data structures necessary for conducting a_j. In the previous peer-critiquing example, the operator stores all texts written in a_1 and distributes them to students in a_2. Simply stated, an operator is a little piece of software that stores and processes data. Another example is a

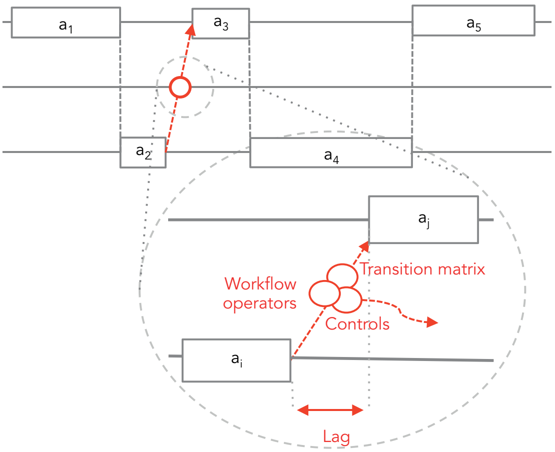


Figure 1.16 A zoom into the edges of an orchestration graph.

scenario in which the learner who finds a solution to activity a_1 is then asked to collaborate in a_2 with a student who did not find the same solution in $a_{1.}$ Chapter 3 presents a variety of operators and explains how they form a workflow.

Controls: The parameter {controls} allows richer graph structures, with loops and forks. These controls are described in Point 7. If an edge ends with several possible activities, these controls decide which activity will be selected next. Edges with multiple ends a_j, a_k, ... will be modeled in a slightly different way.

In that case $E = \{ e_{ij}\}$ contains tuples e_{ij} of the form (a_i, {a_j, a_k,...}, {operators}, {controls}, label, weight, elasticity).

The next 3 parameters (label, weight, and elasticity) define the conditional dependencies between two activities. Edges connect consecutive activities (j=i+1), but conditions are transitive; if a_i is a condition for a_j and a_j for a_k, the edges e_{ij} and e_{jk} implicitly contain e_{ik}. As mentioned earlier, I do not represent e_{ik}, which keeps the graph readable.

Label: The label defines the pedagogical relationship between two activities. Why is activity a_i a condition to run activity a_j: is it a prerequisite, a motivation trick, or an illustration? The edges encapsulate the key pedagogical ideas behind the pedagogical scenarios. I will therefore devote the next chapter to these labels.

Weight: The weight of edge e_{ij} is denoted by ω_{ij} and expresses the importance of the conditional relationship between a_i and a_j, that is, how much a student's performance in a_i will determine his performance in a_j. Two activities related by a heavy weight edge are strongly interdependent. One example of strong dependency is when a_i is a cognitive prerequisite to a_j, that is, if the learner cannot learn the contents of a_j without having acquired the contents of a_i. Another example is the previously mentioned peer-critiquing scenario; a_2 cannot be conducted without a_1 since the learner cannot comment on a text that has not been written. Edges with a low weight correspond to opposite cases, where a_i can be "nice to have," but not strictly necessary for a_j. For instance, activity a_1 is sometimes introduced to motivate students to do a_2 by showing them how the skills to be acquired in a_2 are useful in real-world problems. If a_1 has to be skipped, the chances that students are interested in a_2 will be lowered, but not reduced to nil; a_1 can still be conducted with reasonable chances of success. Another example is if a_1 presents the demonstration of a statistical theorem; understanding the demonstration is great for shaping students' minds, but not strictly necessary for applying statistical methods. The weight of edges influences the *flexibility* of a graph: if it is necessary to skip a_i, for instance due to time pressure, it will be less detrimental to do so if ω_{ij} is low.

The weight of an edge could be represented on the graph by the thickness of the line that connects two activities, by a color intensity scale, or by a numerical value displayed next to the edge. I will leave this detail to the future implementation of this modelling language into a platform.

Let's consider a problem-solving activity in which the solution is either correct or incorrect. The weight can then be expressed in Bayesian terms as the probability of succeeding a_j relies on the condition of having succeeded a_i.

$\omega_{ij} = p\,(A_j \mid A_i)$ where A_i is the event in which "the learner found the correct solution to a_i"

In many cases, however, the result of an activity is more subtle than simply "correct/incorrect." If it is a metric value, for example, a score between 0 and 20, then a correlation coefficient could be used, even though the relation between the success at two activities may not be linear. In other cases, the result of an activity will be a set of discrete states. In this case, the probability will not be a single value, but a probability matrix that represents the transition between each possible state at the end of a_i and at the end of $a_{j.}$ We will see in Chapter 4 how the information contained in such a transition matrix can be summarized by a single numerical value, which will be used as the edge weight.

Elasticity: The effect of a_i on a_j often fades out with time (e.g., as the lag between two activities increases); learners might forget what they learned in a previous activity, and the motivation created at the beginning of a lesson by presenting an interesting application example does not last forever. In other words, the weight between two edges can be strong if lag is short, but may decrease as lag increases. Therefore, I use the metaphor of elasticity, in its everyday meaning. If we stretch a rubber band, it will lose thickness, and if one continues stretching, it will eventually break. Similarly, the conditional relationship between two activities will decrease as the time lag between them increases. This concept is refined in Point 25.

The *elasticity* of an edge e_{ij} is a function that defines how ω_{ij} decreases as the time lag l_{ij} increases.

Point 6 Skills and competencies

The goal of a teacher who designs a pedagogical scenario is that students acquire knowledge, competencies, or skills. These target skills are called pedagogical goals, objective outcomes, or learning outcomes. One often dissociates the contents or knowledge to be acquired (e.g., "*What is the definition of the 'variance of a distribution'*"), from the skills (e.g., being able to calculate the variance), and from competencies (e.g., being able to argue with data-based arguments). The latter are also called "transversal skills," as they apply across domains.[9] D'Hainaut (1983) dissociated the *ends* of an educational system (e.g., to train civil engineers), the intermediate *goals* (e.g., to estimate risks), and the course-specific *objectives* (e.g., to combine uncertainty measures along 3 dimensions of a oil reservoir). I will not enter into the terminological details concerning the differences between all the terms used in the last sentences for describing the same lexical field, not because these differences are not important, but because this debate conceals the message contained within. Please allow me a detour to defend a teleological approach to education.

When a civil engineer has to build a bridge, he does not first decide that the bridge will have four arches because he and his father have always built bridges with four arches. He first analyzes the situation in terms of who is going from where to where. However, in contrast, we do instead tend to teach the way we have been taught. A rational approach to education—which this book argues for—starts from the endpoint: what should students be able to do at the end of their course? It does not really matter if "*What the students are able to do*" is called a skill, a competency, a learning outcome, or a pedagogical objective. It does not really matter if the learning outcomes are formulated according to a specific syntax. The point is that a pedagogical scenario is built in reverse. Defining the final skills is not a pointless pedagogical sophistication or a bureaucratic chore. It is about designing education with an engineering mind.

Let's define C as the set of content knowledge, skills, and competencies to be acquired. Even if this is only presented as the 6th point in this book, the first step in instructional engineering is to define C. The second step is to analyze C, that is, to determine the components of C. In general, C is divided into a set of skills $\{c_1, c_2, \ldots, c_m\}$. The number of skills, m, may differ from the number of activities in the graph. Once C as been decomposed, different subsets of $\{c_i\}$ are associated with different activities. Figure 1.17 illustrates two methods for task decomposition: for procedural skills, C is broken down into an algorithm (on the left), while declarative knowledge is instead depicted as a graph of concepts or a semantic network (on the right).

The decomposition of C raises the same issue as the definition of activities; at what point should we stop decomposing C, that is, consider the subskills in c_k as an atom? The designer may stop decomposition in the following cases:

- If c_k can be considered as a prerequisite for the average student. This should be verified at course entry (see the prerequisite test, Point 4).
- If c_k is a cognitive black box, that is, if it cannot be decomposed into meaningful subactivities. This is the case for "*Capital (Belgium) = Brussels*" or for "$4 \times 5 = 20$."

---

[9] But they are often not entirely domain-independent, as revealed by theories of situated cognition.

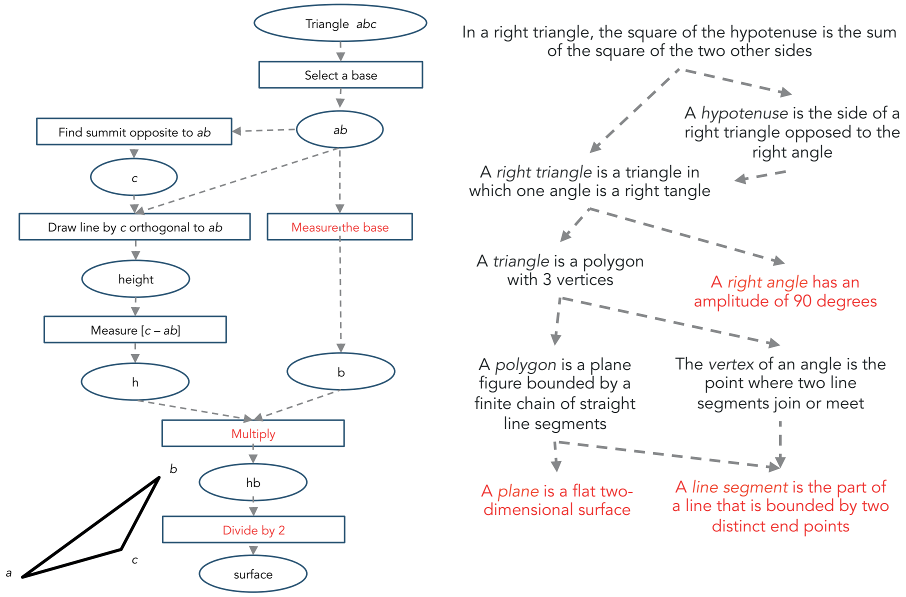

Triangle abc
Select a base
ab
Find summit opposite to ab
c
Draw line by c orthogonal to ab
Measure the base
height
Measure [c – ab]
h
b
Multiply
hb
Divide by 2
surface
b
a
c
In a right triangle, the square of the hypotenuse is the sum of the square of the two other sides
A hypotenuse is the side of a right triangle opposed to the right angle
A right triangle is a triangle in which one angle is a right tangle
A triangle is a polygon with 3 vertices
A right angle has an amplitude of 90 degrees
A polygon is a plane figure bounded by a finite chain of straight line segments
The vertex of an angle is the point where two line segments join or meet
A plane is a flat two-dimensional surface
A line segment is the part of a line that is bounded by two distinct end points

Figure 1.17 Left: A "mathetic analysis" (D'Hainaut, 1983) draws the algorithm of the activity, that is, given the input presented to the learner, what are the points he must perform in order to calculate the expected output. Each element may be decomposed into sub-elements. Red elements are prerequisites and therefore not further decomposed. Right: A semantic analysis (D'Hainaut, 1983) consists in building a semantic network. Red elements are prerequisites and therefore not further decomposed.

- If c_k is a didactic black box, that is, if there is no intervention in the learners activity during c_i. For instance, if a graph includes an argumentation task, two models are possible. If the designer defines argumentation roles (Weinberger et al., 2007), one learner proposing claims to be refuted by his peer, these roles should be modeled explicitly as subactivities (see Point 7). Conversely, if argumentation is free, it should be modeled as a single activity in the graph. This won't prevent the teacher from regulating it. A MOOC video is therefore considered as a single activity, even if it covers several topics.
- If an activity related to c_k does not provide analytics. For instance, if students have to write an essay, should the grammatical subskills be modeled? If students upload a document in which most grammatical mistakes have been automatically corrected, grammatical skills won't be measurable. If they use specific text-entry software that records all edits, they could be modeled.

In behaviorist pedagogy, there is direct mapping between $\{c_i\}$, a set of subskills, and $\{a_i\}$, a set of graph activities—a_1 is associated with c_1, a_2 with c_2, a_3

with c_3, and so on. There can also be incremental mapping: a_1 is associated with c_1, a_2 with c_1+c_2, a_3 with $c_1+c_2+c_3$, and so on. However, this is not a general rule in instructional design—any kind of mapping is possible. Moreover, an orchestration graph may include activities that do not target specific skills such as motivation and logistical activities, as well as activities that target multiple skills. In other words, activity graphs and skills graphs are connected, but they are not isomorphic.[10]

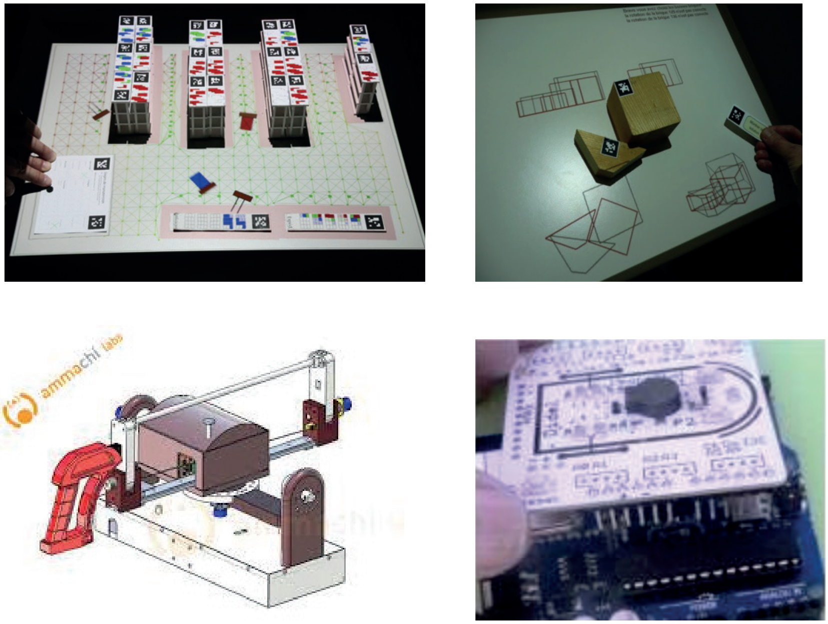


Figure 1.18 Enriching online education with physical interactions.

Example 1 (top left): A tangible interface for a tangible simulation—apprentices place small plastic shelves on the table, and the augmented reality system overlays the warehouse activity, such as forklift movements (Zufferey et al., 2009).

Example 2 (top right): A tangible interface for carpenters—apprentices develop spatial reasoning skills by moving wooden blocks, and the augmented reality system overlays their three orthogonal projections (Cuendet et al., 2013).

Example 3 (bottom left): Haptic saw—apprentices have to cut a virtual piece of wood on a computer, with the same force feedback as with a real saw, but completed with digital feedback on the saw movements (Akshay et al., 2013).

Example 4 (bottom right): Microcontroller toolkit—in this MOOC[10], the assignment consists of physically assembling a microcontroller. The assignment is evaluated by connecting the microcontroller to a computer. This toolkit is sold for 20 Swiss francs.

---

[10] https://www.coursera.org/course/microcontroleurs

When talking specifically about MOOCs, there is a fear that some skills cannot be taught via MOOCs, because they cannot be assessed online.

- The first limitation concerns high-order thinking skills. Universities are not only expected to teach concepts, laws, and procedures, but also high-order skills, such as critical thinking, rigorous reasoning, creativity, collaboration, information retrieval, and project management. The evaluation of these skills seems a priori not to be compatible with multiple-choice questionnaires and only partly compatible with peer grading. It requires rather ill-defined, open, multi-steps tasks that are not easy to evaluate on a large scale. This is a concern that needs to be taken into consideration if MOOCs are to be used for entire curricula. One solution is to ask participants, after completing 3 or 4 MOOCs, to elaborate a capstone project, that is, a project that integrates the skills acquired in each MOOC, but new ways of following large numbers of projects nevertheless have to be invented.
- The second limitation in regards to the range of skills comes from the interface between the learner and the digital environment; beyond the keyboard, mouse, or touch surface, do MOOCs allow for the teaching of physical skills, such as professional gestures? In actual fact, the development of tangible interfaces, augmented reality, and haptic devices extends the capacity of online education to include physical skills, as illustrated by the examples in Figure 1.18.

Point 7 Control structures

Up to now, the presented graphs have been restricted to a linear sequence of activities. Two more structures are necessary to broaden the range of pedagogical scenarios that can be described with orchestration graphs—loops and conditions. It is, of course, possible to add other structures such as functional calls, that is, embedding subgraphs within graphs. However, as often mentioned, I strive to keep orchestration graphs simple. My experience is that sophisticated scenarios are hard to orchestrate for teachers. In addition, learners also suffer from intricate scenarios, since they also have to learn what to do, when to do it, and how to do it. This was evident to us when we tested a scenario where the complexity of the graph absorbed most of the attention and energy that the learners should normally devote to learning, and teachers to teaching (Berger et al., 2001). In summary, I have kept this modeling language simple, but, by adding loops and conditions, it becomes possible to model richer graphs.

Control structures are associated with edges in order to separate activities, as independent software components, from the control structures, as other independent software components (namely a set of “if-then” conditions).

Branching: A key decision to be taken when a graph is executed is "*What to do next?*" A branching operator implements the choice between two or more activities. In Figure 1.19, after a short lecture (a_1) that reminds students of the theory, the teacher asks them to solve exercises (a_2). The branching condition will be based on their score in a_2; if the class is working fine globally, the teacher will resume the lecture (a_3), but if the test results were not sufficient, the teacher will propose more exercises (a_4).

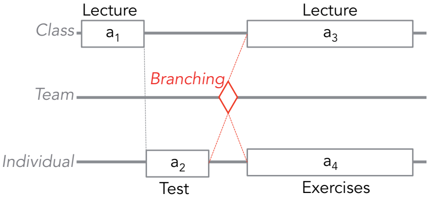


Figure 1.19 A branching structure.

Who actually makes branching decisions? If the decision criteria are able to be calculated, the system can decide; if a more global grasp of the context is required, the teacher can decide; the teacher can also make decisions based on analytics generated by the system, and finally, learners themselves can make decisions. The ratio between the number of decisions made by the system/teacher and the number of decisions made by learners is called the "balance of control." Along with the individual-social pendulum (see Introduction), the notion of balance of control has generated a passionate debate in learning technologies. Early computer-assisted instruction systems gave low control to learners, while educational microworlds such as LOGO gave them full control. In a nutshell, the former were relatively boring and the latter rather ineffective. This generated a strong argument between, respectively, the quest for effectiveness and the goal of developing user autonomy. The same difference exists today between xMOOCs and cMOOCs; the graph is rather rigid in the former, while full control is left to learners in the latter. One would expect mixed control, where both the system and the learners share decisions, to be optimal. This optimum is, however, rather difficult to implement, as learners must have the freedom to chose their activities, and the system must be able to monitor their uncontrolled activity in order to make relevant suggestions.

Looping: Some sub-sequences of the graph sometimes have to be repeated several times. They constitute a loop, defined by three arguments: when does it start and end,and how many times is it repeated (e.g., "*5*" or "*Until time is over*"). In some cases, certain activities are only executed on some iterations (e.g., the first time, the last time).

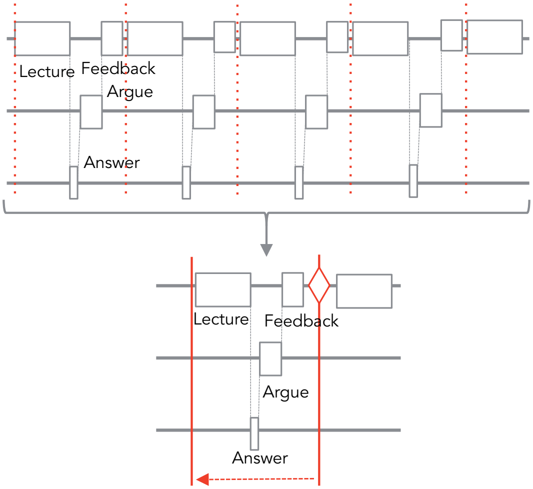


Figure 1.20 The peer instruction graph (top) from Point 3 is presented as a long sequence and is then modeled as a simple loop—question-answer-argue-feedback (bottom).

Point 8 Parallelism

In a graph, parallel activities are activities that occupy the same time slice of the lesson,[11] on the same plane or on different planes. These parallel activities can be independent from each other or interdependent. Activities are independent if they satisfy two criteria. First, each learner participates in only one activity at the same time. Second, the actions of learners in one activity do not influence the actions of other learners in other activities.[12] In other cases (which will be illustrated hereafter), the activities are called interdependent.

I will firstly describe independent activities. Typically, they are conducted when the class is partitioned into subclasses, often 2 to 4. The subclasses form a partition of the class in the mathematical sense of the word “partition”; each student belongs to one and only one subclasss. A subclass is different from a team; teams (π_2) have to achieve a task together, while in a subclass, students might all do individual exercises without interacting with each other (π_1), or they might do different types of team projects (π_2), or they might attend different lectures (π_3).

One reason for creating subclasses is logistical constraints (Point 35), for instance, the need to reduce the number of students per activity. This is

---

[11] I use the term “parallel” rather than “synchronous” to avoid confusion with synchronous communication tools used in online education.

[12] If, for example, each team within the class conducted an urbanism study where they analyze the same concepts but for a different city, I would model all their projects as a single activity, because even if the object varies, the activity is the same.

typically due to physical constraints, such as the room size or the number of devices. This occurs in science labs, where the number of devices often defines the subclass size (Figure 1.21). Logistical constraints also apply to online education, for instance, when there is limited access to resources such as remotely controlled experiments, a time-consuming automatic grader, or human coaching.

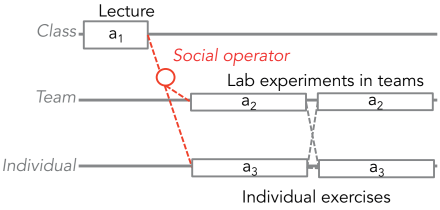


Figure 1.21 Parallel activities for logistical reasons. After the lecture, the class is split into 2 subclasses. While one subclass does lab activities (a_2), another subclass does individual exercises (a_3), and then they switch. I represent subclasses with thinner rectangles to give a visual impression that they hold fewer students than the whole class. The red circle represents the social operator used to split the class into 2 subclasses (see Point 16).

A second reason for splitting a class into smaller subclasses is the management of verbal interactions; it is difficult to manage seminars, case studies, or language games with more than 25 students.

A third reason for parallelism is in order to operate adaptive instruction (Figure 1.22), that is, to differentiate the activities assigned to different subclasses.[13] A subclass gathers students who have some characteristics in com-

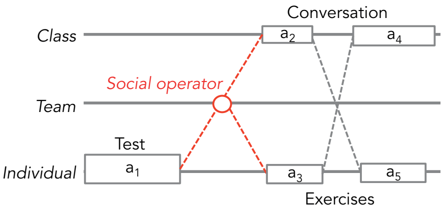


Figure 1.22 Parallel independent activities for subclasses with different levels. The graph illustrates a German lesson. All students start with exercises on sentence construction (a_1), which are automatically graded. In the edge $e_{1,2}$, the social operator forms two subclasses, based on a_1 scores. Then, the best students participate in dialogue activities with the teacher (a_2) while the others continue individual exercises (a_3). It is easier for the teacher to manage dialogue among students who have a homogenous level of dialogue skills in German. After a break, the groups switch activities.

---

[13] These homogenous subclasses are often named “level group”; I do not use this name since it would lead to confusion with group/team activities (π_2).

mon; for example, "*Those who failed the pretest,*" "*Those who had never studied biology before,*" and "*Native speakers*." To form subclasses based on student similarities or differences, a graph applies social operators, described in Chapter 3. The basic idea behind subclasses is to reduce class heterogeneity; each subclass is less heterogeneous than the main class.

These ways of orchestrating class activities illustrate a first type of parallelism—namely between activities conducted by different students independently from each other. A second type of parallelism occurs between activities that are not independent, that is, they remain coupled despite the fact they occur at different planes. They create dependencies between these planes. A common example of interdependent activities is when students are assigned different roles for a team activity; roles differentiate the activity of each individual (π_1) within the team effort to achieve the task (π_2). In project-based learning, a team is often structured around several roles, such as leader, notes taker, timekeeper, information seeker, and summarizer. Roles can even be used within pairs, for instance, when one learner is presenting a claim and the second one is critiquing it. Roles can be domain-specific, as in the example represented in Figure 1.23. In a lesson on earthquakes, my colleague A. Parriaux (2009) introduced the problem (a_1): "*How to avoid a major earthquake in San Francisco? In Denver, the city injects water in the fault. This increase of fluidity facilitates the friction between tectonic plates, which creates tiny earthquakes. These small earthquakes reduce the tectonic tensions, which could prevent a main earthquake. Should San Francisco do the same?*" Each team comprises 4 roles: the mayor of San Francisco (a_{2a}), a seismology expert (a_{2b}), a security officer (a_{2c}), and an insurance agent (a_{2d}). These four roles are represented as individual activities. After a while, students are redistributed into "expert groups"; those playing the mayor's role discuss their views with the mayors of other teams (a_{3a}), and so forth. In fact, the sequence a_2–a_6 is repeated several times. Finally, teams make their decisions (a_4), which are then aggregated by the teacher (a_5).

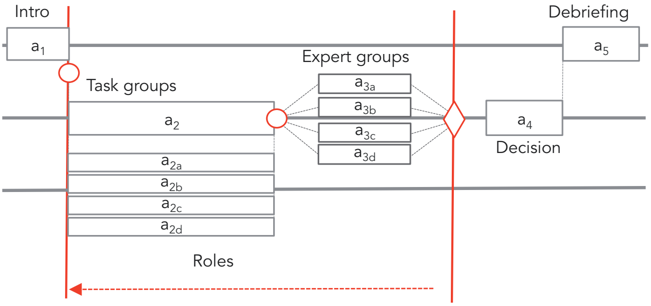


Figure 1.23 Role-based activities structure teamwork—a scenario in geology.

To implement interdependent activities, roles can be set up in 3 ways.

- Roles can be defined by the instructions given to students, such as "*Your role is to regulate the participation of all team members,*" "*Your role is to verify the data produced by other team members,*" or "*You will participate in this debate as if you were the president of an extreme-right party.*" Some pedagogical scenarios include a role-training activity in which learners practice their role and get feedback on how well they played their role.
- Roles can be enforced by the interface. In online education, the interface can be designed in such a way that a learner playing a role only has access to the subset of the functionalities that is specific to his role. In physical classrooms, when several learners sit in front of one computer, roles can be implemented by multi-mice interfaces; each mouse can only access the interface elements that correspond to its role (Infante et al., 2009).
- Roles can be induced by a distribution operator (see Chapter 3); the operator purposely provides different pieces of information to different team members, which will lead them to contribute differently to teamwork. For instance, the learner playing the role of the geology expert in the above example will receive geology maps, while the one playing an insurance agent will receive a set of cases where an insurance company has to pay for a natural disaster.

The idea of roles is not necessarily related to the notion of role-play, even though the engagement that a role-play activity may produce is not negligible. Actually, in collaborative learning, roles are mainly designed for reducing collaboration pitfalls such as social loafing (one team member lets the other ones do all the work), (see Point 21). There is evidence that roles enhance collaborative learning (Schellens & al., 2005). In other approaches (Weinberger, REF), students are expected to internalize the roles they play; playing the role of "provide counter-evidence" would lead students to acquire richer argumentation skills.

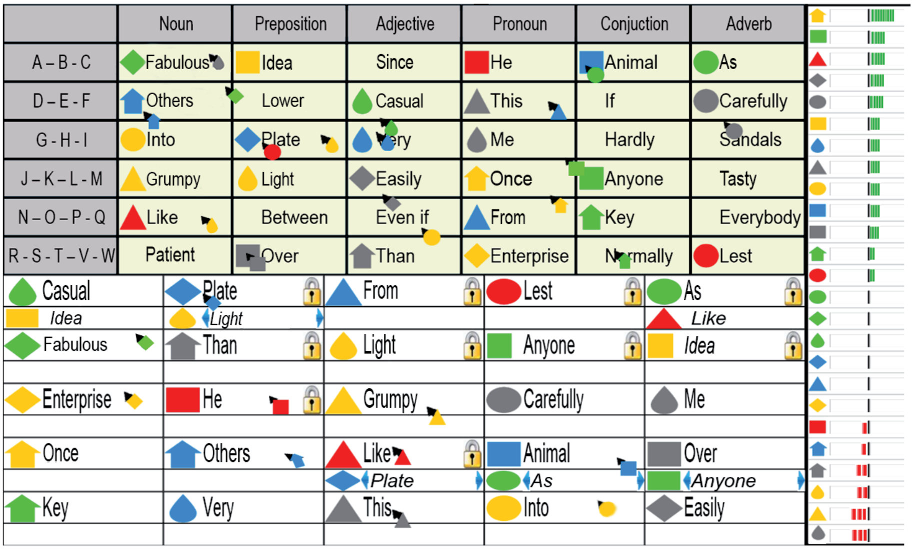
These roles implement dependencies between π_1 and π_2. Can a graph include dependencies between other planes? An example of π_2–π_3 dependency is "concurrent design,"[14] used with EPFL Space Center students. Several teams located in the same room work in parallel on complementary aspects of a satellite (energy, communication, or mechanics) while constantly following what other teams do. Any design change by one team is immediately propagated to the design process of the other teams.

In regards to dependencies between π_1 and π_3, Szewkis et al. (2011) invented "silent collaboration," an original graph in which 40 students edit a table together (Figure 1.24). Every learner (π_1) is in charge of one cell on a digital table, projected at the front of the class (π_3). The table contains 25 words—5

---

[14] http://space.epfl.ch/page-39445-en.html

columns refer to the grammatical nature of the word (adverb, adjective, noun) and 5 rows describe its first letter. Initially the words are misplaced, and each student, responsible for one cell, has to find a peer with whom to exchange his word until every word is in the right place. In the physical classroom, each student's mouse—up to 40 mice connected to the same computer—controls his own cell in the table, where he can propose, accept, or reject an exchange. In a MOOC, the same graph could be implemented with the operators described in the next chapter.



	Noun	Preposition	Adjective	Pronoun	Conjuction	Adverb
A - B - C	Fabulous	Idea	Since	He	Animal	As
D - E - F	Others	Lower	Casual	This	If	Carefully
G - H - I	Into	Plate	ery	Me	Hardly	Sandals
J – K – L - M	Grumpy	Light	Easily	Once	Anyone	Tasty
N – O - P - Q	Like	Between	Even if	From	Key	Everybody
R - S - T - V - W	Patient	Over	Than	Enterprise	N rmally	Lest

Figure 1.24 The "silent collaboration" scenario (Szewkis et al., 2011) (courtesy by M. Nussbaum).

Conclusions

This first chapter describes a very generic structure, which could model many processes other than pedagogical scenarios. So far, orchestration graphs constitute a syntactic structure, without much semantics from the field of education or learning sciences. The semantics will now be explored systematically through 3 libraries. In Chapter 2, I will present a library of edges, with each edge encompassing a pedagogical idea. Chapter 3 describes a library of data transformation operators associated with edges. In Chapter 4, I will propose a library of learner states that have a specific relevance for orchestration graphs. In other words, I have so far described the bones of orchestration graphs; the next chapters will add the muscles, the digestive system (operators), and finally the blood system (the evolution of states as a stochastic process).

Chapter 2

The Edges Library

Defining an educational scenario as a set of edges and vertices may sound terribly technical. This chapter introduces educational semantics. The edges of an orchestration graph embed the pedagogical idea that led the graph designer to place one activity next to another. The label of the edge e_{ij} defines the pedagogical relationship between a_i and a_j. The term "label" is used in graph theory to associate an edge with a value, an identifier, or anything that is meaningful in the domain modeled by the graph.

This chapter proposes a library of 28 labels that can be associated with an edge e_{ij}, that is, an edge between a_i and a_j. This library is structured into 4 categories. It is not proposed as a new ontology for education, but simply as a way to structure the chapter:

- The **preparation** edges connect two activities when the learner has a higher probability of succeeding at a_j if he carried out a_i before a_i.
- The **set** edges connect two activities when the skills or contents addressed in a_i and a_j are in relationship with each other; for example, subset/superset, whole/part, and siblings.
- The **translation** edges connect two activities in which the same content is addressed under different formats, representations, notations, or viewpoints. Learners therefore have to translate the representation used in a_i into the representation used in a_j.
- The **generalization** edges introduce variations of the content or skills across the space of generalization, namely introducing the student to more general, less general, or analogical contexts from a_i to a_j.

The labels in the categories "set" and "generalization" are associated with a symbol (+, – or =) if a_i is positioned above, below, or at the same level as a_j in the space of transitions that will be presented in the following points.

There is a typical approach in computer science to distinguish the structure from the contents that fill the structure. These contents are typically organized into some kind of library. Graph structure has been presented in Chapter 1. The edges library (and the other libraries presented later on) results from this content/structure dissociation.

An edge between two activities can be associated with more than one label; for instance, an interesting case study can prepare learners for the next activity, both as a motivation edge and as an analogy edge.

Table 2.1 A library of edge labels for orchestration graphs.

Preparation	Set	Translation	Generalization
(P) Prerequisite	(S+) Aggregation	(T) Proceduralization	(G+) Induction
(P) ZPD	(S+) Expansion	(T) Elicitation	(G+) Deduction
(P) Adv. organizer	(S–) Decomposition	(T) Alternate	(G+) Extraction
(P) Motivation	(S–) Selection	(T) Reframe	(G+) Synthesis
(P) Anticipation	(S=) Juxtaposition	(T) Reverse	(G=) Analogy
(P) Logistics	(S=) Contrast	(T) Repair	(G=) Transfer
(P) Data collection	(S=) Identity	(T) Teach	(G–) Restriction

Some of these edges could also be used to describe the cognitive processes that occur inside an activity. Since any activity can be split into subactivities, as explained in Chapter 1, there is no fundamental difference between an intra-activity relationship (i.e., inter-subactivities) and an inter-activity relationship.

The specific terms I have chosen for naming edges are not essential. One could certainly argue about many of them. What is important is not the meaning of specific labels but the **variety** of labels described hereafter. This diversity is important for two reasons. First, it illustrates the scope of pedagogical scenarios that can be modeled with orchestration graphs. Even if a graph has only three activities, placed at six possible planes, hence two edges, this library of 28 edges allows for $28^2 \cdot 6=4704$ graphs. Second, the diversity allows for building graphs that explore the whole knowledge space. The set of edges builds a complete mesh around this space. I will now describe point by point the many dimensions of this mesh.

Point 9 Preparation edges

This first category includes edges (noted as "*P*") that simply connect two activities, where one activity prepares learners for the next one. This is rather basic, but it gives the fundamental idea of educational design; given the target skills *C*, which activity a_n (the last activity) would prepare learners for *C*, then which activity a_{n-1} would make learner *s* able to do a_n, and so on recursively. The former activity increases the chances for the latter one to be effective. In some cases, the latter simply cannot proceed without the former being completed. There is a continuum of dependency levels between the two activities, the dependency being measured by the weight of the edge.

Edge (*P*) Prerequisites: This label characterizes an edge e_{ij} if students acquire in a_i the skills $\{c_i\}$ that are prerequisite to succeed in a_j and to acquire $\{c_j\}$. The term "prerequisite" implies that the probability of acquiring $\{c_j\}$ is very low if the student has not previously mastered $\{c_i\}$. For instance, subtractions without borrowing should be mastered before those where students

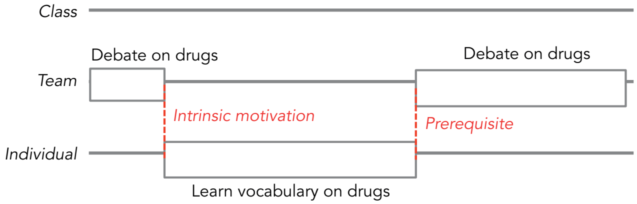


Figure 2.1 Edge labels in a "*German as foreign language*" graph. The graph includes a group debate on a topic; for example, drugs. This generates some frustration among learners due to their lack of vocabulary. This frustration hypothetically creates motivation to learn new vocabulary. This activity is a prerequisite for the debate that will follow.

must borrow. The prerequisite relationship between skills can be extracted by applying the content analysis methods described in Point 6.

The importance of prerequisite edges is central to mastery learning (Bloom, 1984); as explained in Point 4, it is a waste of effort for learners to try and achieve a_j if they have not mastered a_i before. Taking care of prerequisites is a basic idea, but prerequisite gaps, especially when accumulated over school years, is a main cause of school failure.

Edge (*P*) ZPD: This label characterizes an edge e_{ij} if a student acquires in a_j the skills $\{c_j\}$ that are located in his "zone of proximal development," given the skills $\{c_i\}$ that he has acquired during a_j. The concept of ZPD, well known in instructional psychology, has been defined by Vygostky (1962) as "*the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance, or in collaboration with more capable peers.*" In other words, ZPD refers to the difference between what the learner could perform individually in a_i (π_1) and what he can collaboratively perform in a_j (π_2), with a more knowledgeable person. Such a graph is illustrated in Figure 2.2. From a workflow viewpoint, a edge with a ZPD label will be associated with a social operator (see Point 16) that will select

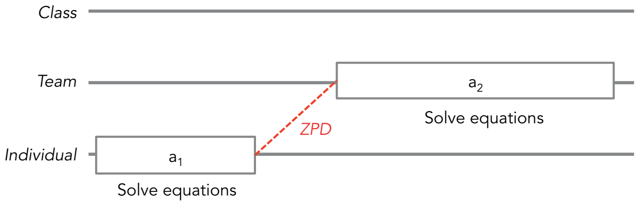


Figure 2.2 In the first activity, students solve equations individually. The teacher analyzes their work and identifies those who concentrate on algebraic manipulations compared to those able to think in terms of problem-solving strategy. In the second activity, a student from the first category is asked to work with a student from the second category. The latter is expected to convey his strategies by arguing about the choice of equation manipulations proposed by the former.

two learners s_1 and s_2, based on their knowledge state in a_i (see Chapter 4), in such way that, by interacting with s_2 during a_j, s_1 will be able to perform a_j, an activity that he could not perform alone.

Some colleagues will be upset with my claim that ZPD edges constitute a special case of prerequisite edges, but they do, even if they originate from radically different theories of learning; while the term "prerequisite" is often used when talking about cognitive skills, from an individual cognition viewpoint, the ZPD stresses verbal interactions in a social context, namely among peers (or in child-adult interactions). In socio-cultural theory, the learning mechanisms behind the ZPD is that a learner appropriates the language used by a more advanced peer while collaborating on tasks that he could not tackle individually. If I were to talk with a physicist about quantum theories, I would not understand and hence not appropriate anything he tells me. He is simply not in my ZPD. The claim that ZPD constitutes some kind of a prerequisite illustrates the pedagogical ecumenism I promote in this book.

Edge (*P*) Advance organizer: This label characterizes an edge e_{ij} if a_i pre-activates cognitive structures that will facilitate a_j. This pre-activated structure has been called an "advance organizer" by Ausubel (1960). In the example presented in Figure 2.3, the teacher first explains that the goal of the lesson is to be able to locate a point on a plane. Then, students play naval battles in pairs, which serve as advance organizers for the lecture on the Cartesian plan. I tried this graph with success on low achievers, and it "worked extremely well." The battle is, of course, also an extrinsic motivation factor.

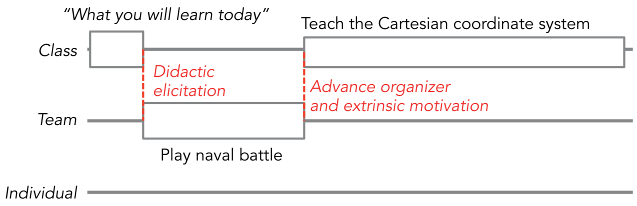


Figure 2.3 Edge labels in a mathematics MOOC.

One could also consider an advance organizer as a kind of prerequisite, but the activity a_i does not actually target specific skills; it only pre-activates structures that will hence be more rapidly activated during learning. It is partly similar to the "priming" effect observed in psychology; if I say "*banana*" to you 5 times and then ask you to cite a color, there are more chances that you will say "*yellow*." Now, an advance organizer is more than just a priming effect; it pre-activiates a structure on which the students will be able to associate the pieces of knowledge acquired during the scenario. This effect is rather useful for inductive learning. In these scenarios, also called "guided discovery learning," teachers expect learners to induce a new concept or rule by identifying the features that are common to a set of instances. In reality, an infinite set of common features exists among a finite set of objects, and hence an infinite space of generalization. Two quadrilaterals do not only have in common the features that the teacher expects learners to compare (relative length of the side, angle amplitude, parallelism between sides). They also have many other shared features; they are drawn in the same color, the same teacher has drawn them, on the same day, on the same support, they are neither big nor small, they are boring or cute, and so on. The term "guided discovery" refers to the need for the teacher to guide learners during induction by leading them towards the relevant criteria for induction. Advance organizers are among the tricks that teachers may use for driving the learner's attention to the relevant criteria.

Edge (*P*) Anticipation: This label characterizes an edge e_{ij} if a_i inform students about what they will do during a_j and/or during the whole scenario—which activities are planned, when, where are scenario activities located in the map of the course, which competencies they are expected to acquire at the end of the graph, and how these competencies will be evaluated (see Figure 2.3). This edge concerns the relationship between the first activity and the whole graph, but as explained earlier about the transitivity of edges, only the edge between a_i and a_{i+1} is represented on the graph. These activities provide students with a mental map of the path they have to follow. Students understand more of the activities they encounter, since they know where they are going. Tourneur (1975) showed that explicitly communicating the learning goals of a lesson increases the learning results of the students. It is therefore not a waste of time to spend 2 minutes at the beginning of a graph to present an overview of what will come next.

Edge (*P*) Motivation: This label characterizes an edge e_{ij} if a_i motivates students to perform a_j and/or do the whole scenario. Motivation sounds like a concept borrowed from grandfather-pedagogy or even as a behaviorist concept; children will be motivated to learn multiplication tables by the promise of receiving chocolate if they perform well. In educational games, discreetly renamed "serious games" (as if learners should be ashamed of having fun while learning), motivation is often extrinsic to learning; for example, the will to win against one opponent, competition with many opponents, or the increase in a score. These incentives may lead learners to increase the effort they put into learning activities. In university teaching or in MOOCs, this extrinsic motivation is the desire to get a certificate. University lecturers know that the trick, "*What I am going to explain now corresponds to an exam question*" boosts attention during a lecture. Extrinsic motivation is not a bad thing, but it tends to be ephemeral. Moreover, it may lead learners to find alternative strategies to reach the target reward with less effort. As some frustrated teachers say, "*They will look for any way of getting the certificate without having learned anything.*" Therefore, it is important to try raising the learner's intrinsic motivation, that is, his motivation to acquire skills or knowledge because these new skills or this new knowledge will later on enable him to do something he could not do before. Intrinsic motivation could simply be the fact that a_i presents examples of how a skill taught in a_j will allow students to solve an interesting or important problem in their future (professional) life. Another intrinsic way to motivate students is to present a problem in which the skills previously acquired are not sufficient (Figure 2.1) or produce an incorrect answer (Figure 2.4).

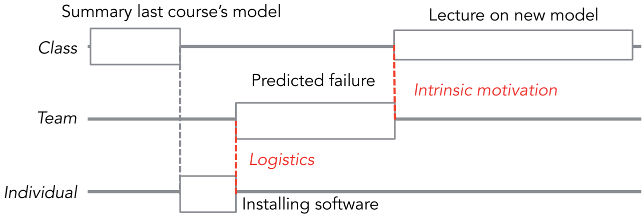


Figure 2.4 Edge labels for a science MOOC. The teacher reminds students of the model taught the previous week and asks students to install a new simulation tool (logistics edge). He gives them a phenomenon for which the previous model produces incorrect results, which will justify a revision of the previous model.

Edge (*P*) Logistics: This label characterizes an edge e_{ij} if a_i is required to set up the environment necessary to be able, practically speaking, to conduct a_j—to log in to a digital environment, to prepare tools for a lab session, to download and install the software necessary for an activity, to cut pieces of paper for a lesson on symmetry, to change the room configuration before teamwork, and so on. The term "logistics" is not negative, these activities are important in order to run the scenario. This emphasizes the pragmatic viewpoint on education that underlies the concept of orchestration.

Edge (*P*) Data Collection: This label characterizes an edge e_{ij} if a_i aims at collecting data to be used in a_j. Many scenarios include data collection activities, such as field trips in schools, measurements in university labs, and data mining in online environments. In some cases, these activities are not actually proper learning activities, but simply provide the data necessary for the next activities. In this case, the data collection edge can be seen as a subcategory of the logistics edge. In other cases, the ability to collect data can be a learning goal; for instance, where to collect it, how to avoid measurement biases, which frequency, which precision, and how to store data. Chapter 3 presents pedagogical tricks regarding this edge; the way data is collected may determine the cognitive processes when these data are processed. For instance, the diversity of the data collected (e.g., the acidity of the river respectively measured upstream and downstream of the city) defines the space of comparisons that can be conducted later on in the scenario.

Point 10 Set edges

This category of edges connects two activities in terms of the relationship that characterizes the contents addressed in each activity. The contents are knowledge elements—concepts, examples, data, principles, laws, rules, algorithms, and so on. Are the contents being addressed in a_j a subset, a superset, or a complementary set to those addressed in a_i? These edges can connect more than two activities. The various labels are described as moving upwards ($S+$), downwards ($S-$) or transversally in the hierarchy of subsets and sets; if K_i and K_j are the sets of objects addressed respectively in a_i and a_j, the edge e_{ij} is labeled $S+$ if $K_i \subset K_j$. Conversely, e_{ij} is labeled $S-$ if $K_i \supset K_j$. In the other cases, the label is $S=$.

Edge ($S+$) Aggregation: This label characterizes an edge e_{ij} if learners acquire skills in c_i that will be aggregated in c_j. Let's consider 3 activities, a_1, a_2, and a_3, with two edges, $e_{1,2}$ and $e_{2,3}$. An example of aggregation would be if the learner acquires skill c_1 in a_1, then skill c_2 in a_2, before doing an activity a_3 in which he has to integrate c_1 and c_2 into a new skill c_3. For instance, if c_1 concerns negative exponents ($n^{-a} = 1/n^a$) and c_2 concerns rational exponents ($n^{1/a} = \sqrt[a]{n}$), these two skills can then be "chunked" (Laird et al., 1986) into a new skill that allows, for instance, calculating the value of $n^{-1/a}$. In an "*English for non-native speakers*" course (Figure 2.5), c_1 could target the ability to build a question in English, while c_2 could be about the construction of negation, and chunk c_3 would be about the construction of negative questions. As explained earlier in regards to multiple edges, I would then label $e_{2,3}$ as aggregation and $e_{1,2}$ as "juxtaposition" (see this label hereafter), but I would not explicitly represent $e_{1,3}$ (simply for the sake of keeping the graph easy to read).

Edge ($S-$) Decomposition: This label characterizes an edge e_{ij} if learners acquire skill c_1+c_2 in a_i, then work only on skill c_2 in a_j, as explained in Figure 2.5. This edge is less common than the previous one, since many

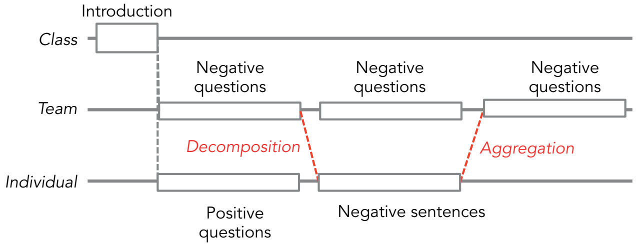


Figure 2.5 After an introduction, the teacher splits the class into two subclasses, those who have already studied how to form questions and negative sentences in English, and those who have not. The novices do individual exercises on each skill (first questions and then negative sentences), and finally these two skills are aggregated during pair dialogue exercises that include negative questions. The more experienced subclass starts directly with the pair dialogue exercises, but the students who encounter difficulties are then redirected towards individual exercises on each skill.

pedagogical scenarios follow the basic pedagogical principle of progressivity; the sequence of activities goes from the simple to the complex, from the easy to the difficult, from blue ski slopes to black diamond ones. Situations do exist, however, where inverse progressivity is more relevant. The first situation is remediation, that is, courses that compensate for the skills that students were supposed to possess at the beginning of a course, but do actually not. If some skills are not mastered, the system will focus on the weakest subskill and so on recursively. Typically, in the exponent example presented for the aggregation label, one would proceed the other way around, from c_3 to c_1 or c_2. Another reason for inversing progressivity is motivation; it allows the teacher to start from a real-world problem that is meaningful and then break it down progressively into elements that students can learn before reaggregating them. This inverted progressivity is, for instance, present in problem-based learning scenarios. It is especially relevant in corporate training, because it allows for starting from the problems that trainees encounter at their workplace.

Edge (S–) Selection: This label characterizes an edge e_{ij} if a_j (usually at the end of the graph) addresses a selection of the key elements that have been addressed in a_i and/or in previous activities. For instance, among all the dates presented in a history lesson, the final activity will select the most significant ones; among all the paintings of Picasso, his best known works will be selected; among the causes of diabetes, the 3 most frequent ones will be integrated; among the many types of whales, the 3 clearly distinct ones will be selected. The selection relies on criteria such as "*It is worth remembering this in 10 years*" or "*This will be tested in the exam.*" I differentiate this type of summary from a real synthesis. A synthesis includes some elaboration on relationships between content elements, some structuring or abstraction, and is therefore listed in the G+ category (Point 12). Here, the summary simply re-represents some selected items among those previously presented. The summary contents is a subset of what been presented in the scenario, which is why it is labeled S–.

Edge (S+) Expansion: This label characterizes an edge e_{ij} if a_j reuses content that was used in a_i but expands on it in terms of number of examples, quality or quanitity of data, etc. For instance, if students uploaded pictures of the geological phenomenon under scrutiny, the expansion consists in adding their geographical location, or links to other pictures of the same location, or links to the relevant phenomenon, or comments, or feedback. The material has been enriched. If students collected pictures of paintings, they would have to find more information on them (scenario illustrated in Figure 2.6). If students have to produce sentences in a_i, the expansion could be to permute the words of these sentences, following certain grammatical rules, in order to expand the set of produced utterances. If during a_i, students calculated some features based on a geometrical figure they constructed on paper, the next activity might be to automatically repeat the same to 1,000 variations of the same figure. The rationale of expansion is to have a sufficiently broad or

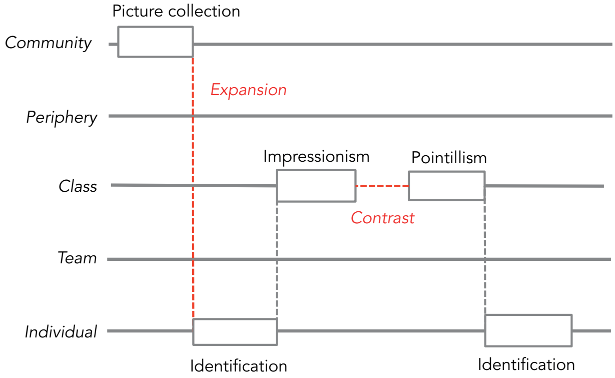


Figure 2.6 Through their social network, MOOC participants collect pictures of well-known paintings from the 19th or 20th centuries that people have in their home, office, or any local place. The participants have to explore the web to find out which pictures belong to impressionism and, if possible, who the painter was, which country he was from, and in which year the picture was painted. The selected pictures and the additional information are uploaded and aggregated through an operator (see Point 13) for illustrating a video lecture on impressionism. The learning environment allows the teacher to sort the paintings by painter, country, and year. Then, the teacher introduces pointillism. The participants' weekly assignment is to identify pointillist pictures among those they collected and again to find out who the painter was.

sufficiently rich set of data, measures, or examples before conducting inductive or abstraction activities (see *G*+ edges in Point 12). With the amount of information available on the web, many ways of expanding material exist: on-line lexicons, translation tools, conversion tools, encyclopedia, bibliographies, and so on. The expansion can be done manually by the teacher or by automated operators, classified in Chapter 3 into the "back-office" category.

Edge (*S*=) Juxtaposition: This label characterizes an edge e_{ij} if the ordering of a_i and a_j is not determined by cognitive prerequisites or by any ontological or epistemological considerations, but simply by the need to address one item at a time. Let's consider a MOOC on physiology, with an activity on the bones in the arm and an activity on the bones in the leg. These activities are not conditional to each other, but are simply two subsets of the skeleton that can be taught in any order. Therefore, this label simply means that the connected activities are independent subsets of a larger goal. It is important to make explicit that the order is arbitrary, so that it can be modified without consequences. The weight of an edge with this label will be very low.

Edge (*S*=) Contrast: This label characterizes an edge e_{ij} if K_i and K_j, respectively addressed in a_i and a_j, are mutually exclusive subsets of K, the set of contents addressed in the scenario. These subsets may be used later on for induction and discrimination (see *G*+ edges in Point 12). While juxtaposition edges connect content that is more or less independent (except when it belongs to the same superset), contrast edges connect two contents that are opposite to each other: positive versus negative instances, crime rates in countries with or without the death penalty, medical cases where the patient has disease X versus cases where he has similar symptoms but another disease, impressionist painters versus pointillist painters (Figure 2.1), the digital solution versus analogical ones, and so on.

Edge (S=) Identity: This label characterizes an edge e_{ij} if a_i and a_j are (almost) identical. Some skills, mostly procedural skills, require intensive practice before learners become fluent. In this context, a_j may be a set of exercises very similar to a_i, but it consolidates the skills practiced in a_i, often with a slight variation or slight increase in difficulty. This is central to most "drill and practice" scenarios, which may sound terribly boring, but are useful for acquiring certain skills and for being able to operate them with a minimal cognitive load.

In summary, S labels describe a sequence of activities that evolve along various subsets of content. If a set and its subsets (and recursively their subsets) are represented by a tree (Figure 2.7), the S edges represent up/down movements or left/right movements within this tree structure. If, as often happens, subsets are not mutually exclusive, the tree is then a graph, which won't fundamentally change the value of addressing the same content at various levels of aggregation.

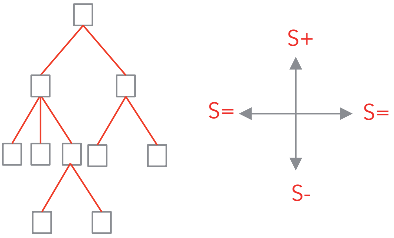


Figure 2.7 Structure of the content sets covered by activities.

Point 11 Translation edges

The same piece of knowledge can be represented in several forms and formats: as a definition versus as an algorithm, with concrete elements versus with abstract symbols, as a pie chart versus a histogram, on a linear scale versus a logarithmic scale, and so on. Translation edges connect activities where the same contents are being processed under different formats, expressions, or representations. Education would greatly benefit from having more activities that imply some representational translation.

Edge (T) Proceduralization: This label characterizes an edge e_{ij} if the aim of a_j is for learners to translate the knowledge acquired in a declarative format in a_i into procedural knowledge. Let's imagine that learners have to learn how to calculate the surface of a triangle. A declarative format for this knowledge is that "*the surface of a triangle is half the product of its base by the height relative to the base.*" The procedural version is an algorithm for calculating this height: (1) select one side, (2) measure it, (3) find the opposite summit, (4) draw a line segment from this summit perpendicular to the selected side, (5) measure it, (6) multiply both measures, and (7) divide the sum by 2. This procedural knowledge should be mathematically equivalent to the declarative knowledge, but, from the cognitive viewpoint, they are two different things. This label is indeed common in instructional design; the learner has acquired some declarative knowledge in a_i that will be applied to specific objects in order to provide an answer in a_j. For instance, he learns the definition of a concept and then has to identify examples in a set; he learns how to calculate the Eigen values of a matrix, which he will apply to n matrices. Proceduralization, as its name indicates, turns declarative knowledge into procedural knowledge (Figure 2.8). This is often associated with compilation; the repeated application of a procedure reduces the cognitive load required to operate this procedure.

Edge (*T*) Elicitation: This label characterizes an edge e_{ij} if the aim of a_j is for learners to translate the knowledge acquired in a procedural format in a_i into declarative knowledge. This edge is the inverse of the previous one, proceduralization. In activity a_i, students have learned an algorithm to calculate a result or perform a task, and activity a_j aims to turn this procedural knowledge into declarative knowledge. Elicitation can be quite difficult; try to explain in words how you tie your shoelaces, how you convince your opponents, how you solve complex problems, or how you manage your team. In many cases,

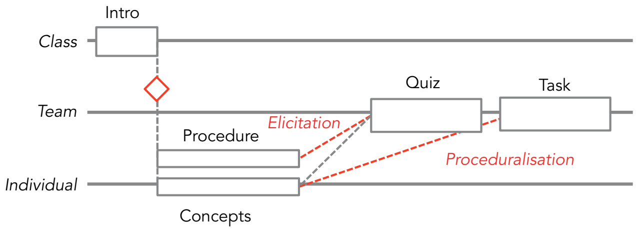


Figure 2.8 After an introductory video, the participants in this MOOC, "*Introduction to statistics*," are split into 2 subclasses for individual activities. In the first subclass, students acquire procedural knowledge—how to manually calculate the standard deviation for a set of 20 data points. In the second subclass, students acquire declarative knowledge—the concepts of dispersion, heterogeneity and variance, illustrated by graphical representations. Then, each student from a subclass is paired with a student from the other subclass, and collaboratively they have to do a quiz that measures declarative knowledge and then a task that requires procedural knowledge. To be able to collaborate with their peer, those who acquired declarative knowledge individually have to proceduralize it with the help of their peer, and those who acquired procedural knowledge individually have to elicit it (next edge label).

procedural knowledge is efficient because it is compiled (or automated), and eliciting this knowledge—turning it into words—is far from simple. There are cases where it seems, however, better to wait for the practice to be semi-compiled before eliciting it. In computer science, defining concepts such as "*A variable is a container that stores values*" or "*Recursion describes an object that includes itself*" do not make any sense for students. In my experience, it is more effective to let them use these elements for a while before eliciting this knowledge through a debriefing lecture or a collaborative task. Forcing peers to elicit their knowledge in order to convince their teammates is one of the reasons why collaborative learning is useful.

Edge (*T*) Alternate: This label characterizes an edge e_{ij} if a_j addresses the same contents as a_i, but with a different external representation. Research in instructional psychology has shown that using multiple representations can have a positive effect on learning outcomes (Ainsworth, 1999); for instance, mathematical contents that have been presented as a matrix in a_i and are then presented as a transition graph in a_j. Such a shift among representations is essential in early numeracy for children, passing from concrete, to semi-concrete, and then to abstract representations of quantity. Shifting representations has a double role. If the learner understood the concept in a_i, the representation shift in a_j will deepen his understanding by dissociating the concept from one particular mode of representation. If the learner did not understand the concept in a_1, the representation shift in a_2 constitutes a second chance to grasp it. Some earth maps display the South Pole at the top and the North Pole at the bottom; they prove that every representation conveys some bias and, therefore, that multiple representations allow a broader semantic coverage of the concepts to be learned. The alternate edge is not only bound to graphical representations, it also concerns alternative notations, alternative formats, and overall alternative approaches, methods, algorithms, and solutions. Time constraints too often prevent teachers from letting students elaborate multiple solutions to the same problem or from presenting multiple solutions to them, even though this could be fundamental for students to understand that the same problem can be addressed in different ways.

Edge (*T*) Reframe: This label characterizes an edge e_{ij} if a_j addresses the same contents as a_i, but with a different viewpoint, approach, scale, frame, context, or unit. For instance, a_i uses a geometric referential that is absolute, while in a_j, all equations are modified using an object-specific referential. In a_i, a medical case is analyzed from the viewpoint of the patient, while in a_j it is analyzed from the point of view of the medical team. In a_i, companies are compared in terms of stock value, while in a_j they are compared in terms of potential growth. In a_i, the transportation costs are described in pounds per gallon, while in a_j they are measured in euros per liter. This edge also applies when implementing a change of **scale**, when what has been studied at the neuronal level is reconsidered at the brain level; from a water molecule to a cloud, from a village to a country, or from a function to an entire software.

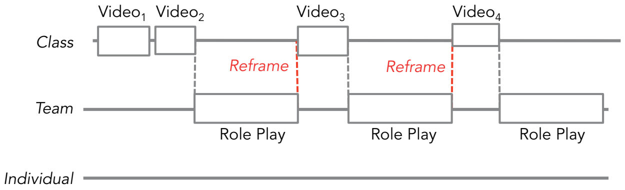


Figure 2.9 This MOOC concerns conflict negotiation skills for humanitarian workers. In the first video, the teacher introduces the history of the conflict between *X* and *Y*. The second video is a set of excerpts from television news produced by camp-*X* and the third video by camp-*Y*. After V2 and V3, the students participate in a 60-minute role-playing game where they simulate negotiation via an online conferencing tool. In the last video, the teacher presents techniques for conflict negotiation that participants apply in the last role-playing activity.

Finally, this edge may connect activities that provide different viewpoints, as illustrated by Figure 2.9: for instance, an and then

Edge (*T*) Reverse: This label characterizes an edge e_{ij} if a_j addresses the same contents as a_i, but in a reverse way—from *B* to *A* instead of from *A* to *B*. A typical transition in pedagogical scenarios is to connect two activities that use the cognitive function f in a_i and f^{-1} in a_j. If the student in a_i learned the formula that calculates the surface of a triangle from its base length and its height, an inversion would be to ask him in a_j to calculate, for instance, the height of a triangle from its surface and base length.

Edge (*T*) Repair: This label characterizes an edge e_{ij} if the function used in a_i to find a solution is applied in a_j in order to find an error in a task, or in an exercise that has been solved. It can be viewed as a subtype of the reverse edge, but it is worth making it a category, as it is a common educational practice.

Edge (*T*) Teach: This label characterizes an edge e_{ij} if the learner has acquired some skills in a_i that he will teach or transmit to another learner in a_j. If the student simply repeats as teacher what he heard as learner, there won't be major learning effects. Learning comes from the processing required to prepare teaching material. By experience, many teachers know that "*you don't know a topic before you have taught it*." This is confirmed by empirical evidence regarding the effectiveness of "learning-by-teaching" (Bargh & Schul, 1980; Chase et al., 2009).

In summary, *T* labels describe transitions between activities in which the contents are addressed under different formats, representations, or viewpoints. These different forms of what could be seen as the same piece of knowledge (e.g., as the same entry in an encyclopedia) are not actually the same piece of knowledge from a cognitive viewpoint. Figure 2.11 shows two axes of the space created by these edges, concrete versus symbolic and declarative versus procedural. Many other dimensions also exist (represented by grey arrows) such as cognitive versus kinesthetic, verbal versus spatial, and so on.

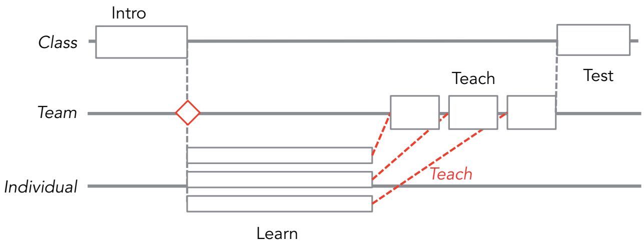


Figure 2.10 After a general introduction on DNA, the class is split into 3 subclasses. In each subclass, students learn about different topics individually, namely DNA replication, DNA transcription, and reverse transcription. Then, students form teams of 3 in which each student has to teach the other two what he learned individually. The scenario ends with a quiz test in the lecture theatre, using clickers, where each team is scored based on the answers given by the team members who have been taught (the teaching peer is not allowed to reply).

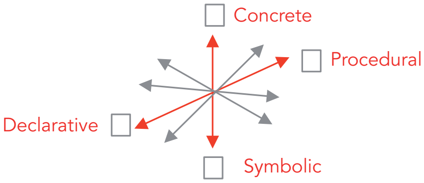


Figure 2.11 The multidimensional space of formats.

Point 12 Generalization edges

The fourth category of edges describes the relationship between two activities that differ by the level of generalization of the contents being addressed. *G*+ edges generalize whatever has been learned in the former activity; for instance, abstract principles or inducing categories. *G*– edges describe the relationship between two activities, such as the second zoom-in on particular cases of the general elements acquired in the previous activity. While *G*+ edges implement inductive learning scenarios, *G*– edges implement deductive scenarios. Finally, *G*= describes the relationship between two activities located at the same level of generality, such as in analogy.

Edge (*G*+) Induction: This label characterizes an edge e_{ij} if the aim of a_j is for students to elaborate some general knowledge from the set of specific elements encountered in a_i. Typically, a set of examples is processed in a_i, then the concept to which these examples belong to is elaborated in a_j. For instance, in botanic studies, students might examine the properties of tree leaves in a_i before classifying them into various categories during a_j. Or, in a_i, students measure the dimensions of various triangles and estimate their surface with paving methods, and then in a_j, they elaborate a general formula for calculating the surface of a triangle. Induction is the essence of constructivism, the art by which a teacher may create the "aha!" moment. Our brain is a permanent inductive machine; once we know two people from the same country, we cannot avoid inferring (over-) generalizations. Despite this innate induction talent, school induction is as difficult as the inductive reasoning involved in scientific discovery. Since learners are expected to induce knowledge faster than humanity discovered it, the pedagogical scenario needs to scaffold the inductive process. This scaffolding will come from advanced organizers (the *P* edge), as well as from the careful selection of positive and negative instances. In particular, teachers can play with so-called "near-miss" examples; these are negative instances that differ from positive ones by a single feature, such as a rectangle, which is a square that doesn't have isometric sides. Induction consists of building a large tree of generalizations, based on the combination of all possible features, such as when we play MasterMind. Near-miss examples allow learners to isolate one feature and therefore to prune the generalization tree. Another form of guidance for induction is the divergence of viewpoints triggered by social interactions; Schwartz (2005) showed that pairs build more abstract representations than individuals, since this shared representation has to bridge the mental images that peers elaborate individually.

Edge (*G*–) Deduction: This label characterizes an edge e_{ij} if the aim of a_j is for students to apply a general rule, definition, or principle to a specific element they acquired in a_i. This edge is the opposite of induction; it goes from the general to the particular. It is overrepresented in pedagogical scenarios, which often start by a set of definitions and then invite learners to apply them to specific cases.

Edge (*G*+) Reflection: Experience per se is not knowledge; some people may repeat the same mistakes their whole life; reflection is required to

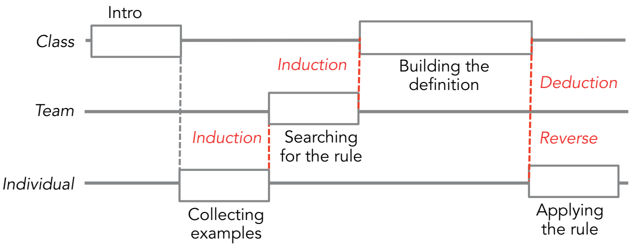


Figure 2.12 The teacher explains the goal—to find the rule that calculates the number of diagonals in a polygon from the number of edges. He assigns a number between 3 and 8 to each student. Each student draws a polygon with the number of edges assigned to him. Then, students form teams of 4 made up of students who drew a polygon with a different number of edges. They try to find the rule. After a while, each team presents its solution(s) to the class. The teacher compares the invented rules, proposes counter-examples that disprove some proposed rules, and ends up writing the formal rule. Finally, he asks students to apply the rule (G– Deduction edge label) in a reverse way (T– Reverse edge label) to calculate the number of edges of a polygon with 35 diagonals.

turn experience into knowledge. The label 'reflection' characterizes an edge e_{ij} if the aim of a_j is for students to elaborate knowledge by reflecting on the experience they gained in a_i. Some scenarios follow an approach à la Dewey (1938), where students acquire experience in a_i from which they are expected to extract knowledge during a_j. Typically in a_i, students carry out some experiments, solve a problem, or explore a complex environment, while a_j includes activities such as teacher-led debriefing or reflection activities (comparing, analyzing, annotating, commenting, etc.). The term 'reflection' describes the activity in a_j but, as an edge label, it refers to the pedagogical relationship between of a_j and a_i. For dual vocational education, we developed a pedagogical model, the "*Erfahrraum*" (Dillenbourg & Jermann, 2010), which orchestrates the reflection process, illustrated in Figure 2.13.

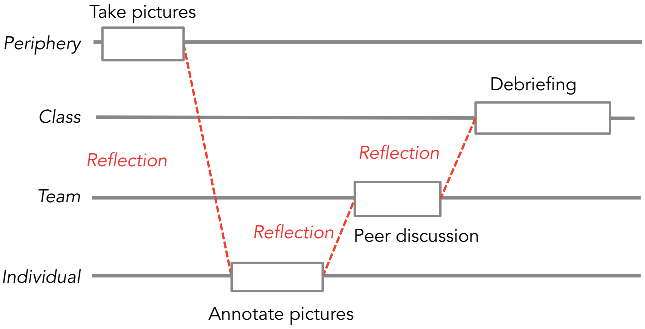


Figure 2.13 In a dual educational system, apprentices work 4 days per week in a company (π_4) and attend school one day per week. In an "*Erfahrraum*" scenario, technologies are used in a_i for capturing experience in a workplace. For instance, in the first activity, future bakers take pictures of their products, or car mechanics record their own actions with a head-mounted camera. This captured experience is uploaded to the apprentices' learning journal. The next activities trigger several forms of reflection upon the captured experience. In the second activity, they annotate the objects saved in their own learning journal. In the third activity, they compare their own objects with the objects captured by their peers; for instance, two breads baked by apprentices, or they analyze the mistakes they made while changing the battery of a car. In the last activity, the teacher extracts lessons from this material; for instance, pointing out the most common mistakes or the differences between companies.

Edge (*G*=) Analogy: This label characterizes an edge e_{ij} if the aim of a_j is for students to solve a problem through analogy with a problem they solved in a_i. Typically, students solve a problem in a_i, for instance with the teacher (π_3), and are then asked to solve a similar problem individually (π_1) by analogy with the first problem. Analogy is a powerful, but slippery form of reasoning; situations may be superficially similar but deeply different. This is especially the case when analogies are made across different disciplines, such as a waterfall to explain electricity measures or a musical metaphor, such as orchestration, for describing pedagogical scenarios. It is, however, central to several pedagogical scenarios, namely case studies, which are very popular in fields such as management, medical diagnosis, and, of course, law studies. Being an expert in a field often means having accumulated a large collection of cases, each of them being representative of a set of similar cases. The expertise lies in the criteria used for selecting the most relevant previous case for tackling the current situation. This expertise is often a highly compiled knowledge and hence requires elicitation edges (*T*).

Edge (*G*=) Transfer: This label characterizes an edge e_{ij} if the aim of a_j is for students to apply the skills they acquired in a_i to a new context. Transfer is the Achilles' heel of formal education; students rarely spontaneously transfer what they learned in a specific situation to a new situation. A pupil who is able to solve "*If 1 bottle of milk costs 5 francs, how many can I buy for 35 francs*" will not necessarily be able to solve the problem "*If a car travels a distance of 1 kilometer in 5 minutes, how many kilometers will the car travel if it continues at the same speed for 35 minutes.*" Liters and kilometers or francs and minutes are not the same entities in the real world. The cognitive schema acquired in one context does not spontaneously apply to another context. Scholars differentiate near-transfer, where only surface features of the situation vary, from far-transfer (Perkins & Salomon, 1992), where the deep structure of the situation has to be adapted. I do not consider near-transfer as transfer and recommend that pedagogical scenarios strive for far-transfer. This issue is not specific to children. Some of my colleagues complain that their students (e.g., in a computer vision class) cannot apply algorithms learned the previous year in a math class. This phenomenon can be explained. Learners' cognitive schema are naturally anchored to the learning context, bound to examples used during lectures or exercises, and these anchors are not recognized in the new situation. Low transfer is part of human nature, but

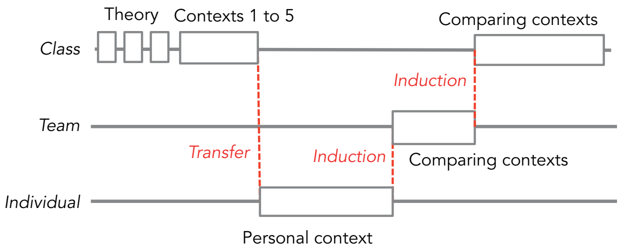


Figure 2.14 This corporate training MOOC starts by watching several videos that introduce management concepts such as corporate silos and developing examples of silos across 5 companies. Then, trainees are asked to prepare a poster that describes a silo they have identified in their own branch of the company. They upload their poster. After this online part of the scenario, the participants go to Zermatt for a residential seminar, where they work in small teams and compare how silos exist in different contexts. In the last activity, the coach leads a debriefing activity in which the comparison is extended to all examples uploaded by participants.

can be compensated for through well-designed pedagogical scenarios. Experiments show that the probability of being able to transfer increases if the examples used in activities vary systematically, re-instantiating them in radically different contexts. This would be the purpose of such an edge.

Edge (*G*–) Restriction:[1] This label characterizes an edge e_{ij} if the aim of a_j is for students to reduce the scope of validity of what they learned in a_i. Orchestration graphs need an edge for a common situation in which the concept, rule, or theory presented in a_i actually constitutes a temporary overgeneralization; it is taught as if it is always true, even if it is not the case. This overgeneralization is the price to pay for simplicity; teaching the rule and its exceptions at the same time could confuse students. Therefore, in several scenarios, an activity first leads students to acquire a general rule, principle, or algorithm, and then another activity reduces the generality and the scope of validity of this general knowledge. An activity on the components of democratic systems would be followed by examples of countries where the president is elected with 99% of the votes. A physics law that worked well with all examples in a_i will then be confronted with cases where it does not apply; for example, extreme temperatures and materials with unique properties. Restriction consists in pointing out exceptions to a general rule, specifying limits of validity, conditions of applicability, pointing out "special cases," and so on.

Edge (*G*+) Synthesis: This label characterizes an edge e_{ij} if a_j provides a synthesis of what students have learned in a_i and in previous learning activities. The summary edge is placed in the *S*– category (Point 10) if this summary is a mere selection of the key elements taught before, that is, a subset. However, a real synthesis is much more than just a selection. It elaborates relationships between elements taught distinctively, points out similarities, false friends, complementarities between methods, the relative strengths of theories, and so on. It helps learners to build a higher-level vision of what they learned, which is why I label it as a *G*+ edge.

In summary, *G* labels describe the evolution of the graph on the level of generality, which is represented vertically in Figure 2.15. The space is represented as a cone because a piece of knowledge at a high abstraction level covers more contexts (*T* or *S*) than a piece of knowledge located at the lower level.

---

[1] Since I have used an "induction" label, one might expect to find a "discrimination" label; learners elaborate concepts or rules by induction over positive instance, but also by discriminating positive against negative instances. It is, however, reasonably hard to dissociate induction and discrimination, since asking "*How is* A *similar to* B?" is the same question as asking "*How is* A *different from* B?" Discrimination is to induction what half-full is to half-empty. So, this is why I have left discrimination within induction.

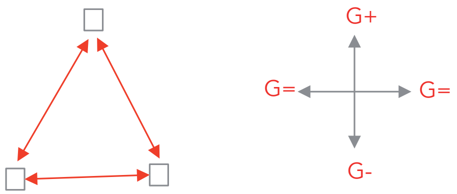


Figure 2.15 The space of generalization.

Conclusions

This list of labels is certainly not exhaustive, and the proposed classification is not perfect. There are some overlaps, but this is not a real issue, since the goal of such a list is to promote diversity. Instructional designers should use the table of edges as a kaleidoscope, that is, to look through this lens in order to diversify the activities. This concern for diversity is not specific to orchestration graphs; it is relevant for any instructional design. Nonetheless, I hope that by making the relationship between activities explicit and by proposing a broad range of potential relationships, the proposed modeling language will enrich the diversity of activities that learners will encounter within pedagogical scenarios.

My personal view on education is that **diversification is more important than adaptation**. At the class level, diversification actually embeds adaptation—the more diverse the learning activities, the higher the probability that learners will encounter an activity adapted to their personal needs. But, the diversification of activities is even more important for an individual learner, because knowledge is not monolithic. Let's consider various pieces of knowledge such the triangle, adverb, passive voice, democracy, surrealism, standard deviation, Pythagoras, Renaissance, Ohm's laws, Fourier transforms, or restricted relativity… Each of these elements can be considered as "a piece of knowledge" and expressed by a definition, a set of principles, or an algorithm. In their external representation, they can be monolithic, a "thing" captured by a finite set of symbols. This is the encyclopedic view of knowledge, a way of storing knowledge in persistent media, the way we often measure it during exams. But, knowledge is different. It is the capacity to act, to tackle situations. In this way, the Pythagoras theorem that can be cited during an exam is not the same piece of knowledge as the one used to calculate the size of a triangle, nor the one for the height of a pyramid. What I have just called a "piece of knowledge" is not actually "a piece," but rather a mesh of knowledge fragments that have been acquired in various contexts and through various representations. In the brain, a piece of knowledge does not correspond to a small, identifiable set of neurons. Any of these concepts can activate very different areas. Knowledge looks more like a tree than like an apple, more like a cloud than like a stone.

The implication of this epistemological stance is that education needs diversity. It is crucial to address the same elements from a multiplicity of formats, viewpoints, representations, and levels of generalizability. If the graphical representations used for each set of edges (Figure 2.7, Figure 2.11, and Figure 2.15) are overlapped, a multidimensional mesh is obtained, as represented in Figure 2.16 (left). I recommend designing orchestration graphs where the learner crosses this mesh in multiple ways, as many ways as time permits. You can't be said to "know" a forest if you only run on a path that crosses the forest or goes around the outside of it; you can only "know" a forest once you have run across it in all possible directions, as well as off the paths.

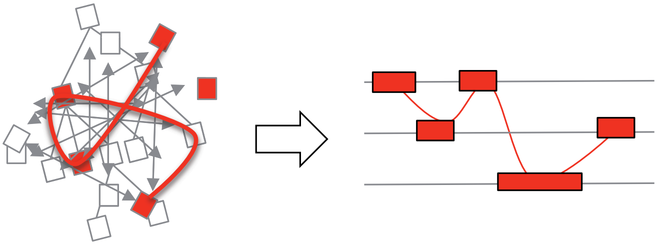


Figure 2.16 Left: The scenario gathers activities that lead learners to cross the mesh in multiple directions. Right: The path is then unwound on the 2D graph space.

The reader may wonder why this space and the navigation across multiple forms of the same knowledge is not integrated in the graphical representation of orchestration graphs. Orchestration graphs are already represented in a 2D space and a 3^{rd} dimension will be introduced in Chapter 4: it would make no sense to try to add the many dimensions to the knowledge mesh. Orchestration graphs propose a simplified view. In metaphoric terms, the sequence of activities form a complex manifold trajectory across the knowledge mesh space and it has to be unwound and placed on the simplified structure of orchestration graphs, as illustrated by Figure 2.16 (right).

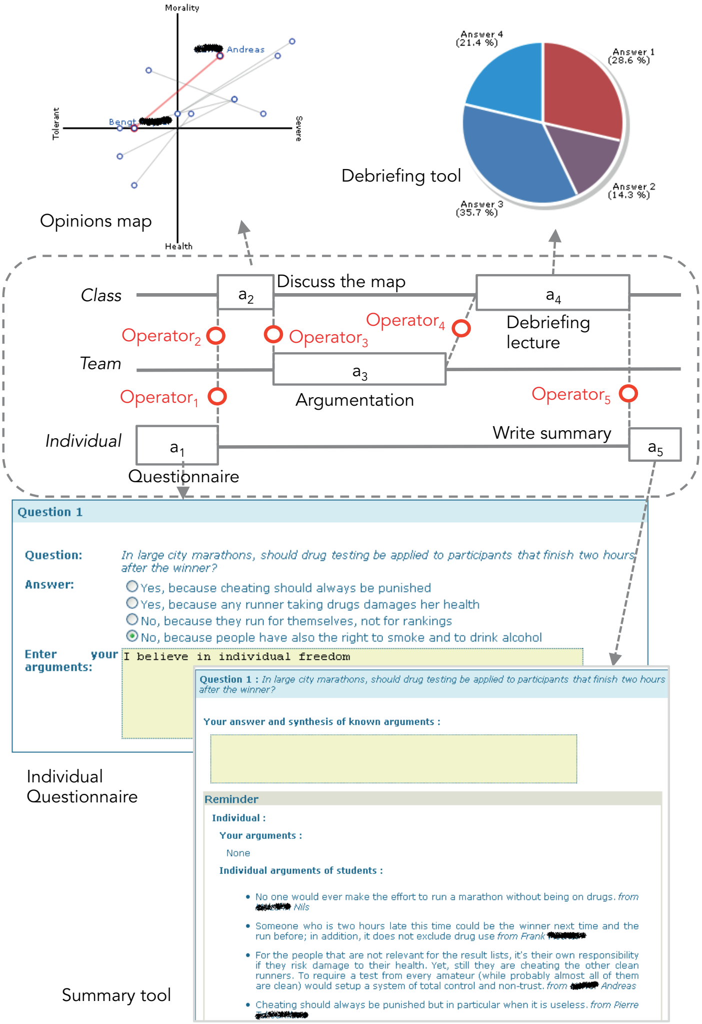
Chapter 3

The Operators Library

Chapter 1 illustrated pedagogical scenarios that correspond to well-known educational practices, such as problem-based learning, peer instruction, and adaptive instruction. The library of edges described in the second chapter opens the door to a much broader variety of pedagogical scenarios. These scenarios are more sophisticated, but a teacher who navigates easily in the knowledge space can manage them with 20 or 30 students. The question addressed in this chapter is how these scenarios can be brought to a scale of 100 or 10,000 students.

Scaling up requires translating some of the orchestrating actions that a teacher would manually perform with a small class into operators that can be automatically computed with a large number of participants. I will illustrate the notion with—please excuse me—one of my own scenarios. ArgueGraph is a scenario that scaffolds cognitive conflict between peers. It is inspired by socio-constructivist theories that predict that the interactions necessary to overcome a cognitive conflict enhance learning (Mugny & Doise, 1978). The ArgueGraph scenario includes 5 activities.

- In a_1, each student responds to an online multiple-choice questionnaire. The questions do not have right or wrong answers, but reflect different viewpoints. For each answer, the student has to write a few words justifying his choice.
- In a_2, the system produces a map of opinions, and the designer associates every answer in a_1 with an (x,y) value on the map. The teacher discusses this map with students, who often comment on their absolute positions and their position relative to their friends. The system forms pairs of students in a way that maximizes their distance on the map, that is, it finds students whose responses in a_1 reveal opposite opinions.
- In a_3, pairs answer the same online questionnaire as in a_1. The environment provides them with the answers and justifications provided by each peer in a_1. Pairs must select a single answer. They are also asked to justify their choice by writing a few words in a text box.
- In a_4, the system aggregates the responses and justifications entered by individuals in a_1 and by pairs in a_3. The teacher uses this list to build a lecture fed with the students' contributions. During this semi-improvised lecture, he asks them to provide further clarifications, to rephrase their



Question 1

Question: *In large city marathons, should drug testing be applied to participants that finish two hours after the winner?*

Answer:
○ Yes, because cheating should always be punished
○ Yes, because any runner taking drugs damages her health
○ No, because they run for themselves, not for rankings
● No, because people have also the right to smoke and to drink alcohol

Enter your arguments: `I believe in individual freedom`

Question 1 : *In large city marathons, should drug testing be applied to participants that finish two hours after the winner?*

Your answer and synthesis of known arguments :

Reminder

Individual :

Your arguments :

None

Individual arguments of students :

- No one would ever make the effort to run a marathon without being on drugs. *from [redacted] Nils*
- Someone who is two hours late this time could be the winner next time and the run before; in addition, it does not exclude drug use *from Frank [redacted]*
- For the people that are not relevant for the result lists, it's their own responsibility if they risk damage to their health. Yet, still they are cheating the other clean runners. To require a test from every amateur (while probably almost all of them are clean) would setup a system of total control and non-trust. *from [redacted] Andreas*
- Cheating should always be punished but in particular when it is useless. *from Pierre [redacted]*

Figure 3.1 ArgueGraph (Jermann & Dillenbourg, 1999). The symbols “Op” refer to dataflow operators as explained hereafter.

justifications, and to compare them. He reformulates the students’ justifications using the correct terminology and integrates them into a consistent theoretical framework.

- In a_5, each student writes a summary of the arguments collected by the system in a_1 and a_3, structured according to the framework presented in a_4.

This graph usually spreads across 3 periods of 50 minutes. The learning analytics we conducted revealed that when students expressed conflicting opinions on a_1, they produce significantly[1] more novel elements and more elaborated arguments in a_3 (Jermann & Dillenbourg, 1999). When I conducted this graph with various classes, I used to break for 15 min between a_3 and a_4. Almost invariably, students would come back after the break, still very high in energy, which was reflected by a high participation during a_4. They were eager to defend the position they selected in a_3. Once, I had to postpone a_4 by a week, and it failed miserably—all the energy had vanished, students participated in a_4 without the same engagement as usual. This anecdote illustrates Point 5—this edge has a limited elasticity; stretching it over time leads to a loss of effect.

The ArgueGraph scenario illustrates the relevance of data operators in orchestration graphs. This scenario includes 5 operators, illustrated by red circles on Figure 3.1.

Operator$_1$:	After a_1, an operator aggregates student answers in order to compute the horizontal and vertical position of each learner and produce the opinion map. This is an example of an aggregation operator (Point 14).
Operator$_2$:	Another operator uses the position of each student in order to form pairs of individuals with conflicting opinions, which is communicated to learners during a_2. This is a social operator (Point 16).
Operator$_3$:	For a_3, an operator aggregates—for each pair formed in a_2—the answers that the two peers gave individually in a_1. This is also an aggregation operator. This operator does indeed combine data produced in a1 and in a_2, except that it is placed on the edge $e_{2,3}$, given the transitivity of edges explained in Chapter 1.
Operator$_4$:	For a_4, an operator counts all the answers and justifications per question, for each individual and each team. This aggregation operator produces several pie charts and tables that the teacher can use during the debriefing lecture.
Operator$_5$:	For a_5, an operator produces a list of all data collected per question, which students will use to write their summary.

A graph of data operators constitutes a workflow—in our case, an educational workflow. A central point of this book is that these **operators are not simple implementation details; they translate a pedagogical idea** (in ArgueGraph, the socio-cognitive conflict) **into an operational structure**. They therefore constitute the core element of this chapter. The peer grading process that is implemented in many MOOCs is another example of a sequence of operators: the uploaded documents are assigned randomly to reviewers and distributed, reminders are generated, reviews are collected and processed to compute the grade, and so on. These operators bring peer grading to a scale that was not possible before.

---

[1] Of course, the results provide some evidence that this scenario is effective, but they do not validate the way I model the scenario.

As a corollary to the previous point, **an educational workflow enables heterogeneous activities to be functionally integrated**. In the past, the reification of pedagogical models into incompatible technologies polarized the educational landscape; for example, the constructivism behind LOGO was considered incompatible with the behaviorist grounds of drill and practice software. Technically speaking, they actually were incompatible. Fortunately, the evolution of software architecture has reduced the technical incompatibility between tools—a microworld can be integrated with a collaboration tool, or a drill and practice with a project-based tool. Components from alien environments exchange information; for instance as web services. A physics MOOC in which a quiz measures students' misconceptions can export data to an educational simulation, which will generate experiments that correspond to the detected misconceptions. Passing data across activities corresponds to the idea of **educational workflow** developed in this chapter.

In a MOOC, the workflow conveys digital objects; for example, written essays, environmental data collected by learners, or the comments associated with other reports. Actually, workflows also exist in physical classrooms; when a teacher distributes rock samples to students, collects response sheets from individuals, or asks peers to exchanges their copies, he operates a workflow in the same way Jourdain was speaking in prose. Such physical workflows do not scale up as easily as digital workflows. However, one reason for which paper is still so ubiquitous in classroom routines is that paper workflows have great advantages in terms of visibility and flexibility (Dillenbourg et al., 2011), which could inspire the design of digital workflows. The fact that a teacher can compute these operators on a daily basis (as long as he does not have too many learners) illustrates another key point of this book: **orchestration graphs are not restricted to digital education; they also model standard classroom practices**.

Point 13 Workflows

Along the graph, some students' products in a_i become the inputs of a_j. In between, they are transformed in various ways. In the ArgueGraph example, individual answers are transformed into positions on an opinion map, and then positions are processed to form pairs with conflicting opinions. Interestingly, the definition of a workflow includes the term orchestration, "*an orchestrated and repeatable pattern of business activity enabled by the systematic organization of resources into processes that transform materials, provide services, or process information.*"[2] A workflow describes a sequence of operations on data structures that are performed by humans or by computers. Workflow technologies were not invented for the field of education, but for business processes.

---

[2] https://www.ftb.ca.gov/aboutFTB/Projects/ITSP/BPM_Glossary.pdf

A classic example of a workflow is the processing of an insurance claim. After an accident, the customer fills out a form, which then arrives at the insurance company. First, the company verifies if this customer has a proper insurance contract. If not, the claim is rejected, if so, they check the nature of the accident. An accident without personal injury is sent to department "*car,*" while an accident with personal injury is sent to the department "*persons*." The "*car*" department then has two procedures, one for cars that are still in use and one for those that have been stored in a garage. I don't think I need to further develop the example. The form, as a data structure, circulates from one corporate unit to another. Its path is decided by the processing of the data it contains. These data can then be completed with new data; for example, the payment decision. This processing is performed either by humans and/or by software components that I call "operators" in this book. The workflow is therefore a sequence of operators that corresponds to a kind of algorithm, but at the company level; a component is not necessarily a function, but can be a human activity, or even a full department. An orchestration graph is a workflow that does not model a bureaucratic process as in the insurance claim example, but an educational process.

A workflow is made up of operators that manipulate data structures.

- The workflow handles data structures (tables, files, databases) that contain both the data provided to students (the object of activities) and the data produced by the students (the product of their activities). These data structures also store the traces left by students while performing activities. In addition, they can integrate data coming from outside (e.g., indices produced in real time by the stock exchange).
- The operators transform these data structures from one activity into new structures required to run the next activities. The range of transformation is almost infinite—anything that can be computed.

In the following points, I propose a library of operators classified into 5 categories.[3]

- Aggregation operators gather data for subsequent activities, generally located on a higher plane (Point 14).
- Distribution operators split data for subsequent activities, generally located on a lower plane (Point 15).
- Social operators modify the social structure of activities (Point 16). They rely on social distance criteria presented in Point 17.
- Back-office operators enrich data with external information, including information manually provided by human actors (Point 18).

---

[3] A computational challenge is to express operators in a language that would be independent of the data structures they manipulate. If they could be implemented with a high level of abstraction, they would increase the interoperability among online education platforms, MOOCs, and others.

I present 26 operators, but there could be many more—the set of operators is a fantastic creative space for instructional designers. What follows is similar to the library of edge labels that I presented in the previous chapter; I don't propose it as a fundamental ontology or an exhaustive inventory, it simply illustrates the variety of operators that can be used and hence the variety of orchestration graphs that can be designed. Several operators can be associated with the same edge; the combination of these 26 operators already covers a broad space of possible data transformations.

Table 3.1 The library of operators for orchestration graphs.

Aggregation	Distribution	Social	BackOffice
(A) Listing	(D) Broadcasting	(S) Group formation	(B) Grading
(A) Classifying	(D) User selection	(S) Class split	(B) Feedback
(A) Sorting	(D) Sampling	(S) Role assignment	(B) Anti-plagiarism
(A) Synthesizing	(D) Splitting	(S) Role rotation	(B) Rendering
(A) Visualizing	(D) Conflicting	(S) Group rotation	(B) Translating
	(D) Adapting	(S) Dropout management	(B) Summarizing
		(S) Anonymization	(B) Converting
			(B) Updating

Teacher activities, such as grading assignments, are represented as operators and not as activities, since the graph only describes student activities. Conversely, peer grading is represented as an activity since the students perform it.

Even when computed automatically, an operator may take time to perform data transformation, which must be taken into account in the design of the graph. For instance, team formation algorithms may be computationally expensive. I anticipate that these operators may at some point be operated as web services by external platforms.

Point 14 Aggregation operators

Aggregation operators collect products and traces from a_i and elaborate objects for a_j, often located on a higher plane than a_i. Aggregation is a common way of reusing data produced by students, especially when there are many of them. Counting the number of students who answered correctly is the simplest example of aggregation. Aggregation is one key for scaling up. I describe 5 aggregation operators.

Operator (A) Listing: When associated with the edge e_{ij}, this operator uses the data collected in a_i and produces a list for a_j. For instance, in ArgueGraph, all individual and group opinions are listed in a_4 for the debriefing discussions led by the teacher.

Obviously though, listing all objects does not scale up very well, as a list of 20,000 products would not be very useful for subsequent activities. Aggregation has to be enhanced with data organization, which will determine the way the data can be exploited in the next activity. Therefore, the following aggregation operators include a differentiation component, that is, revealing differences between data or subsets of data.

Operator (*A*) Classifying: When associated with the edge e_{ij}, this operator uses the data collected in a_i and classifies them for a_j. This classification can be done simply or with sophisticated methods.

- In some cases, data natively belong to a category, such as the answers to a quiz or a numerical value between two ranges.
- Students can classify their own data when entering them into the platform. For instance, the learner has to specify that his text response is a "*counter-argument*" versus a "*warrant*," or if the picture of a church that he uploaded illustrates a "*gothic style*" or a "*roman style*." This classification should initially rely on a set of categories, predefined by the graph designer, but the learners can be allowed to expand this set with their own categories; for instance through **tagging**.

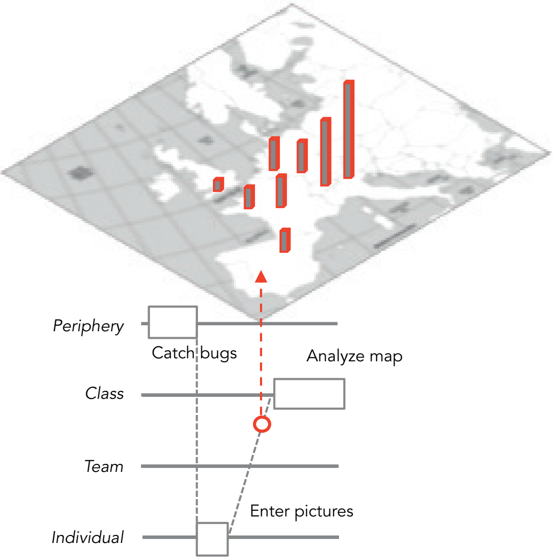


Figure 3.2 For a biology MOOC, 10,000 students have to catch 5 insects around their house with their family or friends (π_4), freeze them, and take two pictures of them—one from above and one from the side. The students then upload the 10 pictures on the website and specify the longitude and latitude of their capture location. The system compares the collected pictures with an existing database of pictures and classifies the 100,000 pictures automatically. It keeps the top 20% of pictures for which the classifier produced the highest confidence value. The remaining 20,000 pictures are used to produce a geographical map of frequency of bug species per country.

- Classification can result from machine-learning techniques. For instance, the teacher manually classifies 100 examples among the 20,000 examples generated by the learners into 4 categories. Next, an algorithm classifies the 19,900 remaining examples based on their similarity with those classified by the teacher, using supervised machined learning methods, as in Figure 3.2.
- Other algorithms, called unsupervised machine learning, compare data and form clusters of similar data. Some classification algorithms can be also applied to texts (e.g., using latent semantic analysis), as well as to pictures (based on computer vision techniques).
- If only a human eye can perform a meaningful classification—for example, classifying an insect as being dangerous or not, — crowdsourcing methods (see distribution operators—Point 15) enable the MOOC designer to cope with scale: for instance, the algorithm randomly selects 10 pictures for each of the 5,000 students who have to classify them. Crowdsourcing methods may produce unreliable results, but several methods exist for calculating the rate of inter-judge agreement, which gives an indication of the reliability of the results produced.

Operator (A) Sorting: When associated with the edge e_{ij}, this operator uses the data collected in a_i and ranks them or sorts them for a_j. Data can be sorted based on one or several criteria; for example, numerical value, alphabetical order, number of characters, size of the group, age of the author (if available), longitude and latitude of the author (IP address), distance from the correct solution (e.g., measured by Levenshtein), or "edit distance."[4] Again, if sorting can only done by a human judge, a crowdsourcing algorithm may randomly select two pictures among 10,000 and ask a learner which one is the best, repeating this operation ten times per user with thousands of them, and hence ranking the 10,000 pictures.

Operator (A) Synthesizing: When associated with the edge e_{ij}, this operator uses the data collected in a_i and produces a synthesis for a_j. A large set of data, which has perhaps been previously categorized or sorted, can be replaced by simpler data such as the frequency per category, the mean and standard deviation, or approximating a cloud of data by a trend curve (Figure 3.3) that corresponds (or not) to the theoretical model. Most arithmetic operators can be used here; namely, any synthetic way of describing data distribution.

Operator (A) Visualizing: When associated with the edge e_{ij}, this operator uses the data collected in a_i and produces a visualization for a_j. The results of previous operators are presented as lists, histograms (e.g., report-

---

[4] The edit distance is the number of atomic character transformations necessary to transform one expression into another.

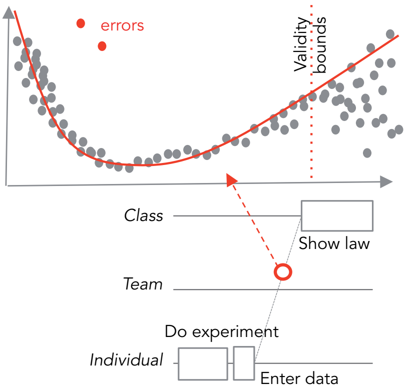


Figure 3.3 In a physics MOOC, students have to take an egg, weight it, and drop it from an altitude of between 100 and 200 centimeters. When the egg lands, they measure the distance between the splashes that are the furthest away from each other. Each student enters the values of the weight, altitude, and distance after impact. The system produces graphs where every experiment appears as a dot. The curve shows the behavior predicted by the theory. The teacher points out which data are measurement errors (red dots) and those poorly explained by the scientific model (on the right of the dotted red line).

ing answers from clickers), geographical maps (Figure 3.2), graphs (Figure 3.3), or timelines (Figure 3.4). The range of possible visualizations is infinite. I won't develop the art of visualization here, as there is abundant literature about it.

A modeling language is mainly descriptive, but I have nevertheless dared to produce some design recommendations.

- **The features of the visualization influence what information students have to process in the next activity**, what they will comment on, discuss, or discover, as well as what the teacher will be able to point out in a subsequent debriefing lecture. The visualization has to be designed with this didactic purpose in mind, that is, how to pedagogically exploit the graphical representation in the next activity, not just for the sake of producing fancy visualizations.
- Students are especially engaged when **their own data** are visualized. These can be the products/traces they produced in previous activities: "*my*" answers, "*my*" comments, "*my*" products, and so on. It would be politically correct to suggest making data (semi-) anonymous here, but this would reduce the engagement effect, since engagement is due to the fact that students see their own name on the data visualization. Several solutions exist, however; for example, replacing a student's name with a pseudo or designing the interface so that the student can see his own name, but not the name of his peers.

- An aggregation operator enables powerful activities when a differentiation operator is used in the previous activity. This design principle will be developed in Point 19.

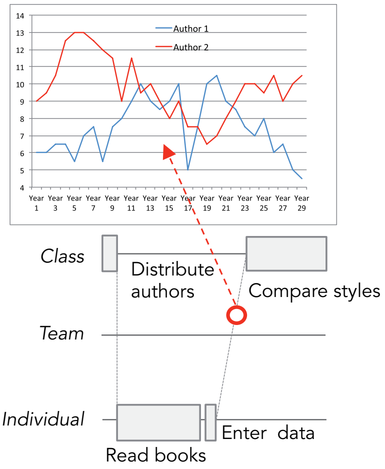


Figure 3.4 For a MOOC in literature, each student chooses a novel published between 1930 and 1980 and counts the number of pronouns and verbs per sentence between pages n and n+5, where n is randomly chosen by an operator. He chooses the author among a list of 10, enters the book, year, and pronoun and verb frequency. Aggregated data are reused in the next lecture, revealing how these authors influenced each other's style.

Point 15 Distribution operators

Distribution operators distribute data from a_i to the participants of a_j, which is generally located at a lower plan. Distribution means either sending data to students (push mode) or giving students access to these objects (pull mode). The peer grading solution in MOOCs is a distribution operator (followed by an aggregation operator applied on collected grades). Several distribution operators can be combined; I describe 5 of them.

Operator (*D*) Broadcasting: When associated with the edge e_{ij}, this operator delivers the same data to all learners performing a_j. This is the simplest form of distribution.

However, the pedagogical interest emerges when different data are distributed to different individuals; the differences between the objects distributed to individuals will influence the interactions that these individuals will have later on, when working in teams to integrate these data. This pedagogical trick is implemented by coupling a distribution operator with an aggregation operator. I refer to this coupling as the distribution-aggregation rebound, a design pattern further explained in Point 19.

Operator (*D*) User Selection: When associated with the edge e_{ij}, this operator lets users choose which subset of data they will be working on in a_j. Each learner or each team selects his/their objects among those proposed by the system. The selection can be done either freely (if it does not matter that some subsets are not selected by any learner or if some are selected multiple times) or among the objects that remain available after the choices made by the students who decided fastest (i.e., if a subset cannot be selected more than *n* times or if all subsets must be selected at least once). The advantage of letting learners select the object of their activity is a potential motivation gain—they can pick what they like most. This motivation effect is important for long activities; it won't impact much on the performance on one exercise, but could be critical for projects lasting weeks. But are students really able to choose, for instance, the exercises they need? There is vast literature on this topic, but in a nutshell, they converge towards the fact that making relevant choices requires already having some knowledge of the topic.

Operator (*D*) Sampling: When associated with the edge e_{ij}, this operator assigns a different subset of data to individuals or teams for a_j. A subset of data is assigned randomly or through an algorithm that guarantees that every subset is assigned the same number of times. One distribution method is stratified sampling; for instance, if 1,000 students have to analyze one city in the world, the sampling operator could, for example, distribute cities in such a way that each continent is represented by a number of cities proportional to its position in economic rankings.

Operator (*D*) Splitting: When associated with the edge e_{ij}, this operator assigns a different subset of data to each individual within a team for a_j. Typically, 3 individuals in the team receive a subset of the information necessary to solve the problem they have been assigned to. The way this information has been split is not random; it has to be designed in such a way that, during teamwork, each learner can complement the other team members. This is the famous "Jigsaw" method; since no one is able to solve the problem alone, they have to interact intensively (Point 18). The subsets received by individuals may partly overlap—a little bit of redundancy will facilitate teamwork.

Operator (*D*) Conflicting: When associated with the edge e_{ij}, especially an edge with the contrast label (Point 10), this operator assigns conflicting subsets of data to individuals within a team for a_j. This operator triggers a key process in collaborative learning—the resolution of cognitive conflict—where team members receive conflicting pieces of evidence. For instance, they receive documents that defend opposite viewpoints, data that support or reject the same hypotheses, or pictures or papers that present contrasting evidence. Learning occurs from the elicitation of knowledge through argumentation, as well as from the de-centration process, that is, the cognitive effort necessary to adopt the viewpoint defended by one's peer. A conflict can also be created between teams; for instance, different teams get opposing sources of evidence. In fact, my experience is that argumentation tends to be even stronger in these situations, because inter-team opposition increases intra-team cohesion; individuals defend their team decision in a stronger way than they would defend their personal position.

Operator (*D*) Adapting: When associated with the edge e_{ij}, this operator chooses the most relevant material for an individual or a team in a_j. This is the oldest idea in computer-based learning; the learning material assigned to a learner is chosen based on his knowledge, skills, motivation, or learning style. Adaptation mechanisms are developed in Chapter 6. The available activities are distributed (i.e., assigned to individuals or teams), creating de facto subclasses. The criteria used to assign activity a_i to student s_n should come from empirical findings; namely, the so-called "aptitude-treatment interactions" law which have shown that the effects of a pedagogical intervention vary according to the individual features of learners. This paradigm led to sophisticated research in education; specifically, adaptation to student features such as "learning styles." In the field of artificial intelligence, the effort was to infer individual misunderstandings through student modeling processes (Dillenbourg & Self, 1992). In practice however, the most important feature that requires adaptation is the individual level of prerequisites. This should be a priority of MOOC developers.

Point 16 Social operators

Social operators produce a social structure as an output. A social structure maps the elements across social planes: individuals to teams, with or without roles, individuals to classes, teams to teams, teams to classes, and so on. Social operators are the key for scaling up collaborative learning methods.

Operator (*S*) Group formation: When associated with the edge e_{ij}, this operator uses the data collected in a_i to select the learners that will constitute teams for a_j. The first operator constitutes groups of students for the next activities. It can also be used to build subclasses. It requires 4 parameters.

FormGroups (*S*, Group size, Distance criterion, Min/Max)

- *S* is the class of students to be partitioned into groups. Groups do not usually overlap.
- **Group size:** The second argument is the number of learners per group, with or without flexibility. Small groups (2 to 4) are recommended for complex convergent tasks, while larger groups (6 to 10) are better suited to divergent tasks (brainstorming), or tasks that can be easily divided into subtasks. Larger groups require more management, which is fine if management skills belong to the target learning outcomes of the graph, but biases performance measures otherwise. Very practically, making larger groups offers the advantage of reducing the number of assignments to

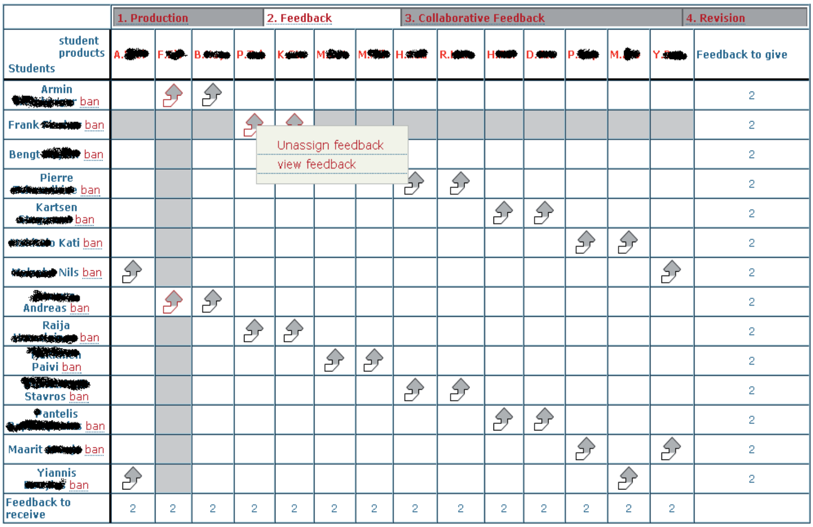
be graded. The group size can be expressed by two values, the lower and upper limits: [4 6] means between 4 and 6, [5 5] means only 5. Given the high dropout rate of MOOCs, I recommend keeping group size as a flexible variable; this will enable groups to continue their work despite missing members.

- **Distance criterion:** To determine the homogeneity or heterogeneity of a group, the operator needs to know the criterion for measuring the distance between two individuals; this distance will be minimized for forming homogeneous teams or maximized when forming heterogeneous teams. The choice of this criterion is a key pedagogical decision—first, the designer chooses the nature of teamwork activities to be set up, and from that, he infers the way teams have to be formed; the differences among team members of created teams will shape their interactions (Point 19). As the choice of this criterion is pedagogically very important, I will explore it further in the next point.
- **Min/Max:** For any distance criterion, this parameter specifies if the distance has to be minimized, creating homogenous teams, or maximized, creating heterogeneous teams. This may require a substantial effort to compute. It is also possible to form groups in which the distance is below/above a certain threshold, as is the case for the ZPD edge (Point 9); the learner s_1 must be more advanced than learner s_2, but nonetheless in the zone where s_2 will be able to perform with the help of s_1.

Group formation is sometimes a natural alternative to a distribution operator; for instance, instead of distributing different pieces of information to team members (the "splitting" operator), one can create teams with students with different background information.

Operator (*S*) Class split: When associated with the edge e_{ij}, this operator uses the data collected in a_i to choose the learners that will constitute subclasses for a_j. This is not very different from group formation, except that, as explained earlier, a subclass is not a team or a set of students who have to collaborate on a task, but a set of students who do the same activity at the same time. For instance, half of the class may participate in a debate with the teacher, while the other half does lab exercises, or half of the class calculates the regression in R and the other half with SPSS. The parameters of this operator are the same as the parameters used by group formation operators.

Operator (*S*) Role assignment: When associated with the edge e_{ij}, this operator assigns roles to individual team members for a_j. The pedagogical value of roles during teamwork has been described in Point 8. When a designer decides to structure teamwork with roles, all teams generally get the same set of roles, one role being assigned to each team member. This operator has the capacity to be more sophisticated. What happens if the social operator has to form teams with 4 roles, but the total number of students in S is not a multiple of 4? How should the system handle teams of 2 and 5? If there are more members than roles, a role can be duplicated, that is, anticipated in the workflow that two members will play the same role in the same team. If there are fewer members than roles, a "joker" can be allocated to the team (Dillenbourg & Hong, 2008); a team whose Role-X member is missing gets access to the products of students playing Role-X in any other team. This solution was implemented for the graph ConceptGrid, inside the online environment ManyScripts (Dillenbourg & Hong, 2008). This web environment includes an orchestration tool for teachers to follow teamwork and modify group formation, if necessary. Another graph is based on peer feedback: A writes a text, *B* gives feedback to *A*, then *A* has to revise and resubmit it. Next, the revised document is itself reviewed again by *B*. The distribution on assessments is automatic, but can be manually modified in case of dropout, by editing the table presented in Figure 3.5.



	1. Production				2. Feedback			3. Collaborative Feedback							4. Revision
student products Students	A.[redacted]	F.[redacted]	B.[redacted]	P.[redacted]	K.[redacted]	M.[redacted]	M.[redacted]	H.[redacted]	R.[redacted]	H[redacted]	D.[redacted]	P.[redacted]	M.[redacted]	Y.[redacted]	Feedback to give
Armin [redacted] ban		⇧	⇧												2
Frank [redacted] ban				⇧	⇧										2
Bengt [redacted] ban															2
Pierre [redacted] ban								⇧	⇧						2
Kartsen [redacted] ban										⇧	⇧				2
[redacted] Kati ban												⇧	⇧		2
[redacted] Nils ban	⇧													⇧	2
[redacted] Andreas ban		⇧	⇧												2
Raija [redacted] ban				⇧	⇧										2
[redacted] Paivi ban						⇧	⇧								2
[redacted] Stavros ban								⇧	⇧						2
[redacted]antelis [redacted] ban										⇧	⇧				2
Maarit [redacted] ban												⇧		⇧	2
Yiannis [redacted] ban	⇧												⇧		2
Feedback to receive	2	2	2	2	2	2	2	2	2	2	2	2	2	2	

Figure 3.5 Orchestration tool for the teacher in the environment ManyScripts (Dillenbourg & Hong, 2008).

Operator (*S*) Role rotation: When associated with the edge e_{ij}, this operator reassigns roles to team members in a_j that were previously assigned in a_i. This operator complements the role assignment operator by redistributing roles that have previously been distributed; the criticizer becomes the defender, the group leader becomes the note taker, and so on. Rotation is more complex if the number of roles is not the same as the group size. The timing of rotation can be fixed (e.g., every week), task-based (e.g., every exercise), or criterion-based (e.g., if a role holder fails). Combined with a loop structure, this operator can implement, for instance, the "reciprocal tutoring" graph (Figure 3.6); because roles rotate at each iteration, the loop and role rotation operators have to be combined. The reciprocal teaching graph is well known because of its empirical effects (Brown & Palincsar, 1987). The loop operator is combined with a role rotation. Two students who have a deficit in reading comprehension read a text together. One child reads a paragraph aloud, then the second child asks him a few questions, such as, "*What's the main idea of this paragraph?*" or "*What do you think will come next?*" At the next paragraph, the roles are rotated (round arrow). This script led children with a reading deficit to reach the average level of their class within a few weeks.

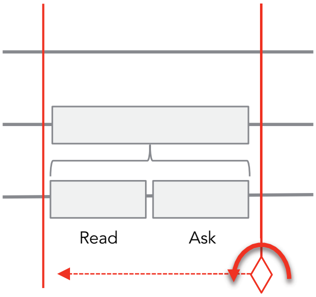


Figure 3.6 Reciprocal tutoring (Csar & Brown, 1987).

Operator (*S*) Group rotation: When associated with the edge e_{ij}, this operator forms teams for a_j with students who played the same role in their respective teams in a_i. This operator is the inverse of roles rotation. Roles remain assigned to the same individuals, but the individuals change group. In the geology scenario (Point 8), teams of four different experts are sometimes interrupted. Learners playing the role "*Mayor of San Francisco*" in their team constitute a new team with the mayors from every team. For 10 minutes, they share their strategy or knowledge, that is, how to play the role of a mayor in their team and how to convince peers, and then return to their original team. In a MOOC with 10,000 students and teams with 4 roles, these "expert" groups would have 2,500 members, which would require the operators to split them into many groups.

Operator (*S*) Dropout management: When associated with the edge e_{ij}, this operator attempts to recompose teams for a_j from which some students dropped out in a_i. The careful constitution of teams through social operators can be jeopardized by the high rate of dropout observed in MOOCs. It may be a good idea to start the team activity on the third week, since the dropout rate is already much lower (many students drop out after the first week, when they realize the online course is too difficult or does not correspond to their interests). If the team activity is a few hours long, spread over a week, it can be postulated that most students who start the activity will complete it. If the activity spreads over several weeks, the probability is high that some team members will drop out and has to be anticipated in the workflow. If the loss of a member increases workload for rest of the group (which is unfair; e.g., collect 10 instances of *X*), the workload can be reduced proportionally to the new group size. If the missing person was holding a necessary piece of information, the joker solution can be activated. This operator reaches the limit of what can be automated; the teachers who manage the MOOC should then be given access to an interface that complements automatic group formation with manual adjustments (Figure 3.5).

Operator (*S*) Anonymization: When associated with the edge e_{ij}, this operator anonymizes a subset of the data collected in a_i in order to use them in a_j. As soon as operators handle user data, privacy concerns appear. This problem also exists in on-campus education (are individual grades publicly displayed?), but the scale of MOOCs and the new possibilities of monetizing private data have brought this concern to the public space. One solution is to completely anonymize data, but this has drawbacks, such as the difficulty of coaching users individually and the impossibility of delivering certificates with the student's name. An alternative is to selectively anonymize the subset of the data that raises concerns; for instance, the data that will be displayed by a visualization operator. Such a "selective anonymization" can actually be difficult. This operator could be classified in the back office category (Point 18), but I put it among the social operators, because it does indeed play a social role. In many cases, it will be used in conjunction with another operator; namely, aggregation operators.

Point 17 Social distance criteria

This point is a parenthesis in the review of the library of operators aimed at further developing the social operators. The social operators select team members based on the similarities or differences between their members. The criteria used by these operators for measuring the differences between individuals will influence the interactions in the formed teams. I call "social distance" the measure of differences between individuals. Here are some criteria for measuring social distance.

Social distance criterion—Level: Two widespread practices are to form groups with students that have the same level of skills knowledge or, conversely, groups with different levels. The level of skills or knowledge can be measured during previous activities, through a pretest (see Point 4), or collected through a questionnaire (i.e., as a self-estimate). Forming homogeneous groups allows adaptive instruction, with high-level teams getting more difficult tasks. Conversely, heterogeneous teams may result in richer interactions. Some teachers fear that better students would be wasting their time with weaker ones. However, studies show that by elaborating explanations, the better student often improves his knowledge (Webb, 1989, 1991). However, a difference of level among peers that is too large could for instance generate a 'free rider' effect (see Point 21).

Social distance criterion—Knowledge type: The difference between learners may not be their level of knowledge, but rather the nature of their knowledge. Hoppe & Ploetzner (1999) designed a creative scenario during which they identified, through previous activities, students who mostly solve physics problems in a quantitative way, applying formula, versus those using more qualitative reasoning. Then, they paired students with different knowledge types and gave them a physics problem that cannot be solved by using only quantitative or only qualitative knowledge. This constitutes a kind of Jigsaw scenario, exploiting natural differences among learners instead of generating artificial differences with a distribution operator.

Social distance criterion—Background: Many classes involve students with different backgrounds; for example, chemists and biologists, or architects and civil engineers, but also men versus women, young versus old, single child versus child with siblings, and so on. This class heterogeneity is often described by teachers as being a difficulty or a pitfall, but it can be turned into an opportunity; for instance, by taking advantage of the multiplicity of viewpoints and by trying to foster interdisciplinary skills.

Social distance criterion—Opinions: As illustrated by the ArgueGraph scenario, opinions can be collected online and then used as criterion to form consensual or conflicting groups.

Social distance criterion—Geography: Teams can be formed based on the similarity or difference of the locations where students live or where they come from. Homogeneous teams: If a chemistry activity is about ski wax, it should be assigned to countries were skiing matters. Heterogeneous teams: For many societal issues, comparing how problems are handled in Denmark and Tanzania opens up great learning opportunities.

Social distance criterion—Time zone: If the graph includes synchronous interactions; for example, a chat tool or using a concept map editor, homogenizing the time zone within teams is practical. Conversely, if a task can be split into subtasks to be sequentially processed, teams can be formed across time zones in such a way that there is always someone working on the project.

Social distance criterion—Friendship: If data about friendship is available; for example, because the MOOC environment includes a social network, this can be taken as an element to minimize or to maximize in team formation.

Let's conclude with a few words about the pair formation algorithm. We have seen that in some cases the social distance will be minimized, forming homogeneous groups, and in other cases, it will be maximized, forming heterogeneous groups. Finding two individuals with the maximal distance is easy. The difficulty is illustrated in the graph on the left of Figure 3.7: if s_1 is associated with s_4, their distance is maximal, but this implies that s_2 would then be associated with s_3 despite their low distance. The graph on the right illustrates the solution for maximizing distance: the operator calculates the partition of S in teams of n students that maximizes the average distance between team members. Calculating all possible partitions of S into teams of n students is computationally expensive when S has 10,000 students.

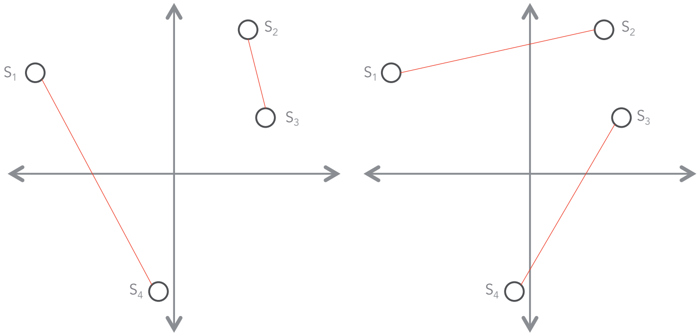


Figure 3.7 Examples of pair formation algorithms.

Point 18 Back-office operators

Various actors can add information to data structures, between two activities. Feedback is the obvious case. These operators constitute the back office of a MOOC. Some information cannot be provided by human agents, but by web services. Here are some examples of these operators.

Operator (*B*) Grading: When associated with the edge e_{ij}, this operator uses the data collected in a_i and produces a grade for a_j. This operator completes a data structure, at any π, with a grade—either a label or a numerical value. If a teacher does the grading, the operator may be simply a field where the value can be input, or a table where the teacher enters a value for each criterion specified in the grading rubric. In other cases, the grading is done automatically; this applies not only to quizzes or numerical answers, but also to programming exercises, graded automatically by a parser, and even for essays. Automatic essay grading is a very controversial issue, but it can be used in combination with human grading.

Operator (*B*) Feedback: When associated with the edge e_{ij}, this operator completes a data structure, at any π, with feedback or comments. The feedback can be piece of text, a numerical value, some picture annotations, or any elements added to a piece of code. The feedback can be generated manually by the teacher, which requires a specific interface, or by an automatic grader, as often used in MOOCs that concern programming languages.

Operator (*B*) Anti-plagiarism: When associated with the edge e_{ij}, this operator analyzes the data collected in a_i in order to measure the probability of plagiarism. Plagiarism can be detected between two activities, and the result can then be processed by edge controls (Point 7).

Operator (*B*) Translating: When associated with the edge e_{ij}, this operator translates the data collected in a_i in one language into another language for a_j. An operator could use an external service for translating the assignments written by students in another language into the main language of a course.

Operator (*B*) Updating: When associated with the edge e_{ij}, this operator updates data with external sources between a_i and a_j; for example, stock exchange values or weather data. Some documents or assignments may include variables, with a dynamic link to web sources of information (RSS Feeds, APIs) in such a way that they are automatically updated.

Operator (*B*) Converting: When associated with the edge e_{ij}, this operator converts the data collected in a_i into equivalent data for a_j. Automatic conversion may be relevant if 500 students upload an architecture project defined in centimeters and 200 do the same with projects defined in inches; if some geology students describe an oil reservoir in depth (3,000 meters) while others use altitude (–3,000 meters); or if students upload business plans expressed in 56 different currencies.

Operator (*B*) Summarizing: When associated with the edge e_{ij}, this operator uses the texts collected in a_i and produces a summary for a_j. As controversial as automatic translation, automatic summary could be used for enriching an assignment with a summary. Less controversial are statistical summaries (e.g., the number of words).

Operator (*B*) Rendering: When associated with the edge e_{ij}, this operator uses the graphic objects collected in a_i and renders them as a list in a_j, turning any object defined mathematically into a 3D object, with textures, shadows, and so on.

These two last examples apply only to some objects (respectively texts and 3D objects), and illustrate the fact that the list of operators could easily be expanded.

Even if some of these operators are mere technical actions, they can bring added value to learning activities. Some of them require external tools, either user interfaces for humans who provide the data (e.g., a feedback tool for the teacher) or software interfaces for other applications (web services). I expect to see the development of many services around MOOCs and online education, that is, companies that do not provide online courses, per se, but various services to the companies who provide the courses or to the students who take them. This is already the case for online proctoring services; some companies offer services such as verifying the identity of learners when they pass an exam online and minimizing their cheating opportunities.

Point 19 Patterns of operators

Is an orchestration graph only relevant for the learning objectives for which it has been designed? Let's consider ArgueGraph (Figure 3.1). It has been implemented in an environment that allows a teacher to modify the questions that students will argue about. However, this does not imply that any topic can be taught with this graph. It is relevant if the learning goals target declarative knowledge, but it is not appropriate for procedural knowledge and problem-solving skills. It is suited for domains where there is no clear right or wrong response, that is, where pros and cons of different solutions have to be balanced. This includes many domains, but does not make ArgueGraph applicable to any domain. In other words, a graph is not applicable to any content, but some graphs can be reused across several domains, keeping the global structure while modifying some details. This point addresses the generalizability of graphs.

Despite the fact that ArgueGraph is not universal, it is based on a pedagogical idea about how to trigger cognitive conflicts, which is reusable in other graphs. For instance, a Simquest[5] graph developed by Gijlers and de Jong (2005) for learning from simulations follows the same idea; students express hypotheses about the scientific phenomena to be simulated, and the system forms pairs of students who associated conflicting hypotheses to the same experiment. In other words SimQuest and ArgueGraph appeared independently from each other as two instances of a general idea, **natural conflict,** based upon already mentioned theories of socio-cognitive conflict (Mugny & Doise, 1978). This pattern is based on a single operator, team formation; the operator forms pairs of students who naturally bring their conflicting hypotheses or opinions to the activity. In a MOOC, the set of students is naturally rich in divergences, as the students live in different countries, different climates, and in different political and economical systems. When the natural class diversity is not sufficient, the graph may purposely be increased to "engineer" the conflict; different data, information, or viewpoints are distributed to different team members in order to induce the expected conflict. This pattern, **induced conflict**, combines two operators, a team formation operator and a distribution operator.

---

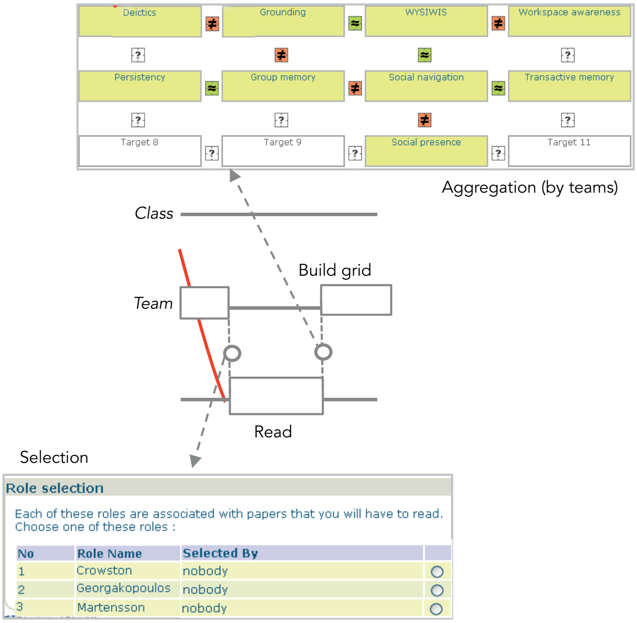
[5] This is the name of the guided-discovery platform in which the graph runs.

The conflict-raising mechanism illustrates what I refer to as a design pattern: **a design pattern is a pedagogical mechanism expressed as an abstract subgraph**. It is a subgraph because it involves one or a few operators within the graph, not the whole graph. It is abstract when the specific content elements are extracted away and only the core pedagogical idea is kept.

Another well-known pattern is the **Jigsaw pattern**, illustrated by the geology graph presented in Point 8. The pedagogical idea is to strengthen interdependencies between team members. The pattern functions by distributing information to individuals and asking them to integrate their individual information in order to perform a task that none of them could perform without the information provided by the other teammates. Each team member has to assimilate his data or information in order to integrate it with contributions from the rest of the team. This assimilation activity can be an individual activity, for instance, reading a text (as in ConceptGrid example below) or analyzing the data. Assimilation can also be supported by having expert groups; that is, activities that bring together students who play the same role across different teams (Figure 1.23). The integration activity can also be a problem-solving task. In conceptual domains, it can be to build a concept map.

In an orchestration graph, these two design patterns, conflict resolution and Jigsaw, take the form of a **distribution-aggregation rebound**, illustrated in Figure 3.8; it connects 3 activities with a distribution operator followed by an aggregation operator. The pedagogical idea behind this rebound is the following: if a distribution operator is applied between π_1 and π_2, the differences between the objects distributed to individuals will influence the interactions that these individuals have to engage with later on, when working in teams to integrate the data. I have depicted this rebound with two examples of graphs. They illustrate the fact that the notion of an operator is not a technical or operational detail of graphs, but that operators actually translate pedagogical ideas.

The ConceptGrid graph (Dillenbourg & Hong, 2008) is an instance of Jigsaw patterns that illustrates the distribution-aggregation idea. Its goals concern declarative knowledge; namely, domains where students have to acquire a certain number of concepts and relate them to each other. Each team has to build a concept grid—a sort of concept map. Each team is composed of several roles (the number of roles can be determined by the teacher) and each role necessitates reading several papers (the number of papers can be determined by the teacher) that correspond to the selected role. Typically, a student will play the role "Piaget" by reading papers from Piaget. Each student selects a role that has not yet been selected by another team member, and the system simply distributes readings assigned to each role. Then, when each student has learned about a subset of concepts, the team has to build a grid in such a way that students can define (text entry) the relationship between two grid neighbor concepts. The way in which concepts are distributed among team



Role selection

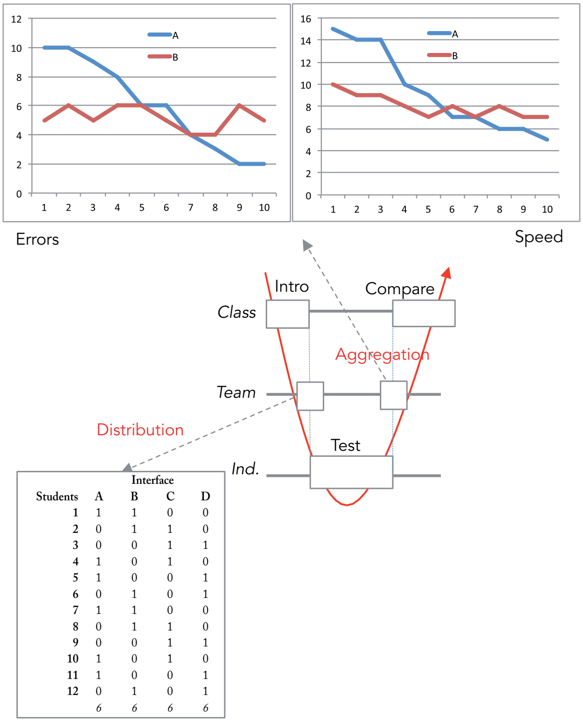
Each of these roles are associated with papers that you will have to read.
Choose one of these roles :

No	Role Name	Selected By	
1	Crowston	nobody	○
2	Georgakopoulos	nobody	○
3	Martensson	nobody	○

Figure 3.8 Distribution-aggregation rebound π_2-π_1-π_2.

members will determine who explains which concepts to whom in the grid construction activity.

The concept grid illustrates a sequence $\pi_2-\pi_1-\pi_2$, but the distribution-aggregation rebound can also be applied to $\pi_3-\pi_1-\pi_3$. Distributing different objects for an individual activity leads students to elaborate different products. The teacher may then collect and exploit the differences between the students' products during a debriefing lecture. The pedagogical idea behind is that by carefully designing the way data are distributed, the teacher determines the range of objects he may use in the next activity. This allows, for instance, for implementing the 'contrasting cases'method, which is efficient for guiding the students' inductive reasoning (Schwartz & Bransford, 1998). I have illustrated this approach with a fictitious graph for teaching human-computer interaction, inspired by a scenario I implemented on a small scale many years ago. The teacher proposes 4 versions of a website in which users order train tickets. Each of the 10,000 students has to order 5 fake tickets with two of the four versions of the website and then fill in a usability questionnaire. The system distributes interfaces to students in such a way that (1) all interfaces are tested by the same number of students, and (2) 50% of the students test *A* before *B* and 50% the other way around. The aggregation operator produces a comparison of the task completion time and the number of errors on each interface. It creates contrasted graphs, such as Figure 3.9, where we can



	Interface			
Students	**A**	**B**	**C**	**D**
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	1	0
5	1	0	0	1
6	0	1	0	1
7	1	1	0	0
8	0	1	1	0
9	0	0	1	1
10	1	0	1	0
11	1	0	0	1
12	0	1	0	1
	6	*6*	*6*	*6*

Figure 3.9 Distribution-aggregation rebound π_3-π_1-π_3.

see that interface *B* generates fewer mistakes at the beginning than *A*, but that the error rate decreases faster with *A*.

The reciprocal tutoring graph presented in Point 16 illustrates the **mutual regulation pattern**, which is relevant for problem-solving tasks that require heuristic knowledge. In this graph, learner s_1 reads a paragraph aloud, after which, learner s_2 asks him comprehension questions. These two roles are switched at each paragraph. The goal is the acquisition of comprehension monitoring skills. For instance, when solving equations, students combine procedural knowledge (how to manipulate algebraic expressions) and heuristic knowledge (which operator to apply when). In many problem-solving tasks, activities may similarly be discriminated against at task level, for basic manipulations, and for the regulation of task-level activities, called metacognition. The mutual regulation patterns work as follows: for one problem, the student is asked to regulate (control, monitor) the activities of his peer, which is easier than to self-regulate, since in self-regulation, the cognitive load induced by task-level actions and the cognitive load induced by regulation is cumulated. For the next problem, the operator/regulator roles are switched.

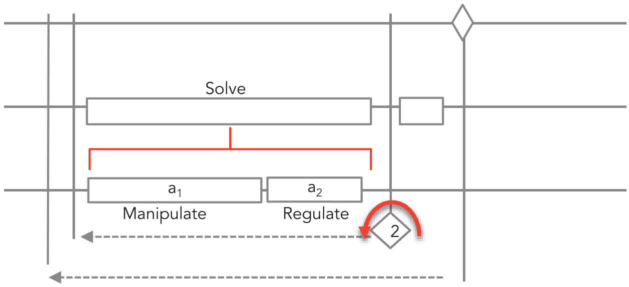


Figure 3.10 Mutual regulation.

Experiments revealed that if a student regulates his peer, he becomes better able at self-regulation (Blaye, 1988).

In this mutual regulation pattern, the team activity is split into the task level and the regulation level in order to make regulation explicit. In the Jigsaw pattern, the team activity is split into different subsets of concepts in order to increase the need for mutual explanations. In the conflict resolution pattern, the team activity is split into opposite viewpoints in order to trigger the argumentation required to bypass this conflict. Across these three patterns, there is an even deeper idea in common, which I have named **SWISH.** This acronym stands for Split Where Interaction Should Happen (Dillenbourg & Hong, 2008): **the differences created among team members determine how they will interact in a collaborative task in order to reach a shared solution despite their differences.** Let's consider a course on economics. Two learners receive data concerning the level of unemployment and the level of taxes across countries. The first learner gets data from 10 countries where there is a negative correlation between tax and unemployment. The second learner gets data from 10 countries where there is no correlation between these variables. This difference between the data set assigned to each learner will lead the pair to discuss the liberal hypothesis according to which taxes reduce employment. The nature of the difference induced by the graph operator (by the data) determines the nature of interactions among peers.

The idea of increasing divergence among learners may seem contradictory to the fact that collaborative learning is often defined as the process of constructing and maintaining a shared understanding of the task (Roschelle & Teasley, 1995). Why then would the task distribution created by SWISH be a good thing for team learning? Actually, learning does not come from the fact that students understand each other, but rather from the effort they employ in order to develop this shared understanding *despite* their differences (Schwartz, 1995). Distributing data is the way to "design" these differences and hence to determine the "*effort towards a shared understanding*". The term "effort" refers to the intensity of interactions, namely explanations, argumentation, and mutual regulation. For tuning the collaborative effort, a graph creates differences ("split") among the students and, subsequently, the interactions necessary to maintain collaboration produce the desired learning outcomes. Increasing collaborative effort may, of course, also damage collaboration (Dillenbourg, 2002). As is often the case in educational design, a trade-off is required; on the one hand, natural conversation tends to minimize collaborative effort (Clark & Wilkes-Gibbs, 1986), on the other, collaboration breaks down if understanding one another becomes too painful. Educational design searches for the sweet spot between easiness and workload, the "optimal collaborative effort" (Dillenbourg & Traum, 2006).

As it is common to several patterns (conflict, Jigsaw, mutual regulation), SWISH can be described as a more abstract pattern. Let's imagine that an operator computes the difference between members of a team. This difference can be a difference of viewpoint, opinion, knowledge, or based on most of the criteria used in Point 17 to compute social distance. Let's call Δ_i the difference among team members at the end of activity a_i (Figure 3.11). It could be a value that ranges between 0, in the implausible case where learners have perfectly aligned viewpoints, and 1 in the equally implausible case where the two learners do not have a single point of commonality. SWISH describes the variations of Δ_I; in team activity a_1, there is some natural divergence (Δ_1) among team members—they rarely have exactly the same knowledge or the same understanding. An activity a_2 is then introduced that purposely increases the divergence inside teams (Δ_2); for example, reading papers that defend conflicting hypotheses. If the next activity a_3 requires teams to build a shared solution (i.e., if Δ_3 has to be small), the effort necessary to minimize Δ_3 is therefore more substantial than it would have been, if a_2 had not increased their divergence. In other words, the SWISH pattern increases "the effort towards a shared understanding" (Schwartz, 1995), this increase

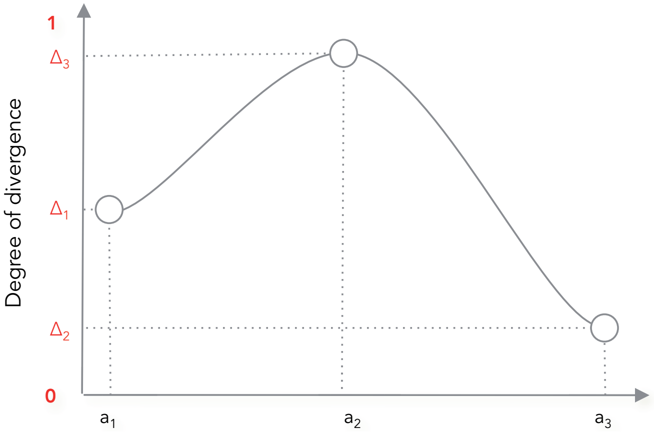
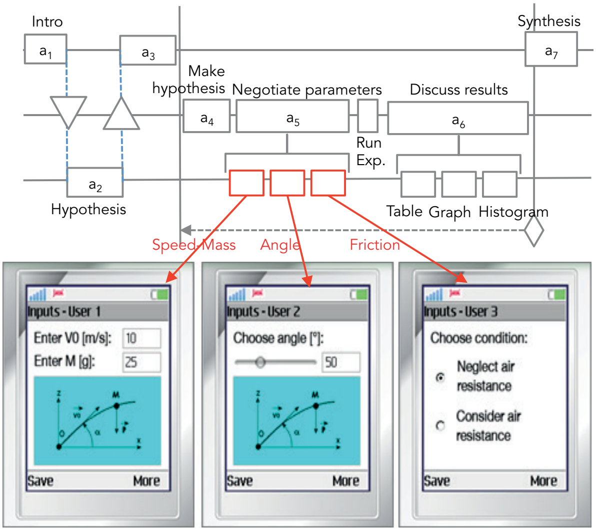


Figure 3.11 The SWISH principle, with the degree of divergence in a team shown vertically.

being summarized as $\Delta_3 - \Delta_2 > \Delta_3 - \Delta_1$. The mere fact of increasing Δ_2 does not increase the learning effort, per se, if a_3 does not constrain teams members to build a shared solution, that is, to reduce Δ_2 to Δ_3. Of course, if Δ_2 is too high, the interactions will simply break down. In Jigsaw graphs, Δ_2 is increased by providing peers with different information. In conflict graphs, Δ_2 is boosted by providing students with conflicting evidence. ArgueGraph forms teams in order to maximize Δ_2.

SWISH is not only about the effort intensity, but also about the nature of interactions; the difference created in a_2 will determine what students have to argue about in a_3, as illustrated below. Figure 3.12 describes another graph that applies the SWISH principle. Teams of 3 learners conduct experiments through a simulation distributed over 3 phones. The graph addresses some pitfalls of enquiry-based learning (de Jong & Van Joolingen, 1998):

- Lack of clear hypothesis: individuals have to express hypotheses (a_2), which forms groups with conflicting hypotheses (Gijlers & de Jong, 2005).
- Inconsistent experiment design: in a_5, each team member sets the parameter value on his phone, which forces discussing these values explicitly.



Inputs - User 1
Enter V0 [m/s]: 10
Enter M [g]: 25

Inputs - User 2
Choose angle [°]: 50

Inputs - User 3
Choose condition:
● Neglect air resistance
○ Consider air resistance

Figure 3.12 The Wisim Graph. (a_1) The teacher conducts an introductory lecture providing background knowledge on the phenomenon to be simulated. (a_2) Each student is invited to express a hypothesis. (a_3) All hypotheses are summarized by the teacher (a_4) Students with opposite hypotheses form a team and decide to test a hypothesis . (a_5) Team members negotiate the values of the parameters which are to be entered, each team member entering one parameter value on his phone (a_6) Team members discuss the results, each of them receiving a different representation of the same simulation results. The group repeats Phases 4 to 6 several times. (a_7) This debriefing session aims to synthesize the results of the simulation, by for instance, comparing all collected values on a graph and integrating them into a theoretical framework.

- Incorrect data interpretation: In a_7, each team member gets a different representation of results, which they have to integrate. Alternatively, they could get conflicting results, which is often the case in scientific experiments.

Is a design pattern a learning theory? No, because a learning theory predicts the learning outcomes and explains the learning process. A design pattern does not make prediction. In architecture, the "upper entrance" is a design pattern for houses located lower than the road or on steep slopes; it is common to many architectural designs, but it does not predict the quality of life of inhabitants. The graph design patterns do not predict learning outcomes, although they are accompanied with some hypotheses (such as SWISH) of the mechanisms by which they may trigger rich social interactions.

The patterns of operators I have presented reflect my personal approach to "scripting" collaborative learning (Dillenbourg & Hong, 2008). Many other design patterns exist that do not reflect this approach. Several authors have proposed multistep approaches for inquiry-based learning, that is, learning from conducting real or simulated experiments/studies. For Mulholland et al. (2012), the orchestration of inquiry learning is a graph with 8 steps called respectively, "*Find my topic*," "*Describe my question or hypothesis*," "*Plan my method*," "*Collect my data*," "*Analyze my data*," "*Decide my conclusions*," "*Share my findings*," and "*Reflect on my progress*." Muukkonen et al. (2002) propose different steps: "*Creating the context*," "*Setting up research questions*," "*Constructing working theories*," "*Critical evaluation*," "*Searching and deepening knowledge*," "*Generating subordinate questions*," and "*Constructing new working theories*." Even if these scenarios have not been described as patterns of operators, it is relatively easy to perceive the workflow idea behind these scenarios.

Another multistep scenario is project-based learning, in which the whole project is often decomposed into phases and milestones. Engeli (2000) developed a graph entitled "Phase X" with an interesting set of operators (see Figure 3.13). In educational projects, some teams find a great project that runs

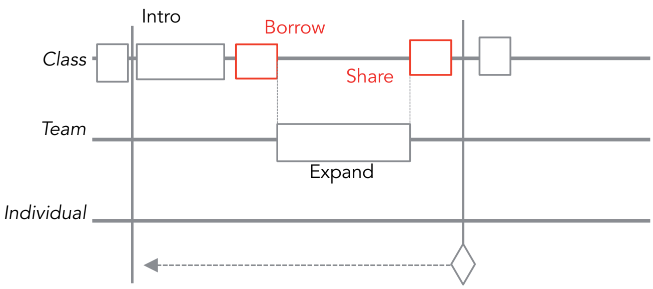


Figure 3.13 The Phase X graph can be used in many project-based MOOCs.

dynamically from the beginning to the end. Unfortunately, other teams are less lucky. They start with what sounds like a good idea, but realize only too late that it was not such a great one. They then have to continue with the same idea until the end of the project. To avoid this pitfall, the workflow in PhaseX offers students the possibility of jumping to a more promising project. Every week, at every stage of the project, teams produce an object, which they upload on the course website, where all objects appear (aggregation operator). For the next phase, every team can borrow any object uploaded by any team at the previous phase and continue to work with it. This creates a class with a continuous mix of ideas, which one could call plagiarism, but was actually viewed as an essential skill in the culture of that class—future architects. This graph may apply to many contents, but only if the objects produced at each phase can be browsed rapidly and reused easily. I have tried with more complex objects without much success.

Conclusions

At the end of this chapter, orchestration can be described as manipulating data structures. Each operator implements an orchestration act. The technical flavor of the term "operator" is probably acceptable when talking about massive online platforms; scale justifies an automating process. However, I claimed in the introduction that orchestration graphs also apply to physical classrooms. Would this reduce a teacher to a database manager? Aren't his emotions, enthusiasm, and personality more important than the operators described so far? Of course they are! My claim is different: the way an elementary school teacher orchestrates a lesson in a physical classroom can be modeled with graphs and operators. A teacher actually applies the operators many times during a lesson. He is certainly not aware of it, and should probably not be aware of it, but his actions can be modeled, that is, externally described, as a graph of operators—when he forms groups, when he collects completed sheets, when he distributes different tree leafs to different teams, when he asks those who have finished exercises to read their book, or when he uses Valérie's example a counter-example to the definition proposed by Odile.

Chapter 4

Stochastic processes

A sequence of edges and operators does not automatically make a good lesson. This chapter addresses the following question: do we expect that by going through such a set of activities, students will actually learn what they are supposed to learn? This question can be rephrased as: what is the probability that a graph produces the expected learning outcomes? Since we are not talking about one activity, but about a sequence of activities, this global probability is decomposed into a sequence of probabilities concerning the state of the learner after each activity. The learning path of a learner along the graph is modeled as a sequence of states that varies from "*perfectly fine*" to "*completely lost*," or even "*drop out*." The evolution of a variable over time has been studied across many domains: it can be the price of oil, the indices produced by the stock market, the temperature of the atmosphere, the electricity consumption in a city, or the frequency of heart beats. In our case, the key monitored variable is the state of the learner. These elements constitute time series, that is, they are ordered in time, for each learner. Some time series are deterministic; if you know the n^{th} value, you can determine the $n^{\text{th}+1}$ value; in mechanics, if you know the position, speed, direction, and acceleration of a car at time t, you can determine its position at time $t+1$. Some time series are non-deterministic; there is a part of indeterminacy in their evolution, a part of randomness. This is the case in education; humans are only partly predictable. Such a partly random process is referred to as a stochastic process, and its modeling relies on specific probabilistic methods, such as Markov chains.

From this point forward, the educational questions will be expressed in terms of probabilities, using a few mathematical notations. If you are not familiar with them, please bear with me; the few formulas that follow aim to bridge educational questions with computer science methods, which is the goal of this book.

Point 20 Learner states

The learning process of a student along the scenario is discretized as a sequence of learner states. Simply stated, the learner's state is a snapshot of what he knows, understands, misunderstands, and so on. In learning technologies, the terms "learner model," "student model," or "user model" have been used to refer to any information that the system or the teacher stores about the learner;[1] for example, how he carried out the previous activities, what he did not understand, is he motivated or anxious or ready to drop out, does he better understand graphical or textual representations, or did he post useful messages in the forum. This definition of the learner model includes two layers of information.

- The first layer contains the traces (Point 4) produced by the learner during activities, that is, his actual behavior as it has been objectively observed, such as what he said or where he clicked.
- The second layer contains interpretations of these traces in terms of global states such as "*He is facing difficulties*" or "*He is very motivated*." This diagnosis is inferred from the traces; it is an interpretation of the learner behavior, with a certain amount of uncertainty as explained in Point 30.

In other words, a learner model contains information that varies in terms of depth of interpretation—from straight observations to hypothetical diagnosis, and subsequently in terms of degree of uncertainty. I dichotomize this continuum by referring respectively to these two layers as the learner's behavior, $b_i(s)$, and the learner's state, $x_i(s)$.

- $B_i(S)$ is the set of behavioral elements produced by students and recorded by the system during a_i: the products (e.g., uploaded documents, answers to quizzes, forum postings), as well as the traces (such as navigation clicks, gaze data).
- $X_i(S)$ is often limited to cognitive elements, but there is no reason to exclude non-cognitive states, such as the emotional state, social relationships, and so on. In this point, I will only examine the state of an individual learner (π_1), but later on will examine how this applies to π_2 and π_3.

As uppercase are used for sets, $B_i(s)$ refers to the set of possible behaviors in a_i, that is, the set of possible values for $b_i(s)$. Similarly, the set of possible states for $x_i(s)$ is denoted $X_i(S)$. The sets of possible values (i.e., the language used for describing behaviors and states) is the same for any learner in S, hence $B_i(s)=B_i(S)$ and $X_i(s)=X_i(S)$. This set may vary across activities; for example, the behaviors produced when writing a text are not the same as when providing feedback on the text produced by a peer: $B_i(S)\neq B_j(S)$ and $X_i(S)\neq X_j(S)$. However, stochastic methods are easier to elaborate when the same set of state values are use for several activities (i.e., when $X_i(S)= X_j(S)$), which implies choosing a set of state labels that are activity-independent such as "*successful*" or "*slow*."

---

[1] I am not talking here about the model that a learner has about himself or that peers have about each other, but these would also be interesting points to investigate.

The behavioral trace left by learner s during activity a_i is denoted $b_i(s)$.

The set of possible behavioral traces left by learner s during activity a_i is denoted $B_i(s)$.

The set of possible behaviors at activity a_i is the same for all students $B_i(s)= B_i(S)$.

The state of learner s at the end of activity a_i is denoted $x_i(s)$.

The set of possible states at the end of activity a_i is denoted $Xi(s)$: $x_i(s) \in X_i(s)$.

The set of possible states at end of activity a_i is the same for all students $X_i(s)= X_i(S)$.

The value $x_i(s)$ can be numerical, such as the level of understanding (0.34), or a label selected among a finite set, such as {lost, motivated, fluent}. This value can be an n-tuple such as "*At the end of activity 3, Lena is highly motivated, demonstrated several misunderstandings, but is a leader in forum discussions*." This can be represented by a sequence of attribute-value pairs such as:

X_3(Lena)= ((motivation high) (understanding low) (forum leader))

The set of possible states is theoretically infinite. Empirical studies show that the effects of a specific learning activity may depend on many individual features: motivation, intelligence, prerequisites, cognitive style, and learning style. As a scientist, it is therefore tempting to elaborate sophisticated learner models. Instead, I recommend keeping the model simple, and easy to understand and process. To define the appropriate level of model complexity, John Self (1990) proposed a useful slogan: **don't diagnose what you can't treat**. For instance, if the next decision to be taken is to choose between two types of exercises, easy versus difficult ones, the diagnosis only needs to discriminate between two states, that is, students who need easy exercises and those who would benefit more from difficult ones. Therefore, in the following examples, I will often use the simple case where the state is defined by a single value, and this value belongs to a set of few possible states, such as:

$$x_i(s) \in X_i(S)= \{\text{fine, active, lost, drop}\}$$

State "*fine*": the learner is performing well
State "*active*": the learner is working, but does not seem to succeed well
State "*lost*": the learner does not understand at all or did not complete the activities
State "*drop*": the learned has dropped out (e.g., no log-in since N days)

The initial state of the learner at time zero (i.e., before any activity) is denoted by $x_0(s)$. It contains some general or background information about the students, that is, information that is not specific to an activity, such as the learner's name, gender, previous training, and age. It may also contain the state of the learner before any activity, at t_0, as measured by a pretest or a prerequisite test (Point 4).

The sequence of learner states articulates orchestration graphs with stochastic models (Figure 4.1). The states are measured at the end of each activity. Therefore, there is a one-to-one mapping between the graph of activities and the graph of states; as explained initially, a graph describes the operational aspects, the sequence of activities with its workflow, and the probabilistic network, modeled as transitions between states.

It could be desirable to model the changes of the states within an activity, especially if it lasts a certain amount of time. For instance, if the learner is lost in the middle of a project, it is better to find this out before the end of the project. To this end, the activity should be decomposed into subactivities, since the proposed modeling language considers the activity as its atom. The need for modeling intermediate states was cited in Point 4 as one reason to decompose an activity into subactivities.

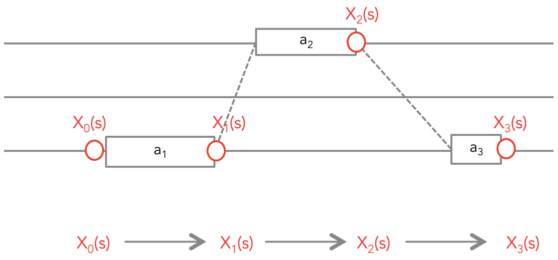


Figure 4.1 Two representations of a sequence of states—as an orchestration graph (top) or as a stochastic model (bottom).

Describing a pedagogical scenario as a stochastic model will enable predictions about the future states of learners. Experienced teachers actually to do this intuitively: "*This will be too difficult for Dan.*" Predicting states enables predictive adaptations; activities can not only be adapted on the basis of the current state of the learner, but also by predicting his future state (Chapter 6).

Several features of stochastic models have to be taken into consideration.

First, the stochastic models do not only capture the evolution of the state of one learner, but also the evolution of the state of all learners engaged in the same graph. This class constitutes a valuable source of information; of course, all learners are different, but the state of a learner is not completely independent of the state of another learner. For instance, if an explanation is unclear, it won't just affect a single learner but probably the majority of them. This will be developed in Point 32, with the notion of a modeling cube.

Second, some stochastic models rely on the so-called Markov assumption: that state $x_i(s)$ can be predicted by using only the information contained in the previous state, $x_{i-1}(s)$, independently from any previous state. Of course, this is not the case in a learning process; the state of a learner is influenced by most of the activities he did before. Learning is cumulative by nature. For instance, let's consider the following case:

x_1 (Lena) = "*lost*" x_1 (Louis) = "*fine*"
x_2 (Lena) = "*lost*" x_2 (Louis) = "*fine*"
x_3 (Lena) = "*lost*" x_3 (Louis) = "*lost*"

Even if Lena and Louis are in the same state at the end of a_3, they have a different history that is worth taking into account. One solution is to waive the Markov constraint by applying n-order or even variable-order Markov models, that is, methods that may take into consideration the last N states. The other solution, chosen here by simplicity, is to respect the Markov property and consider learner states to be cumulative: $X_i(s)$ contains all information necessary to predict the next state, that is, any information acquired about the learner across all previous activities. Therefore, I consider that $X_i(s)$ contains both the current state and the state history since t_0, the beginning of the scenario. The same evolution of Lena and Louis will hence be recorded by adding the last state to the beginning of the list of states:

x_1 (Lena) = (*lost*) x_1 (Louis) = (*fine*)
x_2 (Lena) = (*lost*, *lost*) x_2 (Louis) = (*fine*, *fine*)
x_3 (Lena) = (*lost*, *lost*, *lost*) x_3 (Louis) = (*lost*, *fine*, *fine*)

Third, if the sequence of $B_i(s)$ constitutes a Markov chain, the sequence of $X_i(s)$ corresponds to the so-called hidden states in Markov models. The term "hidden" means that it cannot directly be observed, but only inferred from student behaviors. This makes the prediction process two-fold. For instance, there are two reasons for predicting that Lena could be in the "*fine*" state: either because she gave the correct answer b_i(Lena) or because she was "*fine*" in the previous activities x_{i-1}(Lena). The method for combining these two sources of information is developed in Chapter 5.

Fourth, the stochastic models I will refer to require a discrete time; the evolution of the learner is modeled as a sequence of states, where events are ordered and labeled with integers. This is rather artificial since, naturally, time runs continuously during a pedagogical scenario, not just when we evaluate learner states. To translate a continuous time into a discrete sequence of states, we need to make time discrete (Figure 4.2). This is actually the case in orchestration graphs, since the activities are indexed sequentially (a_1, a_2,

a_3, ...) and the states are labeled accordingly ($X_1(s)$, $X_2(s)$, ...). I could therefore count any activity as a time unit, but this raises two problems. First, time is a critical predictor of learning gains; the time spent on a learning task is often positively correlated with learning outcomes.[2] Second, we have seen that the effect of a_i on a_{i+1} will probably be lower if there is a lag of 10 days, than if there is a lag of 10 minutes. How do we satisfy both the discretization requirements of stochastic models and the time sensitivity of human learning? There are two solutions illustrated in Figure 4.2.

- One solution is quantization; for instance, measuring $X^t(s)$ every second, even if a_i lasts for 120 minutes (Figure 4.2, left). This is often the case in stochastic models; for example, temperature is measured every second, blood pressure every hour, and sound frequency every millisecond. In many cases, the period or "the beat" is actually determined by the frequency of sensors that capture the value of the state.
- The second solution is to count every activity as a unit (Figure 4.2, right), that is, to consider each unit as having a duration of 1, but to include time information as a parameter of this activity. Each activity a_i has a duration d_i and the edge e_{ij} is associated with the lag between activities, denoted by l_{ij} (see Point 1).

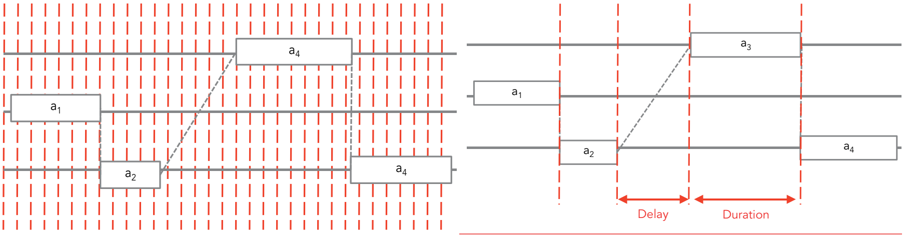


Figure 4.2 Left: Quantizing time. Right: Activity units with time as a parameter.

Let's consider the state of the learner while watching MOOC videos, each video being considered as an activity. In the approach illustrated on the right side of Figure 4.1, the behavior of the learner will, for instance, includes variables such as the play speed, the number of pauses, and the number of plays forward. In the approach illustrated on the left side of Figure 4.2, the number of pauses and other actions will be counted per slice of 5 seconds. The two

---

[2] Time-on-task often results in an experimental bias; the difference in learning gains between subjects who used method A versus method B is explained by the difference of time spent on the two methods instead of by the intrinsic differences between the cognitive processes triggered by these methods.

methods for discretizing time may reveal different phenomena. The choice of a method depends upon the purpose of analytics—namely, the research question. In the remainder of this book, I will follow only one method to avoid duplicating further developments: the activity is the time unit, 1 activity=1 period whatever the activity duration is, but the activity stores time information as a parameter.

Point 21 The states library

The examples of states that I have used so far come from common sense ("*fine*," "*lost*," etc.). There is nothing wrong with common sense, which encapsulates centuries of teaching experience, and some of the states proposed hereafter are "common sense" ones. However, this point proposes specific states inspired by educational research. For instance, "*lost*" sounds negative, but educational situations exist where errors[3] actually contribute to learning. The quality of any modeling depends upon the selection of states that are relevant for the ongoing learning process.

I review some states that have a specific educational interest, either because they create opportunities for learning, or because they reduce the chances for learning. Let me stress that the designer of a graph may choose whatever state ontology he considers relevant; this library only illustrates a diversity of states that other scholars have found relevant.

So far, I only mention individual states (π_1). The states at π_2 can be modeled in two ways. First, it is possible to build a model of the team as a whole. For instance, if a team is made of 3 students s_a, s_b, and s_c, the team model, denoted $X_\mathrm{i}\,(s_a,s_b,s_c)$, would be "*successful*" if the results of the teamwork was good, at a team product. Conversely, it could be interesting to model each individual state in the team. For instance, the state of the first learner s_a, denoted $x_\mathrm{i}\,(s_a)$, could be described as being "*free rider*," because he did not do a fair share of the worklook. Therefore, an activity at π_2 generates several models, one for the team as a whole and one for each individual. The same distinction applies to π_3 where it could be interesting to model the state of any individual attending a lecture or the state of class as a whole. What would class model $X_i\,(S)$ be, beyond the average of individual states? An experienced teacher has some appraisal of the global level of attention, maybe of understanding, of his class. If he uses a class response system, such as clickers in a classroom or voting systems in an online course, he may get a more accurate representation of the class state (e.g., what percentage answered correctly). We have been doing research in this direction, based on awareness devices (Alavi et al., 2009) and

---

[3] Different learners, depending on their personality and the context, may feel discouraged or challenged by different rates of failures.

on computer vision analysis of classroom video monitoring (Raca & Dillenbourg, 2013). One interesting aspect of MOOCs is that analytics potentially gives the teacher a more accurate representation of his class than in a standard lecture theatre.

Therefore, the state library is structured along 2 dimensions. The columns refer to the plane of the activity, that is, whether a_i is located at π_1, π_2, or π_3. The rows refer to the level of modeling, that is, whether X_i describes an individual, a team, or the whole class. In the list of states proposed below, a state labeled (π_2,π_3) is located in the column π_2 (plane of activity) and in row 3 (plane of modeling). A state can be multidimensional, that is, described by several of the elements listed in the library.

Table 4.1 The library of states in orchestration graphs.

		Plane of Activity		
		π_1	π_2	π_3
Plane of Modeling	**Individual Model (π_1)**	$X_i\,(s)$ Active / Passive On leave / Drop / Latecomer Disoriented Linear rigidity Impasse Trapped Over/Under generalization Deep/surface Gaming	$X_i\,(s_1)$ Social loafing Free rider / Sucker Individualistic Leader On/Off role	$X_i\,(s_1)$ With me Central Isolated Bridge
	Group Model (π_2)		$X_i\,(s_1, s_2, s_3, \ldots)$ Undersized/Oversized Cognitive/Emotional conflict Misunderstanding Group think Distributed	$X_i\,(s_1, s_2, s_3, \ldots)$ Cluster
	Class Model (π_3)			$X_i\,(S)$ Good/Bad spirit Slow Split

As for the two libraries previously presented in this book, edge labels and edge operators, this library is not presented as an exhaustive list or fundamental ontology. It illustrates the diversity of elements to be considered in stochastic models associated with orchestration graphs. Since these models manipulate probabilities in a semanticless process, the meaning of the results produced by these algorithms depend on the semantics encompassed in the data provided as input, that is, in the semantics behind the descriptors of the learner states.

Most of the states below can be treated as continuous variables; a learner can be more or less disoriented or teams members may have partial misunderstandings. I will nonetheless treat them as discrete variables, since this fits the stochastic models used in this book. It also fits intuitive forms of pedagogical reasoning, where simple labels are easier to manipulate than differential equations.

The states are described by principles such as "*The learner is in* $X_i(s)=K$ *if* $B_i(s)$ *includes these elements*." Actually, these principles should be expressed in a probabilistic way *"If* $B_i(s)$ *includes these elements, a reasonable hypothesis can be made that* $X_i(s)=K$.*"* Diagnosis is an inference, where a state is inferred from the behavior, and any inferences come with a certain amount of uncertainty (Point 30).

States (π_1,π_1) Active/Passive: These states can be inferred when $b_i(s)$ indicates different levels of global activity. These states are not inferred from a specific type of mistake, but from a global amount of activity. Being "*active*" sounds closer to $b_i(s)$ than to $x_i(s)$, but it implies some inference. For instance, despite many years of teaching, I still make frequent mistakes in detecting active students. Some of them are very talented in giving the impression of being active. In a MOOC, a related dilemma has been raised about participation rates: what is an active MOOC participant? Is a student considered to be active if he watches some or all videos, if he does some or all quizzes, if he uploads some or all assignments, or if he reads forums messages or contributes to the forum? Should all these behavioral indicators be combined? Many actors in the management of MOOC have invented their own metrics, and it would good to converge on some standards.

States (π_1,π_1) On leave/Drop/Latecomer: These states can be inferred when $b_i(s)$ indicates no activity for a certain duration. Like the previous states, they do not require much inference. The "*on leave*" state is important from the orchestration viewpoint; if a learner has not performed any activity for a certain defined period (e.g., being sick or on leave), the teacher has to anticipate how this learner will catch up when he comes back. Similarly, a MOOC may include some flexibility in the assignment deadlines in order to allow participants to catch up. In fact, completely rigid or completely open systems are easier to implement than semi-rigid ones. Managing missing students is a common event in school life that may sound trivial, but which actually requires orchestration operators, as illustrated in Chapter 6. The state "*drop*" also sounds trivial, but, since MOOCs do not include an "*I drop out*" button, the issue is predicting if, for instance, a MOOC participant is just a bit late or on the verge of dropping out. Finally, the state "*latecomer*" is also easy to infer, but hard to handle: how does the teacher orchestrate a scenario that lasts for

14 weeks, if, for instance, some students start the class two weeks after the main cohort? This occurs so often with MOOC participants, that MOOC platforms are evolving towards more flexibility.

State (π_1,π_1) Disoriented: This state can be inferred if $b_i(s)$ reveals navigation difficulties, for instance if $b_i(s)$ includes many selections of the "back" button in the navigation bar or the "cancel" option in a sequence of pop-up menus. These difficulties occur when learning activities rely on environments that require navigation in a multidimensional space, such as Wikipedia or a rich database. In the 1980s, when hypertexts entered education, empirical studies showed that such a nonlinear structure generated navigation difficulties (Rouet, 1992). Many students get "lost in hyperspace." Those who navigated efficiently were the students who previously had quite a good mental map of the contents—a global representation of the field to be explored. The subsequent design recommendation is to provide novices with such a map, displayed on the screen, to be used as a compass for navigation. This state is not specific to texts, as it may also occur in any multidimensional space, such as using a simulation environment with many parameters or a complex data repository. Actually, even in 3D learning environments, there is a risk that students waste a significant part of their time or energy in navigating rather than in learning. This explains why I am not convinced that embedding learning activities in something like Second Life or MineCraft brings anything to the learner other than the extraneous cognitive load, that is, a cognitive load that does not contribute to learning (Pass et al., 2003).

State (π_1,π_1) Linear rigidity: This state can be inferred if the sequence of problem-solving steps in $b_i(s)$ is significantly more linear than the expected heuristic process. Linear rigidity could, for instance, be detected by the absence of "undo" or "edit" actions in $b_i(s)$. This state is important when teaching problem-solving skills. In a seminal paper, Schoenfeld (1988) showed that university students who encounter difficulties in mathematical problem solving do actually follow a linear model of problem solving; starting from the initial state, they apply one operator, then another, up to the moment where they get stuck, that is, when no other step seems applicable. Conversely, good problem solvers follow a heuristic path; they try some operators, backtrack when there is a deadlock, and try another one. Schoenfeld noticed that this inefficient model of problem solving as a linear process is indeed reinforced by the behavior of those teachers who are usually considered "good teachers"; they smoothly present a proof or a demonstration in a linear fashion, which they know by heart, writing one line below another on a clean blackboard, without any errors. This state is more or less the opposite of the previous state "*disoriented*" but applies to a different type of activity, problem solving in this case and information search in the previous state.

State (π_1,π_1) Impasse: This state can be inferred if $b_i(s)$ includes an error to which the learner has been deliberately guided by the teacher or the designer. For instance, the graph illustrated in Figure 4.3 includes two impasses. In a_1, students learn the laws for predicting the distance of a ballistic shot, based only on gravity. In a_2, they have to predict the shoot distance with an accuracy of x meters on a distance of 30 meters, which they manage to do with the basic equation. In a_3, the shooting distance is set to 30 kilometers. The students will fail this activity because they need more parameters: $x_3(s)$ = "*impasse*." This designed deadlock, called "impasse" by some colleagues (Van Lehn, 1988) prepares the students for the next lecture (a_4) about friction forces. The same operation is then repeated with a second impasse in $X_3(s)$ in order to introduce the Coriolis force in a_7. This graph illustrates a design pattern as old as education itself, "*learning from mistakes*." Students do learn from some of their mistakes, but only in the right conditions; namely, if they are aware of the mistake they have made, and if they find the resources that will allow them to repair their misunderstanding in their environment. It is, for instance, important to keep the activity that leads to an impasse very short, otherwise the resulting state could be negative: $x_3(s)$ = "*unmotivated*."

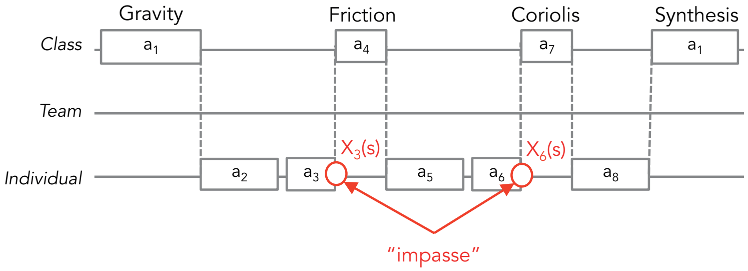


Figure 4.3 This orchestration graph uses the "*impasse*" state to motivate the introduction of new parameters in the ballistic equations.

State (π_1,π_1) Trapped: This state can be inferred if $b_i(s)$ reveals a misconception to which the learner has been deliberately guided by the teacher or the designer. It is a specific case of the previously described state, "*impasse*," in which the error comes from a so-called "misconception." A misconception is a belief that learners acquire from their daily life, which is consistent with their personal experience, but still scientifically incorrect. "*The sun rotates around my planet*" is a good example of a misconception. Another example is the belief that "*wool is warm,*" which matches many years of sensory experience, while "*wool insulates body heat from the air*" is the scientific concept. Educational research has shown (Vosniadou, 1994) that teaching a correct concept is not enough to eliminate the student's misconceptions. Misconceptions survive in parallel to the correct academic knowledge; they continue to be used in everyday life context, while the school-taught conception is used in the school context. To dismantle a misconception, it is necessary to "trap it," that is, to bring the learner to use it in a situation where it will produce an incorrect prediction. For instance, the teacher places two ice cubes on a table, one covered by a woolen blanket and one without anything on it. Then, he asks students "Which cube will melt faster?" The students who answer, "The cube with a blanket" are "*trapped*," which is good for learning, because it will allow the teacher to introduce the correct concept. In other words, being "*trapped*," despite its negative connotation, is actually a positive state. This illustrates another simple design pattern or orchestration graphs: (1) trap the misconception of the learner and (2) introduce the scientific concept.

States (π_1,π_1) Overgeneralization/Undergeneralization: These states can be inferred when $B_i(s)$ reveals that while learning a concept, the learner classifies as positive some instances that are negative—overgeneralization—or conversely, if the learner classifies as negative some instances that are positive—undergeneralization. As an example of undergeneralization, pupils tend to be confronted with rhombuses presented in their canonical forms (the longer diagonal being vertical) and do not recognize those presented, for instance, as a parallelogram (2 sides being horizontal). As an example of overgeneralization, a learner may believe that all French words that end with "-ment" ("longuement," "patiemment," "fréquemment,") are adverbs. These states are not negative, per se. If the teacher or the system detects them, he or it will be able to select the next positive or negative example to be presented, respectively for under- versus overgeneralization, as illustrated by Figure 4.4.

State (π_1,π_1) Deep/Surface: This state can be inferred if $b_i(s)$ reveals that learners who appear to be carrying out the activity correctly, actually achieve it without truly engaging with the material, or without trying to give meaning to it. The difference between these two states is often subtle. An experienced teacher may detect it, but it is more difficult for an algorithm. For instance, if the learner has to discover a scientific law with a scientific simulation, a learner in the "*surface*" state tries to succeed by successive approximations or

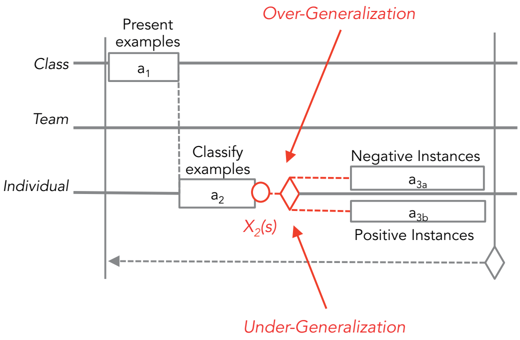


Figure 4.4 Teaching a concept by induction/discrimination.

uses random guesses. Instead, a deep learner will formulate hypotheses, test them, and refine them progressively. If a surface learner has to generate sentences in a grammatical exercise, he may reuse sentences he knows by heart, rather than constructing sentences based on the grammatical structures he is supposed to learn. In a mathematical activity, if a learner who produced the right answer receives feedback that explains why his answer is correct, the learner who spends time carefully reading the feedback is probably in the "*deep*" state. There is rich literature on the difference between learning styles, and "deep versus surface" is one of them (Beattie et al., 1997). A "learning style" is usually defined as a trait, that is, a stable feature of the learner, rather than a time-bound state. By using this as a state, I express the opposite; the same learner may behave in a surface or deep way within different contents, different contexts, or different phases of learning. The notion of deep/surface is still a controversial issue at the theoretical level, but, for some learning environments, it may be useful to identify surface approaches and send them specific feedback or modify the activity.

State (π_1,π_1) Gaming: This state can be inferred if $b_i(s)$ indicates that learners attempt to use the rules of the system in order to manipulate the system. All students have to manage time constraints in order to get the best score from a reasonable time investment, but gaming is one step further; it is about fulfilling the task without learning. Gaming is an extreme case of surface learning. For instance, if participation is counted as the number of postings in the forum, "*gaming*" characterizes the state of a student who posts meaningless messages to match the criterion "*number of postings*." If the system uses transactivity (Suthers et al., 2010) as criterion, that is, the extent to which a learner's reasoning and verbal interactions build upon the reasoning and verbal interactions made by other learners, a learner in the "*gaming*" state will often cut and paste from previous postings in order to fool the system. Detecting gaming is as difficult as detecting plagiarism, but sophisticated methods exist, and better ones are being developed. Whatever rules are integrated in a learning environment, humans are wonderfully talented in gaming the system. Conversely, creative designers may anticipate this and build activities so that gaming the system would nonetheless trigger rich cognitive processes.

State (π_1,π_2) Free rider: These states can be inferred if there is a disequilibrium in the distribution of actions among $b_i(s_1)$, $b_i(s_2)$, ..., $b_i(s_n)$, n being the size of teams at π_2. As teachers know, one team member often does not participate or contribute fairly to the team workload, letting the other team members do most of the work. A free rider is a team member who reduces his participation, assuming that the task will be mostly done by the others (Salomon and Gloverson, 1995). Conversely, the sucker effect occurs when a highly engaged team worker reduces his engagement in order to avoid doing most of the work himself. A simple way to detect them is to calculate the balance of participation within the team. This balance can for instance be the percentage of task actions performed by each individual inside a simulation environment or a concept map tool. It is also possible to count the percentage of utterances produced by each team member in a chat. These measures are simplistic and easy to game (previous state) by producing actions or utterances that do not contribute to the task or discussion. We nonetheless found that providing learners with feedback on their degree of participation helped the group to self-regulate (Jermann & Dillenbourg, 2008; Bachour et al., 2010).

State (π_1,π_2) Individualistic: Compared to the previous state, the opposite problem is when one student no longer cares about involving his peers, as he finds it more effective to do everything alone.

State (π_1,π_2) Leader: This state can be inferred if the actions in $b_i(s_1)$ influence more of the actions in $b_i(s_2)$, ..., $b_i(s_n)$ than vice-versa, n being the size of teams at π_2. The "*leader*" state is quite easy to detect by observing a team working around a table for a while, but it is rather difficult to detect in online interactions. A leader may produce few utterances, but these few utterances significantly influence the behavior of the rest of the team. This influence can be estimated in forums by the depth of the conversational threads. It is important to remember that diagnosis is a probabilistic reasoning (Point 30); the "*leader*" state can be detected as a hypothesis, that is, with a probability factor. The probability will be higher if the interface used for the activity a_i, that is, the tool that will record $b_i(s_1)$, has been designed purposely for measuring influence. Let us consider, for instance, an activity in which team members make propositions that have to be approved by the rest of the team, by clicking on a validate button. In this activity, $x_i(s_1)$="*leader*" if the propositions made by s_1 and stored in $b_i(s_1)$ are more often adopted by the team than the propositions by $b_i(s_2)$, ..., $b_i(s_n)$.

State (π_1,π_2) On role/Off role: These states can be inferred if $b_i(s)$ includes actions that do or do not correspond to the role assigned to s by the social operator and/or does not include the actions that correspond to this role. As explained in Point 8 and Point 16, some scenarios structure teamwork by defining roles that individuals have to play in their team. This applies to role-play activities (e.g., s_1 pretends to be an unhappy customer, while s_2 pretends to be the help desk person) as well as to collaboration scripts such as Jigsaws. If the learner is "*off role*," the scripted activity is not working properly, and hence the activity has to be adapted (see Chapter 6). Some graphs include a short activity for training learners to play their role before the main activity where they have to play it.

State (π_3,π_3) With me: This state can be inferred if $b_i(s)$ includes behavior that indicates a high level of attention to the class activity, in particular to the teacher who is lecturing. A teacher acquires a sense—far from perfect—of who is following him. This estimation is inferred from the learners' body language: who follows him visually, who establishes eye contact, who nods, who write notes, how timely they smile after a joke, and so on. When students watch video lectures, we can measure with eye trackers how closely their gaze follows the teacher's voice (if the teacher refers to X, does the student look at

X) and the teacher's deictic gestures on the slides. Even if the learner's attention is only a shallow indicator of learning, we actually found that "*with-me-ness*" was correlated to learning outcomes (Sharma et al., 2014).

States (π_1,π_3) Central/Isolated/Bridge and (π_1,π_3) Cluster: These states can be inferred by applying social network analysis to the traces $b_i(s_1)$, $b_i(s_2)$, …, $b_i(s_n)$, n being the size of the class S. Social network analysis (SNA) methods detect the social structure of a set of people in interactions (Borgatti et al., 2005). In a MOOC, it could be interesting to analyze who answers to whom in the forum, who annotates pictures posted by whom, and so forth. A learner is central if many links are directed to him (again, centrality is a continuous variable that is expressed here as discrete). He is "*isolated*" if he has no or very few links. SNA reverses the social operators described in Chapter 3; the latter attempts to engineer social structures that the former measures. These methods are not restricted to online education; when I was an elementary school teacher, I applied social network analysis to my class of 31 pupils by asking questions such as "*Who would you like to sit beside next week?*" This analysis revealed the existence of unconnected subgroups, called social circles or clusters (in the $\pi_2 \times \pi_3$ cell of the state library). Some of these clusters are called "cliques" if they are extremely tight, that is, if there is a link from each individual to each other member of the cluster. A learner is in the "*bridge*" state (in the $\pi_1 \times \pi_3$ cell of the state library) when he is the single link or one of a few links between two clusters.

The following states do not describe the individuals within a team, but the team as a unit.

State (π_2,π_2) Oversized/Undersized: These states can be inferred when $b_i(s_1, s_2, s_3, \ldots)$ reveals that the size of the team $(s_1, s_2, s_3, \ldots)$ is lower or higher than the size determined by the group formation operator (Point 16). Like the "*drop*" state, this state is easy to detect, but important for orchestration; the size of the class may not be a multiple of the target group size—some teams members may leave the class, others may join late. For some group activities, there is no problem if a member is missing or if there is an extraneous member; if a_i is a brainstorming task, for instance, and the social operator created teams of 8, a_i will still work with 6 or 9 students. In smaller groups though, a missing member may increase the workload for the other members. When the teamwork is scripted, for instance, with specific roles, the scenario may simply not work anymore. Point 16 presented an operator to cope with this issue. It is therefore important to detect this state in order to trigger this operator.

State (π_2,π_2) Cognitive conflict/Emotional conflict: These states can be inferred when $b_i(s_1, s_2, s_3, \ldots)$ reveals that there are clear divergences between the team members—whether these divergences concern cognitive/epistemic elements (difference of ideas, viewpoints, theories, …) or concern interpersonal items and become emotional. As explained in Chapter 2, the emergence of cognitive conflict among team members may be a state desired by the designer, since it often leads to verbalization of tacit knowledge and possibly de-centration of individual viewpoints. Detecting this state is therefore important for applying the SWISH pattern presented in Point 19. However, the conflict only contributes to learning gains if it remains at the cognitive level, but is not a personal conflict. A cognitive conflict is easier to infer if the interface has been purposely designed to support this inference: for instance, if s_1 votes for a solution and s_2 votes for the opposite one, if s_1 changes the value of the simulation parameters set by s_2, if s_1 deletes an object created by s_2, and so on. The emotional conflict can be inferred from the use of emoticons, or when dialogue utterances include references to a peer's personal characteristics (skills, intelligence, age, race, or gender), including insults, or when the same message is posted 20 times in a chat.

State (π_2,π_2) Misunderstanding: This state can be inferred when b_i (s_1, s_2, s_3, ...) reveals misunderstandings between team members. This is a softer case of conflict; peers may not have clearly opposite opinions, but nonetheless misunderstand each other. For instance, many people agree with the statement: "*The rich should pay more taxes than the poor,*" as long as there is some ambiguity around what is meant by "*rich.*" In the previous chapter, I stressed that—up to a certain threshold— misunderstanding is important for collaborative learning, as it requires verbal elaboration. A graph designer may induce misunderstanding; for instance, by providing peers with ambiguous data or, in a 3D environment, with different viewpoints. Misunderstanding can be inferred from dialogue among peers by the frequency of so-called "repairs," that is, utterances by which peers try to re-establish broken common grounds (Clark, 1996). Repairs include rephrasing (e.g., "*What I meant is 'richer than me'*"), requests for clarification (e.g., "*What do you mean by rich?*"), with or without deictic gestures (e.g., "*This one?*"). In face-to-face situations, misunderstanding can be emphasized by facial expressions such as eye frowns, while in online chats they often lead to messages such as "*?????*". However, misunderstandings are only useful if peers become aware of their misunderstandings. When collaborating online, one property of the interface is to allow participants to see what the other does. In a simulation, for instance, if Dinesh says "*Increase it,*" while referring to the temperature parameter, and Sara answers "*OK,*" but increases the pressure, Dinesh will detect the misunderstanding, if he sees Sara's action on the shared simulation interface.

State (π_2,π_2) Group think: This state can be inferred when b_i (s_1, s_2, s_3, ...) reveals that team members agree upon a solution without having sufficiently considered the other solutions. This state is the opposite of conflict; it corresponds to a special case of surface learning at π_2—sometimes the social pressure to reach consensus (Whyte, 1952) or the teacher's pressure, leads the team to select the first solution that results in a consensus (even if it is not the optimal solution), without exploring alternatives. This state is negative, because the team neglects solutions that could have been better. When this state is detected, it should trigger specific reactions from the teacher or the system, such as pointing out ideas that have been mentioned by some members, but that the team neglected in order to reach faster consensus. This state can be inferred if the activity interfaces keep traces of all the solutions that have been explored by the team.

State (π_2,π_2) Distributed: This state can be inferred when b_i (s_1, s_2, s_3, ...) reveals a clear division of labor among team members. Even when there is no predefined role for an activity, roles often spontaneously appears during teamwork. Some emergent division of labor is frequently observed (Miyake, 1986); for instance, some peers tend to focus on low-level aspects. When two learners use the same computer, this is often the learner who has his hands on the keyboard and mouse. Naturally, the second learner focuses more on regulation aspects, such as the strategy in problem solving or the text structure in collaborative writing. Blaye (1988) showed that this mutual regulation alternates with time (*A* regulates *B* then *B* regulates *A*) and is progressively internalized as a self-regulation skill. In larger teams, various task distributions may emerge, depending on the nature of the task and on the interface. The way functions are displayed on the screen may, for instance, induce a task distribution. This state can be inferred by building a matrix storing how many times each team member (row) performs each action/subtask (column). The matrix of a distributed team will have one or a few dominant values per row, as on the left side of Figure 4.14. As for many states, this could be a continuous variable; the degree of distribution varies from none to total (i.e., no overlap between the tasks performed by *A* and those performed by *B*), which is something that could be measured by the method used for computing the entropy of a transition matrix (Point 23).

	Temperature	Pressure	Volume
Lena	3	**14**	2
Louis	**13**	2	4
Manu	3	1	**12**
Olga	**9**	1	3

	Temperature	Pressure	Volume
Lena	7	6	6
Louis	6	8	4
Manu	4	4	8
Olga	5	4	4

Figure 4.14 The matrix indicates who does what in the simulation; the team on the left is in the state "*distributed*," i.e., with a clear division of labor.

Finally, the state can describe the class *S*, as a whole, that is, at π_3. This global state can be an average or an aggregation of individual states. The state "*with me*" that has been described for an individual-in-a-class (cell π_1,π_3 of the state library) is also relevant for the class—a teacher would feel that the class is "with me" if most of the students in that class are "with me." The exact ratio that corresponds to "*most of the students*" is unknown; it probably varies with teachers. This "average state" can be extended to other individual states such as "*active*" versus "*passive*," as well as "*deep*" versus "*surface*."

State (π_3,π_3) Split/Homogeneous: A "*split*" state can be inferred when b_i (S) reveals a bimodal distribution; a "*homogenous*" state is the opposite one. Some teachers perceive a dichotomy among their students: "*Those who get it and those who don't*." In an introductory programming class, the notion of a variable often introduces a "split" in the audience. In addition to intuition, the teacher who runs the graph in Figure 4.5 can use clickers to capture the state of this class. If the state is "*split*," he will stop the lecture and propose exercises; otherwise, he'll continue the lecture. This state requires specific forms of adaptation as described in Point 37.

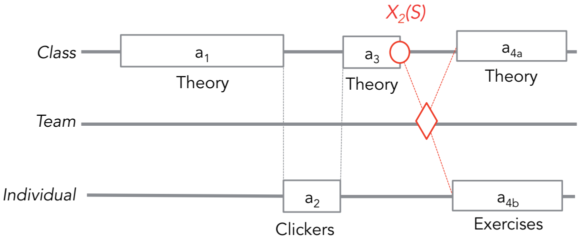


Figure 4.5 Teacher's decision if the class state is "*split*."

Point 22 States transitions

Orchestration graphs are deterministic; unless there is an accident, activity a_2 will follow[4] a_1 with a probability of 1. The probabilistic nature of the graphs concerns the state of the learners, $x_i(s)$. An activity is expected to bring the learner from one state to another; a learning activity changes the state of knowledge, a motivation activity is expected to increase the state of motivation, a prerequisite activity modifies the state of knowledge, and so on. The term "expected to" means there is some probability that this will occur, but no certainty. The transition from a state $X_i(s)$ to the next state $X_{i+1}(s)$ is what I model by a stochastic model.

---

[4] If the graph includes branching operators, choosing between a_2 and a_3 is algorithmic; it is not random.

Let's consider that 100 students are in a "*lost*" state at the end of a_i ($x_i(s)$="*lost*"). Among them, 39 remain in the same state after a_j, 24 switch to "*active*," 10 to "*fine*," and 27 to "*drop*." Let's also consider that these statistics are available for each $x_i(s)$. This set of data can be represented as a transition graph (Figure 4.6), which is also used for describing Markov models. Please note that the states have to be mutually exclusive—a learner belongs to only one state at a given time. Therefore, the probabilities of edges leaving from the same state sum to 1. The state "*drop*" is an absorbent, that is, there is no arrow that brings the learner back from "*lost*" to any other state. Figure 4.6 is a statistical summary of how students evolve at the end of after a_1 to the end of a_2.[5]

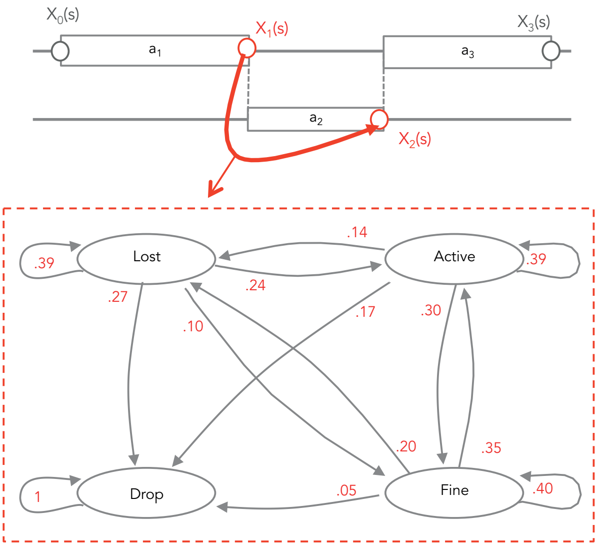


Figure 4.6 Transition probabilities between learner different values for $X_1(s)$ and $X_2(s)$.

A matrix can represent the same information as the graph in Figure 4.6. The transition matrix between the set of states $X_i(S)$ and the set of states $X_j(S)$, is called $M^{ij}(S)$. It is a $p \times q$ matrix in which p is the number of possible values in $X_i(S)$ and q is the number of possible values in $X_j(S)$. If the transition is from activity a_i to a_j and $j>i$, the values of $X_i(S)$ define the rows of $M^{ij}(S)$, and the values of $X_j(S)$ define the columns of $M^{ij}(S)$. The cell (k,l) is the probability

---

[5] The point here is to capture transition probabilities, not to choose an activity based on these probabilities. Such a decision, which can be implemented as a Markov decision graph, is discussed in Chapter 6.

of being in the l^{th} state of $X_j(S)$ after having been in the k^{th} state of $X_j(S)$, for $0 < k < p$ and $0 < l < q$. Since the states are mutually exclusive, the sum of probabilities per row is 1.

Table 4.2 A matrix of transitions from state $X_1(s)$, vertically, to $X_2(s)$, horizontally: 39% of students who were in a "*lost*" state remained in the same state, while 24% moved to "*active*," and so on.

$M^{ij}(S)$	$X_2(s)$				
$X_1(s)$	Lost	Active	Fine	Drop	Total
Lost	39%	24%	10%	27%	100%
Active	14%	39%	30%	17%	100%
Fine	20%	35%	40%	5%	100%
Drop	0%	0%	0%	100%	100%

In pedagogical scenarios, the transition matrix between $X_i(s)$ and $X_{i+1}(s)$ is rarely the same as the matrix between $X_{i+1}(s)$ and $X_{i+2}(s)$. This is different from stochastic models of stationary processes, in wich the same probability distribution does not change over time (the same matrix applies to all state transitions).

Where do the matrix data come from? One could hope to find the probability values in scientific literature, but empirical research only sparingly covers the data a graph designer would need. Therefore, the most promising approach (central to learning analytics) is that these probabilities are acquired empirically. This empirical construction of the transition matrix makes especial sense for MOOCs, since they collect data from large sets of participants. These two approaches, theoretical and empirical, are of course not exclusive—Bayesian methods allow integrating a probability a priori (the theoretical probability), with a probability as posteriori (empirically collected).

Point 23 Matrix entropy

Initially, I defined the weight (ω_{ij}) of an edge (e_{ij}) as a probability value that captures "*How much the student performance in* a_i *will determine his performance in* a_j." This temporary definition holds if the states after a_i and a_i are both numerical values. It would then be possible to calculate a simple correlation. However, I propose modeling the learner's state as a discrete set of values and representing the relationship between two states by a transition matrix. To compute ω_{ij}, the probabilities stored in M_{ij} have to be aggregated into a single parameter, which would represents the weight of the edge. I address this question in two points, proposing two values—the entropy of a transition matrix is presented below, and the utopy[6] of a transition matrix is explained in the next point.

To calculate these values, I have to make the assumption that the same set of states is used both in a_i and in a_j, which implies that M_{ij} is a square matrix. Actually, the condition is not simply that two connected activities have the same number of states, but also that they have the same set of states, that is, that $X_i(S)=X_j(S)$. In this case, the diagonal of the matrix means "staying in the same state."

The first measure I propose for summarizing the state transition matrix is entropy. For readers who are not familiar with this topic, the term *entropy* comes from information theory;[7] Claude Shannon proposed measuring the uncertainty of a so-called "random variable". Of course, the state of the learner is not random, but it varies over time in a way that is far from being deterministic. It is precisely this part of randomness that entropy tries to measure. A typical example is tossing a fair coin; whether it falls on one side or the other is completely random, and the entropy is maximal. Conversely, if someone is cheating with an unfair coin that always falls on the same side, the entropy will be null. In our context, the fair coin situation translates as follows: if a learner who is in a "*success*" state in a_i has a 50% chance of being in the same state in a_j and a 50% chance of moving to a "*failed*" state, the uncertainty of the transition matrix M^{ij} is maximal. Conversely, the unfair coins translates as follows: if a learner who is in a "*success*" state in a_i has a 100% chance of being in the same state in a_j, the uncertainty of the transition matrix M^{ij} is null. If he has a 100% chance of being in a "*failed*" state, the entropy will also be null, which indicates that entropy is only one aspect of the transition. If we don't use a coin, but a dice with 6 sides, the possible outcomes of throwing the dice will be a vector with 6 values, corresponding to the probability of each side. A fair dice would be associated with the vector [1/6 1/6 1/6 1/6 1/6 1/6] while an unfair dice, more predictable, could for instance be associated with [1/2 1/10 1/10 1/10 1/10 1/10]. In other words, if the vector of probabilities has a uniform distribution, entropy is maximal. The entropy hence measures the distance from the uniform distribution. To make a long story short,[8] the entropy, denoted by H, is computed with Shannon's formula:

$$H(X) = -\sum_i P(x_i) log_2 P(x_i)$$

where x_i is one of the possible values of the random variable X.

---

[6] I know that the word "utopy" does not exist in English, but I like to invent the words I need.

[7] The term has a related, but different, meaning in physics.

[8] Plenty of online material exists that explains the meaning of this formula and why 2 is a convenient base for the logarithm.

This formula can be translated into the learner modeling context. The random variable X is $X_i(s)$, that is, the set of possible states of the learner s in individual diagnosis. The values of this random variable (i.e., the x_i in the Shannon equation) are the different possible states A = {"*fine*", "*lost*", ...}. The entropy regarding the state of a learner s is therefore:

$$H\left(X_i\left(s\right)\right) = -\sum_{a \in A} P\left(X_i\left(s\right)=a\right) \cdot log_2 P\left(X_i\left(s\right)=a\right)$$

I will now explain in 4 steps how I propose to estimate the uncertainty of a transition matrix. Mathematicians would certainly propose better ways to compute this matrix entropy, but, in the meantime, I have developed an intuitive approach.

(Step 1) Compute the entropy of each row of M^{ij}

The formula above computes the entropy for the transition between one state at the end of a_i and the next possible states at the end of a_j. This corresponds to one row in the transition matrix, that is, to a vector of probability values. To consider all states in a_i, it is necessary to compute the entropy of the whole transition matrix, that is, for each $X_i(S)$ in M^{ij}, H ($X_j(S)$) has to be calculated and these values aggregated into a global measure $H(M^{ij})$.

Let's consider the transition matrix M1 below, which is very close to an identity matrix; for any state in a_i, it can be predicted almost with certainty that the student will remain in the same state in a_j. For M2, the prediction is also safe; for any state in a_i, it can be predicted almost with certainty that the student will end up in a "*fine*" state in a_j. At the opposite end of the scale, the predictability is extremely low in M3 where, whatever the state was in a_i, the state in a_j could be anything. As explained earlier, this predictability is estimated by computing the entropy of each row, as one vector. In M1 and M2, the

M1	Lost	Active	Fine	***H***
Lost	0.98	0.01	0.01	0.16
Active	0.01	0.98	0.01	0.16
Fine	0.01	0.01	0.98	0.16

M2	Lost	Active	Fine	***H***
Lost	0.01	0.01	0.98	0.16
Active	0.01	0.01	0.98	0.16
Fine	0.01	0.01	0.98	0.16

M3	Lost	Active	Fine	***H***
Lost	0.34	0.33	0.33	1.58
Active	0.34	0.33	0.33	1.58
Fine	0.34	0.33	0.33	1.58

M4	Lost	Active	Fine	***H***
Lost	0.5	0.3	0.2	1.49
Active	0.1	0.4	0.5	1.36
Fine	0.1	0.1	0.8	0.92

Figure 4.7 Entropy values (*H*) for various transition matrices (numbers are fictitious; they have not been computed empirically).

entropy of each row (column *H* in Figure 4.7) is close to 0, which corresponds to what I called above "safe prediction." Conversely, in M3, every state of $X_j(s)$ is almost equally probable, and hence entropy is maximal for each row. The matrix M4 falls between these two extreme cases.

$$\text{(Step 1)} \qquad H\left(M^{aj}\right) = -\Sigma_{b=1 \text{ to } q} m^{ij}_{ab} \, log_2\left(m^{ij}_{ab}\right)$$

where M^{aj} is the a^{th} row of M^{ij}, $1 \leq a \leq p$ and M^{ij} is $p \, X \, q$

(Step 2) Normalize the entropy of each row of M^{ij}

Let's now consider M5 (Figure 4.15). The uncertainty in M5 is similar to M3 (all states $X_j(s)$ being equally probable), but *H* is higher because the number of states is higher, and because entropy is computed as a sum of values for each state. Since the number of states may vary between graphs, the comparison of entropy between different transition matrices may be biased. It would hence be difficult to get the 'global picture', an estimate of entropy variation along the graph. Therefore it I propose to normalize the entropy, i.e., to make it independent from the number of states. This is achieved by dividing it by the maximal entropy, $log_2(p)$, where *p* is the number of states. The normalized entropy per row (H_0) hence varies between 0 and 1.

M5	Lost	Active	Fine	Great	***H***	***Ho***
Lost	0.25	0.25	0.25	0.25	2.00	1.00
Active	0.25	0.25	0.25	0.25	2.00	1.00
Fine	0.25	0.25	0.25	0.25	2.00	1.00
Great	0.25	0.25	0.25	0.25	2.00	1.00
					***H*(M5)**	**1.00**

Figure 4.15 Comparing M5 to M3 reveals that entropy varies with the number of states (numbers are fictitious; they have not been computed empirically), hence the introduction of Ho, normalized entropy.

$$\text{(Step 2)} \qquad \boldsymbol{H0}\left(M^{aj}\right) = \frac{\boldsymbol{H}\left(M^{aj}\right)}{log_2\left(q\right)}$$

where M^{aj} is the a^{th} row of M^{ij}, $1 \leq a \leq p$ and M^{ij} *is $p \, X \, q$*

$$\Rightarrow \qquad \boldsymbol{H0}\left(M^{aj}\right) = \frac{-\sum_{b=1}^{q} m^{ij}_{ab} \, log_2\left(m^{ij}_{ab}\right)}{log_2\left(q\right)}$$

By normalizing this value, we loose information regarding to the number of states, which is important in information theory. In our case however, the magnitude of state variations is more or less constant: it ranges from 'completely lost' or 'perfectly fine'. The discretisation of this continuum into several states is a design decision, based on the diagnosis possibilities (Point 30). The loss of information regarding to the number of states hence seems acceptable.

(Step 3) Compute the average entropy for all rows in M^{ij}

The matrix entropy could simply be the mean value of the row entropy values. However this average does also constitute a loss of information. Let's consider matrix M6. The entropy is low for learners in state 'good', most of them remain in the same state, while there is maximum entropy for those who were in state bad. The average of the two entropy values is 0.43. If it occurred that actually 90% of students were in state 'good' (column 'prior distribution') and only 10% in state 'bad', the actual uncertainty for a random student is actually lower than as computed so far: the low entropy of the row 'good' should influence proportionally more the matrix entropy than the high entropy of the row 'bad'. We therefore compute H'(M6) as the weigthed average of each row entropy, by multiplying HO by the prior probability. The matrix entropy is hence 0.11 instead of 0.43, which is a better account of the reality.

Prior distribution	M6	Bad	Good	H	Ho	H′
10%	Bad	0.5	0.5	1	1	0.1
90%	Good	0.9	0.1	0.14	0.14	0.12
					0.43	0.22

Figure 4.16 Integrating prior distribution for weigthing the row entropy.

$$\text{(Step 3)} \qquad \boldsymbol{H}'\left(M^{ij}\right)=\sum\nolimits_{a=1}^{p} P_a \boldsymbol{H0}\left(m^{aj}\right)$$

M^{ij} is $p\ X\ q$ where P_a is the prior probability of row a

(Step 4) The weight of M^{ij} is the opposite of its entropy

Now, the goal was to estimate the strength of an edge, that is, how strongly the student state at the end of a_j is dependent upon his state at the end of a_i. This dependency has been called the weight of an edge in Point 5. The weight of e_{ij}, is denoted ω_{ij}. The notion of strength or weight is actually the opposite[9] to the notion of entropy; the weight should be zero when entropy is maximal and 1

---

[9] Alternatively, the weight could be inversely proportional to the uncertainty. However, $\omega_{ij} = 1/H'(M^{ij})$ would not be a value between 0 and 1 anymore.

if entropy is null. Therefore, I define the weight of the edge e_{ij}, as 1 minus the average normalized entropy of its transition matrix M_{ij}. This notion will be further refined in the next step.

$$\text{(Step 4)} \qquad \boldsymbol{\omega}_{ij} = \mathbf{1} - \boldsymbol{H}'\left(M^{ij}\right)$$

$$\Rightarrow \qquad \boldsymbol{\omega}_{ij} = 1 + \frac{\sum_{a=1}^{p} \sum_{b=1}^{q} m_{ab}^{ij} \, log_2\left(m_{ab}^{ij}\right)}{p \cdot log_2\left(q\right)}$$

Point 24 Matrix utopy

As I mentioned, a transition matrix that describes a situation where all students succeed has the same entropy as a matrix where all students fail. In Figure 4.17, the two matrices have the same weight (0.45), but M7 describes an activity in which 75% of learners end up in a positive state, while M8 describes the opposite situation—75% of learners end up in the "*lost*" state. The fact that M7 and M8 lead to the same $H'(M^{ij})$ value is acceptable from a pure probabilistic viewpoint, but here their purpose is to model education processes, where the goal is that students reach the learning objectives. Therefore, it is necessary to introduce a second variable, which I refer to as the utopy of a matrix.

M7	Lost	Active	Fine	*H*	*Ho*
Lost	0.01	0.24	0.75	0.87	0.55
Active	0.01	0.24	0.75	0.87	0.55
Fine	0.01	0.24	0.75	0.87	0.55
				ω(M5)	0.45

M8	Lost	Active	Fine	*H*	*Ho*
Lost	0.75	0.24	0.01	0.87	0.55
Active	0.75	0.24	0.01	0.87	0.55
Fine	0.75	0.24	0.01	0.87	0.55
				ω(M6)	0.45

Figure 4.17 These two matrices have the same entropy, but different utopy levels (numbers are fictitious; they have not been computed empirically).

Utopy is a variable that will estimate the degree of utopia behind a transition matrix. The term is perhaps too strong—I could have called it call "optimism" or the "progress index," but even if we are talking about mathematical indices, there are always values behind educational data. Please allow me an emotional detour.

> *The endemic confusion between selection and training is a tumor in educational systems. Let us imagine a teacher in the first year at elementary school, when children learn to read. If 100% of the children, whatever their entry level is, end their school year with great reading skills, everyone would agree that this teacher is excellent. This situation constitutes an example of utopia, but the reality is less beautiful; a teacher rarely manages to compensate for all individual differences. Now, let's consider a 1st*

> *year teacher at university level. If he has a success rate of 100%, he would not be praised, as would the elementary school teacher. Instead, he would be suspected of having lowered the level of expectations. The problem I want to point out is the confusion between selection and training. Selection is actually acceptable in a first year university class; it's better to fail at entrance than 3 years later. But selection should not replace teaching excellence. Teachers should bring university students as high as possible and, only then, filter out those who have no chance of succeeding the following year. The index "matrix utopia" turns this philosophy into numbers.*

I will now explain in 4 steps how to calculate the utopy of a transition matrix.

(Step 1) Count learners who improve and those whose state degrades

To calculate the utopy of a matrix, let's assume that the set of states is ordered from the least to the most desirable, the first row and column being the least desirable state. This is often the case in education, where various states indicate increasing levels of mastery. The improvers are located in the triangle submatrix M^+, above the diagonal and colored green in M9. Conversely, the cells located below the diagonal and colored red in M9 concern students who lowered their state and constitute M^-. The difference between M^- and M^+ tells us something about the progress of the whole class. To calculate this index, we have to add each element in M^+ and M^-.

(Step 2) Take into account how much they improve or degrade

A learner who moves from state-3 to state-5 improves more than a learner who moves from state-3 to state-4. Therefore the sum of M^+ and M^- cell values is weighted with the distance from the cell to the diagonal; each cell m_{kl} is multiplied by $(k\text{-}l)$ which is the distance from the diagonal. Using the distance to the diagonal implies that the scale of states is linear, i.e. that the difference of the states between two neighbor rows (or columns) is always the same. In this case, a cell located at a distance of two from the diagonal is twice better or twice worse than a cell located next to the diagonal and using this distance as 'penalty' makes sense. If the scale is non linear, one could use the square of the distance or any other way to produce non-linear penalties.

(Step 3) Calculate the difference

The notion of utopy refers to the fact that an activity leads to more improvement than degradation of the learners' state. Hence, utopy is the difference between the weighted sum calculated respectively for M^+ and M^-.

(Step 4) Normalize the difference

The difference calculated in the previous step is not independent from the size of the matrix. In order to normalize the result from –1 to +1, we have to make it relative to the matrix size ($p \times p$). The maximal distance from a cell to the diagonal is $p-1$ for the first row, $p-2$ for the second row, and so on. We hence divide the weighted difference between M^+ and M^- by the sum of $p-1$, $p-2$, $p-3$, ... until $p-p=0$. This recursive sum can be replaced in the formula by $p.(p-1)/2$.

$$\gamma(M) = \frac{2}{m(m-1)} \sum_{k=1}^{m-1} \sum_{l=k+1}^{m} (l-k)m_{kl} - \sum_{k=2}^{m} \sum_{l=1}^{k-1} (k-l)m_{kl}$$

$$= \frac{2}{m(m-1)} \sum_{k=1}^{m} \sum_{l=1}^{m} (l-k)m_{kl}$$

A matrix that would be totally stable (M10 in Figure 4.18) leads a utopy index of 0. At the opposite end of the scale, the dream situation captured in M11 has a utopy index of 1, since all learners end up in the highest state. Finally, the dramatic M12 has a utopy of –1 since all learners end up in the worst state. M13 and M14 respectively are optimistic and pessimistic matrices between these two extremes.

M9					
	0.2	0.2	0.2	0.2	0.2
	0.2	0.2	0.2	0.2	0.2
	0.2	0.2	0.2	0.2	0.2
	0.2	0.2	0.2	0.2	0.2
	0.2	0.2	0.2	0.2	0.2
			U(M)		**0**

M12					
	1	0	0	0	0
	1	0	0	0	0
	1	0	0	0	0
	1	0	0	0	0
	1	0	0	0	0
			U(M)		**–1**

M10					
	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1
			U(M)		**0**

M13					
	0.2	0.2	0.2	0.2	0.2
	0.1	0.1	0.2	0.3	0.3
	0	0	0.2	0.3	0.5
	0	0.1	0.2	0.2	0.4
	0	0	0	0.2	0.8
			U(M)		**0.47**

M11					
	0	0	0	0	1
	0	0	0	0	1
	0	0	0	0	1
	0	0	0	0	1
	0	0	0	0	1
			U(M)		**1**

M14					
	0.5	0.1	0.2	0.1	0.1
	0.2	0.2	0.2	0.2	0.2
	0.7	0.2	0.1	0	0
	0.2	0.2	0.2	0.2	0.2
	0.8	0.2	0	0	0
			U(M)		**–0.42**

Figure 4.18 Calculating the utopy of a transition matrix (numbers are fictitious; they have not been computed empirically).

M14 illustrates a situation where a_1 actually inhibits learning in a_2, for instance. This may sound strange, but it can occur. For instance, Schwartz &

Bransford (1998) showed that initially providing learners with a definition of the concept to be learned prevents them from applying deep induction in later examples.

Some states presented in the state library (Point 21), such as "*trapped*," "*impasse*," or "*misunderstanding*," constitute conditions for learning, despite their negative label. They have to be placed in the M^{ij} in such way that if many learners move to that state, the matrix utopy index will be positive.

This utopy index relies on the assumption that matrices are square. One could probably live with this assumption if the activities along the graph were similar. Another solution (if various sets of states existed across activities), is to reduce the number of states to a few levels common to all activities by merging states, resampling them with quartiles or with z-scores, etc.

Another assumption in the way I compute utopy is that the set of states constitutes a metric scale, since an improvement of two steps in the scale of states is two times more desirable than the improvement of a single step. This assumption is implicit in the way we multiply each cell value by its distance from the diagonal. If the scale cannot be considered as metric (but is still ordinal), this distance factor can be removed, and the index will still give a balance between students who improved and those who lowered their state.

The last point is to integrate entropy and utopy into a single value to serve as ω_{ij}. When the matrix utopy is negative, it's better to have higher matrix entropy; it is better to be unsure of failing than to be sure of failing. Inversely, if the matrix utopy is positive, with less entropy, the better it is. Therefore, I simply propose to estimate the **weight of an edge as the product of its non-entropy index and its utopy index.**

$$\omega_{ij} = (1 - H'(M^{ij}))\ U(M^{ij})$$

Since $H'(M^{ij})$ varies between 0 and +1 and since $U(M^{ij})$ varies between −1 and +1, the value of ω_{ij} will be between −1 and +1:

- If $\omega_{ij} = +1$, a_i is very critical for a_j;
- If $\omega_{ij} = 0$, a_i is more or less useless for a_j and can be skipped if needed; and
- If $\omega_{ij} = -1$, a_i is very detrimental to a_j and should be removed or replaced when improving this pedagogical scenario.

These values, entropy and utopy, are proposed here as a bridge from orchestration graphs to stochastic models. If our community considered them useful, I am convinced that colleagues with strong mathematical backgrounds would refine these values. Their purpose is to provide a quantitative summary of a transition matrix.

Alternatively, the summary can be produced by simple visualizations of the raw transition probabilities. Let's consider the transition matrices M13 and M14, presented in the previous pages. We will replace each cell in the matrix by a small square at the same row and column in the visualization. The probability in the cell determines the color of the small square: white if the cell value is zero, black if it is 1, with grey levels in between. The positive utopy of M13 can be intuitively perceived through the right side unbalance of the picture, while M14 negative utopy is visible through the left side unbalance.

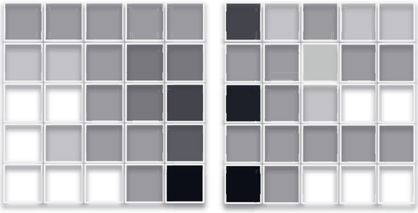


Figure 4.8 A visualization of transition matrices M13 (left) and M14 (right).

The meaning of this simple visualization depends, of course, upon the nature and the ordering of states; the unbalance around the diagonal only refers to utopy if the states have been ordered from the least desirable to the most desirable. Any other matching between the states and the cells would require a specific interpretation. The point is to provide a global visualization of the graph, as in Figure 4.9, where it is immediately visible that the third activity created a problem.

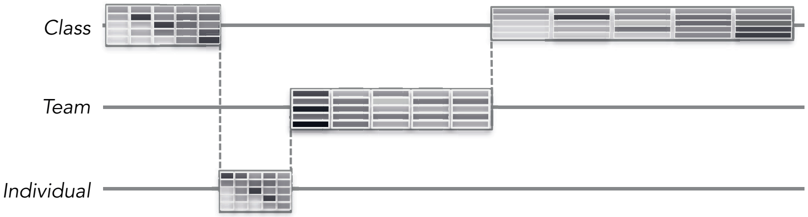


Figure 4.9 A visualization of transitions in an orchestration graph.

Point 25 Edge elasticity

The effect of one activity on the next activity is not time-proof; effects tend to fade out with time. Hence, we have to model how ω_{ij} decays when l_{ij}, that is, the time lag between a_i and a_j (Point 1) increases. Let's consider an edge with a motivation label. The motivation asset raised in an activity often has a short life span, especially for extrinsic motivation tricks. The same is true for an advance organizer label: the pre-activated structure will not remain activated indefinitely. It is slightly different for prerequisite labels; it is expected that the skills acquired in an activity will not vanish immediately, but after a few weeks, it will probably be necessary to refresh these skills. In experimental pedagogy, this decay is considered an indicator of the quality of a pedagogical scenario. Researchers often include a delayed post-test, for instance, 4 weeks after the lesson, to verify how much acquired skills have resisted the ravages of time. If participants obtain high scores at the immediate post-test, but low scores at the delayed post-test (i.e., they learn, but then they forget), the pedagogical scenario cannot really be considered effective.

How far can an edge be stretched without losing its conditional power, that is, its strength? The *elasticity* of an edge is modeled as a *time decay* function that quantifies how the weight of the edge between two activities varies as the lag between them increases. Even if the effect of the method does not fade out, the uncertainty about its effect will probably increase, and hence the weight will decrease.

Let's consider that a scenario has been conducted several times and illustrated in Figure 4.10. In 2013,[10] the lag was 60 minutes (l_{ij}^{2013}=60) and the weight was 0.74 (ω_{ij}^{2013}). The MOOC will be repeated every year until 2016,

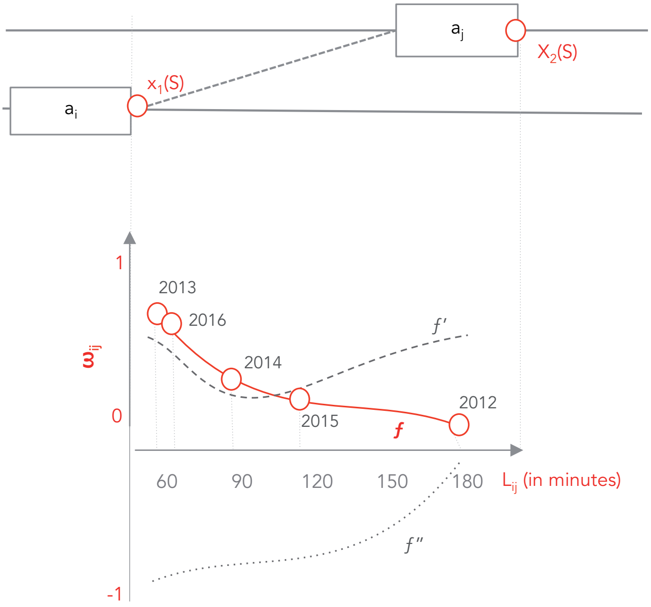


Figure 4.10 Edge elasticity is modeled as a strength decay function.

---

[10] I use an annual course frequency in order to give a concrete example.

with variations of l_{ij}. This set of data (l_{ij}^{year}, ω_{ij}^{year}) can be approximated by the decay function f. What is the shape of this decay function? Is it linear? Which slope does it have? I am afraid that the answer will be specific to the nature of the edges, that is, the edge label. Figure 4.10 presents another function, f', which reveals a first phase shaped by the so-called recency effect (Stainer & Rain, 1989), followed by what could be called a phase of digestion in lay terms. If ω_{ij} is negative, the decay function illustrated by f'' in Figure 4.10 will tend towards zero, and hence ω_{ij} would increase as the lag increases. One example could be in sport education, when students have to physically recover from a_i before starting a_j; ω_{ij} would be negative if l_{ij} is short, but it would increase if the lag increased and enabled recovery.

Defining decay function in this way is somehow theoretical. In practice, many sessions of the same MOOC would be necessary before having enough points (l_{ij}, ω_{ij}) to estimate f. In addition, the size of S in every MOOC would have to be large enough to make sure that differences of weight are not due to demographic variations. Even if the sample size is large enough, the data would be influenced by many other external factors. Education is a complex field. In the absence of sufficient data for estimating a function, one could use a binary function similar to the food expiration date: “Do not use a_j more than n days after a_i.”

In summary, the edge weight decay function can be modeled as follows. Since the scenario can be run at different time intervals, I don’t refer to years, but to the session number: 1, 2, …

ω_{ij}^1 is the weight of e_{ij} computed from a first run of the scenario

l_{ij}^1 is the lag a_i and a_j during a first run of the scenario

l_{ij}^2 is the lag a_i and a_j for the second run of the scenario

ω_{ij}^2 is the weight of e_{ij} to be predicted from a second run of the scenario

$\omega_{ij}^2 = f(\omega_{ij}^1, l_{ij}^1, l_{ij}^2)$ and f is the decay function illustrated in Figure 4.10

This edge elasticity illustrates the concept I have of orchestration graphs as a kind of organic species. A pedagogical scenario is not carved in stone. It has to be adapted on a regular basis to match the evolution of the learner states and the evolution of the context (see Chapter 6). The weight and elasticity of edges influence the flexibility of an orchestration graph, that is, how easily the timing or the ordering of activities can be modified without breaking down the expected effects.

Point 26 Operators and matrices

So far, transitions matrices relate two activities located at π_1, while graphs stretch over multiple planes. Let's first consider what $X_i(s_{1,2})$ is at π_2, that is, modeling a team of learners. If two activities a_i and a_j are both located at π_2, then the state $X_i(s_{1,2})$ can be related to the state $X_j(s_{1,2})$ within the same matrix as for individual states, M^{ij}. Examples of states for π_2 and π_3 have been proposed in the state library (Point 21).

The transition matrix would become more complex if a_i was located at π_1 and a_j at π_2. Can we calculate the probability of $p(x_j(s_1,s_2)=$"*failed*") knowing that $x_i(s_1)=$"*failed*" and $x_i(s_2)=$"*success*"? This would produce a matrix as M15 in Table 4.3; it illustrates the case where the probability of a team succeeding a_j would be higher if each learner individually succeeded a_i than if they both individually failed a_i. However, sometimes a pair of students who failed individually, but who engaged in rich verbal interactions may nonetheless succeed, a mystery that researchers have investigated for many years. In other words, the social mechanisms that predict $X_j(s_1,s_2)$ from $X_j(s_1)$ and $X_j(s_2)$ are more complex than the product of two probabilities. As we know that collaborative learning effects are very task specific (Dillenbourg, 2002), a transition matrix probably has to be elaborated empirically for any edge of a graph.

If the pair (s_1,s_2) has been formed in a specific way; for instance, with a high and a low achiever or with a learner who got a certain subset of data with a learner who got a different subset, then for each pair we can determine who is s_1 and who is s_2, as in M15. In this case, row 2 (pass-fail) and row 3 (fail-pass) of M15 are different. M15 is not square, but can be split into 2 square matrices, one for s_1 and one for s_2. Now, if there is no difference in roles while selecting teams members, learners will be randomly assigned to s_1 and to s_2. Hence row 2 (pass-fail) and row 3 (fail-pass) of M15 are equivalent and merged in M16. In this case, the matrix can still be made square by duplicating the second row.

Table 4.3 Transition Matrix when moving from π_1 to π_2.

M15		$X_j(s_1,s_2)$	
$X_i(s_1)$	$X_i(s_2)$	Pass	Fail
Pass	Pass	0.8	0.2
Pass	Fail	0.6	0.4
Fail	Pass	0.7	0.3
Fail	Fail	0.3	0.7

M16		$X_j(s_a,s_b)$	
$X_i(s_a)$	$X_i(s_b)$	Pass	Fail
Pass	Pass	0.8	0.2
Pass	Fail	0.6	0.4
Fail	Fail	0.3	0.7
$s_a=s_1$ and $s_b=s_2$ or $s_a=s_2$ and $s_b=s_1$			

To calculate such a transition matrix, the system must know the composition of teams. This information is actually available, since an operator associated to the edge e_{ij} has been used for forming the group. For instance, the team formation operator (Point 15) selected s_1 and s_2 based on criteria for minimizing or maximizing distances among them. In this case, the information contained in the operator helps build the matrix. Conversely, the team formation operator could be expressed as searching for **the partition of S into pairs $s_{1,2}$ that maximizes the probability that $X_j(s_{1,2})$= "*Pass*."** As mentioned, $x_j(s_{1,2})$ can hardly be predicted by an universal team formation algorithm. However, if a transition matrix has been built from the previous sessions of the same graph, $M^{ij\text{-}Past}$, the team formation could apply the following algorithm:

1. Select two students from S, s_1 and s_2
2. Find s_1'& s_2'$\in S^{Past}$, a former class such as $x_i(s_1') \approx x_i(s_1)$ and as $x_i(s_2') \approx x_i(s_2)$ (find two former students who were in a similar state as two current ones)
3. Retrieve $x_j(s_1', s_2')$ from $M^{ij\text{-}Past}$ (retrieve the state as a team once these former students have been paired)
4. Store $x_j(s_1', s_2')$ as the predicted value in $M^{ij\text{-}predicted}$, a matrix that stores all predicted team states
5. Select two other students from S and repeat from step 2, for all students in S;
6. Compute $U(M^{ij\text{-}predicted})$;
7. Repeat from step 1 with all possible partitions of S into pairs
8. Choose the partition that obtains the highest $U(M^{ij\text{-}predicted})$

Computing the matrix utopy of all possible partitions of S into pairs implies very intense computation. This algorithm should be optimized.

M15 and M16 illustrated a transition from an activity a_i at π_1 to an activity a_j at π_2. What about the other way around, from π_2 to π_1? In many cases, the team state $X_i(s_{1,2})$ is not sufficient to predict the next individual states $X_j(s_1)$ and $X_j(s_2)$; for instance, if s_1 and s_2 were successful as a team in a_i, it does not imply that they will encounter individual successes in a_j. The individual differences that existed before teamwork, in $a_{i\text{-}1}$, will probably not have completely disappeared after team activities. A more accurate estimation may require integrating information from $X_{i-1}(s_1)$, the individual state of the learner two activities before. Actually, to satisfy the Markov assumption, we decided that the current state includes all previous states, which implies that $X_{i-1}(s_1)$ is available inside $X_i(s_{1,2})$. If this is not the case, the transition could be modeled between these 3 activities as in M17 (Table 4.4). In M17, the third row contains the probability that a student (s_1) succeeds a_j if his team succeeded a_i, and if he individually failed a_{i-1}.

Table 4.4 A transition matrix when moving from π2 to π1.

M17		$X_j(s_1)$	
$X_i(s_1)$	$X_i(s_{1,2})$	Pass	Fail
Pass	Pass	0.9	0.1
Pass	Fail	0.7	0.3
Fail	Pass	0.4	0.6
Fail	Fail	0.2	0.8

The transition matrices M15 to M17 connect activities located at different planes. I call these *cross-plane transition matrices*. Both cross-plane matrices and social operators connect data across planes, the former in an analytics mode, and the latter in an operational mode. This relationship between cross-plane matrices and social operators illustrates the deep intertwining between two viewpoints on orchestration graphs that I have developed throughout this book, namely the graph as a workflow and the graph as a probability network.

Other operators can also affect the state transition matrix. For instance, a decision operator could split a matrix. Let's imagine that the operator uses the following rule:

IF $x_i(s)$= "good" THEN select a_j for s OTHERWISE select a_k for s.

This gives two transition matrices, M_{ij} and M_{ik}, which illustrate the value of adaptive instruction; by choosing a_j for learners who succeed a_i and choosing a_k for learners who failed a_i, the utopy of the matrix increases as it can be hypothesized that the two split matrices would have a higher utopy index than a matrix where all students do the same activity.

Table 4.5 A transition matrix in adaptive instruction; should the next activity be a_j or a_k?

M_{ik}	$X_k(s)$		M_{ij}	$X_j(s)$	
$X_i(s)$	Pass	Fail	$X_i(s)$	Pass	Fail
Fail	0.8	0.2	Pass	0.9	0.1

Point 27 Theory plug-ins

A theory plug-in is a fragment of theory that could—in a perfect world—provide the designer with the initial cell values for the transition matrices of the graph to be designed. These plug-ins could come from any learning theory; for example, behaviorism, constructivism, situated cognition, or mastery learning. A learning theory comes with 3 interwoven fibers:

- A learning theory **predicts** that a learner who performs an activity or a sequence of activities will acquire some skills. If the skill to be acquired is denoted by c_i (Point 6), we can express the probability that c_i is acquired after participating in a_i: $p(c_i \mid a_i) \geq \alpha$. The value of α should be reasonably high—close to 1—if c_i="*landing a paraglider*," since this skill has to be mastered by the practitioner, and probably lower for c_i="*speaking French*," as it is possible for a practitioner to live without it.
- A learning theory **explains** why some activities trigger cognitive processes (e.g., induction, compilation) that produce new knowledge. This

explanation is part of our scientific knowledge. It is also important because it determines the space of generalization, and the variations around the activity that would probably generate similar learning results because they would probably trigger similar cognitive processes.

- A learning theory often comes with a legacy of educational **practices** and with a philosophical flavor—a certain vision of humankind. There is nothing wrong in associating values with an educational theory, but the drawback is that value-loaded theories become schools of thought. This book tries to avoid this pitfall.

From the beginning of the book, I have argued for "integrated learning." I stressed the interest of building graphs of activities that do not belong to a single theory. This is why I refer to theory plug-ins instead of theories. I borrow the metaphor proposed by Wenger (2010); he used the terms "plug and play" to refer to theory fragments that can be combined, as multiples lenses through which scientists analyze the world.

A cognitive process does not occur in a vacuum, but transforms the competency at a_i into competency at a_{i+1}. Therefore, I should write $p(c_{i+1} \mid a_{i+1} \wedge c_i) \geq \alpha$. Some learning theories are indeed defined by the sequencing of activities more than by the activities themselves. For behaviorism, a_{i+1} should introduce a single piece of novelty with respect to a_i. For mastery learning, c_{i+1} should not be started if the learner does not master its prerequisites c_i.

Unfortunately, these probability values are rarely available in literature for 3 reasons:

- The probability values computed in empirical research are generally measured in a relative way; method A is found to be significantly more effective than method B with $p < 0.05$. Transition matrices contain absolute terms; if the learner was in state $x_i(s)$ before using method A, the probability that he ends up in state $x_s(s)$ would be, for instance, 0.34.
- There is no established referential that describes either pedagogical methods or cognitive states in an operational way. This referential would be necessary for extracting findings from the available literature that closely match the designed activities. This book is a small step in that direction, but it will take more of this effort to build such a referential and turn educational research into a truly cumulative science.
- Empirical results are very sensitive to the context in which they have been collected; every detail matters. There is a low probability that the results observed in one or several experiments will generalize across all contexts where the same scenario is to be used. This explains why pedagogical reforms often fail to scale properly.

Do these restrictions in fact make this 27th point meaningless? The values found in literature will not be accurate (especially given the extreme simplication that follows), but could at least provide some probability estimates as initial values for the transition matrix. They would also support a Bayesian approach. These values would be inaccurate, but they would reduce the matrix entropy compared to a matrix with no prediction at all. Then, the values would need to be refined with time, as empirical data are collected for specific graphs. The examples of plug-ins presented below aim to demonstrate how general principles behind learning theories could be turned into predictions.

Association: If the learner frequently associates items *x* with *y*, such as "*nitrate*" and "NO_3^-" during a_i, this increases the probability that when presented with *x* during a_j, the learner will be able to cite *y*.

Reinforcement: This is a special case of association. If learner behavior $b_i(s)$ is triggered by stimulus *x* and then followed systematically and immediately by a positive feedback during a_i, the probability increases that if stimulus *x* is presented during a_j, the learner will produce behavior $b_i(s)$.

Compilation: If the learner applies procedural skill *c* many times during a_i, and if a_i and a_j are very similar to each other, the learner will probably apply *c* faster and with a lower cognitive load during a_j.

Chunking: If the learner applies c_1 and c_2 sequentially during a_i, the combined skill c_{1+2} will generate a lower cognitive load during a_j than the sum of the cognitive load triggered by c_1 and c_2.

Reflection: If, during a_i, the learner hesitates between possible answers that differ with respect to element *x*, an immediate feedback during a_i will inhibit the elicitation of *x* during a_j.

Argumentation: If two learners argue about *x* during a_i and if *y* is an element used in the argument $y \rightarrow x$, the probability increases that these learners may apply $y \rightarrow x$ in a_j.

Explanation: If, during a_i, a learner elaborates a new explanation with a chain of elements $[x \rightarrow y \rightarrow z]$, and a_i and a_j are very similar to each other, then the probability increases that the learner will be able to use $x \rightarrow y$ or $y \rightarrow z$ while performing a_j.

Induction: If, during a_i, a learner compares positive {e+} and negative {e-} instances of a concept *K* and if {f} is the set of features that are common to {e+} and simultaneously absent from {e-}, then the probability increases that the learner will include {f} in the definition of *K* after a_j.

Mutual regulation: If a student is able to regulate the problem-solving process of his teammate during a_i, and if a_i and a_j require similar problem-solving strategies, the probability increases that he will be able to regulate his own problem-solving process during a_j.

Internalization: If, during a_i, a student s_1 participates in meaningful dialogue with a more advanced student s_2 within the zone of proximal development of s_1, the probability increases that s_1 will replay this dialogue during individual reasoning for a_j, that is, as a monologue.

Conclusions

While the previous chapters described orchestration graphs in a rather static way, this chapter has led us to model the dynamics, that is, the transitions between activities, or the way a learner state evolves over time. Modeling complex processes as sequences of discrete states has become a pervasive approach in modern sciences and fits quite well with the way educational processes have been modeled with orchestration graphs. It may sound like overkill to introduce concepts such as entropy, but they fit intuitively well with the uncertainty that a teacher can experience while facing a classroom.

Chapter 5

Learning analytics

Learning analytics refers to the process of collecting and analyzing the traces left by learners during their activities. Recent learning analytics rely on data mining and machine-learning methods, which can produce reliable results when provided with massive sets of data. Since MOOCs produce massive data—the participants' traces—they have triggered a new boost in learning analytics research. The goal of this chapter is to articulate analytics in regards to orchestration graphs.

The role of analytics is two-fold. First, it aims to improve the effectiveness of pedagogical scenarios in order to determine the most relevant adaptation of learning activities to the needs of the students or to propose modifications for the next occurrence of the pedagogical scenario. Second, the role of analytics is to produce new knowledge about education, as a research method. The first goal is engineering; the second is science. How analytics fulfill these roles is described in the next chapter.

Learning analytics is often conducted as a post hoc activity, that is, for the analysis of what happened in a pedagogical scenario, after it has been completed. One of the core ideas in this book is that learning analytics should instead be anticipated from the design phase. If the analytics process is designed after the completion of the course, it's difficult to extract meaningful lessons. I am not simply talking about anticipating which interactions will be stored in which formats in the log files. Anticipation includes the formal description of the whole sequence of activities, their social plane, their input/output, the relationship between them (the label of edges), the way data are transferred (edge operators), the way students are modeled (the set of states), and so on. This idea is one of the reasons for formalizing educational scenarios, in addition to scaling up. It is captured by the previously mentioned slogan "design for analytics."

Learning analytics is not restricted to MOOCs; it applies to any educational activity, including classroom activities (as is the case for all elements proposed in this book). A teacher who is reflecting about the data he has collected through his senses is carrying out learning analytics, even if he is not using any technology. Furthermore, learning analytics have been applied to data captured in a classroom. For instance, Raca et al. (2013) placed cameras in a classroom and used computer-vision to estimate the average level of attention. Alavi et al. (2009) gave "lanterns" (see Point 41) to students in a classroom. These devices indicate to teaching assistants which learners are working on which exercises, for how long, and whether they need help. At the end, lanterns collect statistics such as the average working time and the time that students spend waiting for the teaching assistant. The range of low cost sensors that can be used in a classroom grows every day: light EEGs and mobile eye trackers that are almost as lightweight as glasses, skin conductivity bracelets that approximate stress measures, heart frequency devices, pressure sensors inside chairs to capture the learners' body postures or movements... This multiplication of data collection tools opens a new space for researchers, even though the ethical issues are as huge as the educational affordances are high. Actually, collecting everything that can be collected and then crunching massive data sets is not the best research strategy. Analytics requires careful thinking, as emphasized by the phrase "design for analytics."

Point 28 Cognitive diagnosis

The cognitive state of a learner is not directly accessible. It is not possible to open a student's brain and count its "knowledge." The state has to be inferred from the learner's behaviors. Cognitive diagnosis is the process of inferring the learner state from his behavior during the same activity. The adjective "cognitive" was introduced at the beginning of this field of research, probably for differentiating this inference process from the medical diagnosis. Actually, both forms of diagnosis are instances of abductive reasoning: inferring causes (respectively health problems versus misunderstandings or knowledge gaps) from their consequences (respectively symptoms versus behavior). Several causes may lead to the same consequences; several diseases may produce similar symptoms and several cognitive states may lead to the same learner's answer. The art of docimology—the science of educational measurement—is to design activities (test items) in such a way that each cognitive state to be detected will produce a distinct behavior. This is addressed in Point 30.

The set $B_i(s)$ does not only include learners' products (e.g., solutions, answers) but also the smaller traces of their activities that I call **behavioral dust**. Here are some examples of collected dust:

- The learner's response time, which can be easily recorded, is sometimes more predictive of the learner's state than the answer itself. Conversely, silence or non-response over a long time is also interesting information; for instance, for anticipating dropout.
- The learner's mouse path before selecting a response could indicate, for example, how self-confident the learner is in choosing this answer or which other answers he considered. This parameter is probably correlated with response time.

- The learner's gaze patterns can be recorded by eye-tracking devices. Soon, the camera embedded in any laptop will be accurate enough for eye tracking. We have found that gaze patterns accurately predict knowledge levels (Liu et al., 2009) for an individual learner, and that they predicted programming expertise in a programming task (Jermann & Nüssli, 2011).

Dust is produced at each social plane of the graph, as illustrated in Table 5.1.

Table 5.1 Type of behavioral traces across planes.

Behavior/Planes	π_1	π_2	π_3
Products	Individual responses	Joint solution to a problem	Number of daily forum postings
Dust	Mouse path, response time, gaze paths	Eye contact between learners, rate of acknowledgement	Gaze directions, leg movements, whisperings in a lecture theater

The diagnosis methods may not be the same for processing products and dust, but both feed into the diagnostic process. Therefore, in the following points, I will not discriminate between them any further. I wanted to stress the affordances of behavioral dust, revealed by the field "social signal processing" (Vinciarelli et al., 2009); the input for diagnosis is much richer than a set of correct or incorrect answers. In addition, this massive set of data enables the probabilistic approach that emerged in the previous chapter—elaborating transition matrices requires frequent and systematic data collection points. They probably wouldn't work if they only considered a few products generated now and then during the scenario.

Point 29 Behavioral abstractions

In computer-based education, the set of possible behaviors for $B_i(s)$ is defined by the interface, that is, by the ways in which users interact with the computer: mouse clicks, text entries, gestures or speech, pictures uploaded, and so on. This set of inputs is finite and small in multiple-choices questions and infinite in open questions such as writing an essay or sketching a new logo. In some cases, the diagnostic input is not a single behavior, but a sequence of behaviors. This is the case for behavioral dust such as mouse paths or gaze paths (previous point), but also for sequences of game actions or quiz responses. The input of the diagnostic processes is then a vector such as $[A\ D\ B\ A\ A\ B\ D]$ where A, B, and D are responses. Interpreting a sequence of answers, rather than a single one, is especially relevant in problem-solving activities. When several paths to a solution are possible, providing feedback on the solution path (i.e., the problem-solving strategy), may be more relevant than providing feedback on each problem-solving step.

The diagnostic process consists in mapping the set of behaviors to the set of cognitive states. In principle, the set of cognitive states is infinite, but, as mentioned earlier, in practice it is much smaller; if the next decision is to choose between two types of exercises, easy or difficult ones, the diagnosis only needs to discriminate between two states for learners—those who need easy ones and those who may benefit from difficult ones. How does the mapping occur? When behavioral information is a complex data set, diagnosis is sometimes conducted in several steps or layers; some features are extracted from the raw behavioral data in order to provide abstractions. I denote them $B'_i(s)$. These features may themselves be abstracted into higher-level features, $B''_i(s)$; for instance as successive steps of inference. Examples of features are given for each plane in Table 5.2. The term “feature” is used in machine learning, but this word could be ambiguous outside this community. Therefore I prefer the term “behavioral abstraction”: extracting from raw data patterns of elements that are more likely to be predictive of the final outcome.

For instance, eye trackers provide the raw position of gaze every 200 milliseconds as raw data. The number of back-and-forth transitions between two pictures that learners have to compare is an instance of behavioral abstraction that speaks more than the raw milliseconds. In our study of online collaborative problem solving (π_2), we used a higher-order feature; gaze recurrence is the percentage of time that one peer looks at the same object at the same time than his teammate. This behavioral abstraction was predictive of misunderstanding (an interesting π_2 state) and even team performance (Jermann et al., 2011). Of course, two people do not look at exactly the same pixel and not at the same millisecond (the time distance between two gazes is up to 2 seconds). The feature “gaze recurrence” is hence a complex computation from the raw behavioral data. It is important that our community develops a library of features that have potential for predicting states. In contrast to the libraries that I have previously presented (edges, operators, and states), this library is quite underdeveloped.

In other words, diagnosis may require multiple steps of data processing, as implemented, for instance, in multilayer neural networks. The art of learning analytics (and machine learning) is to find out which features allow an algorithm to map the input to the output. However, in the next steps, I will consider the diagnostic process as a one-step process, with behavioral data as the input and a hypothetical cognitive state as the output.

Table 5.2 Library of abstractions from raw behavioral data.

Plane	Examples of behavioral abstractions
π_1	• Rate of back/cancel actions in a navigation task. • Redundancy: Did the learner ask a question for which he already had an answer? • "With-me-ness": Did the learner look at the object mentioned by the lecturer in the video? • Attention map: Which areas does the learner look at most often?
π_2	• Balance of participation: Did all team members do a fair share of the workload? • Task-distribution: Do team members perform specific subsets of the tasks? (Even if there is a 50/50 balance, it may be that John only does specific tasks and Lena others.) • Rate of acknowledgement: What percentage of utterances from a learner received acknowledgement—from a simple nod to an acknowledging action (*A* says "*This is a mistake*" and *B* deletes "*This*"), or to a full reformulation of the initial utterances? We found this simple rate to be a good predictor of the quality of collaboration (Dillenbourg & Traum, 2006). • Transactivity: Did team members build utterances upon the utterances produced by their peers? • Cross-recurrence: Did team members look at the same object at (more or less) the same time? • Rate of redundancy: Did the learner ask a question for which another team member already had the answer?
π_3	• Conversation depth: The average depth of conversation threads in forums. • Connectivity: What is the minimal number of students that need to be removed from the social network to disconnect the other nodes from each other (Diestel, 2005)? • Homophily: Do students form ties with similar versus dissimilar students (McPherson et al., 2012)? Ties can be forums postings; similarity is measured through students' profiles. • Reciprocity: If student *A* often replies to another student *B* in the forum, is the opposite true? • Propinquity: The tendency for actors to have more ties with those who are geographically close (Kadushin, 2012). • Density: The proportion of direct interactions between two students relative to the total number of possible interactions between all students (Xu et al., 2010).

Point 30 Diagnosis entropy

Since the real cognitive state of a learner is unknown, the diagnostic process elaborates a hypothesis about $X_i(s)$ based on $B_i(s)$. Hypotheses have various levels of uncertainty. If a medical doctor sees a child whose face is covered with red dots, the hypothesis of measles is reasonable. While a massive mosquito attack is less probable, it is still not impossible. To determine the certainty of a diagnostic hypothesis, I use the entropy measures that were introduced in the previous chapter.

Let's consider an introductory course on statistics. We decide that $X_i(s)$ has 2 possible values: (m) misunderstanding or (g) good understanding. Let's imagine that activity 5 is a quiz with 4 possible answers, illustrated in Figure 5.1.

In order to reduce the variance of the set [1 2 2 3 3 3 4 5 8], 3 numbers can be removed. Which ones?
a) Remove all occurrences of number 3
b) Remove the numbers that appear several times
c) Remove 1, 5, and 8
d) Remove 4, 5, and 8

Figure 5.1 Example of a question.

A reasonable hypothesis is that answer c) indicates a state of good understanding. However, this hypothesis is not robust, as there is a 25% chance that the learner selected c) randomly. The probability that the learner has understood the concept of variance given that he replied c) should be lower than 1. In addition, let's imagine that the teacher knows from experience that only 20% of students understand the concept of variance after this activity ($P(X_5(s)=good)=0.2$). In this case, the probability that the learner is in the 'good' state should be much lower, whatever answer he gave. The Bayes theorem allows us to integrate this prior information and the current behavioral information.

Let's assume that the probability that a student who understands well the concept of variance chooses answer C is 1: $P\,(B_5(s)= answer\text{-}c \mid X_5(s)=good)$. We need to compute the inverse, i.e. the probability that a student who has choosen answer C does indeed understand the concept: $P\,(X_5(s)=good \mid B_5(s)= answer\text{-}c)$. The Bayes formula will give (for the sake of brievety, we replace 'good' by 'g' and 'answer-c' by 'c')

$$P\big(X_5(s)=g \mid B_5(s)=c\big)=$$

$$\frac{P\big(B_5(s)=c \mid X_5(s)=g\big)\cdot P\big(X_5(s)=g\big)}{P\big(B_5(s)=c \mid X_5(s)=g\big)\cdot P\big(X_5(s)=g\big)+P\big(B_5(s)=c \mid X_5(s)\neq g\big)\cdot P\big(X_5(s)\neq g\big)}$$

In our case, if we integrate the probabilities mentioned above, we would have

$$P\big(X_5(s)=g \mid B_5(s)=c\big)=\frac{1\cdot 0.2}{1\cdot 0.2\;+\;0.25\cdot 0.8}=0.5$$

In this reasoning, we made the reasonable hypothesis that a learner who understood the concept will always correctly. However, there are factors that could explain why he selected the wrong answer, despite having a good understanding: badly formularted question, distraction, tiredness, a touch interface with buttons that are too small, and so on. So, the probability that he a learner who understood the concept could select the correct answer should be slight lower: $P\ (B_5(s)= answer\text{-}c \mid X_5(s)=good) = 0.90$.

$$P\left(x_5\left(s\right)=g \mid b_5\left(s\right)=c\right)=\frac{0.9\cdot 0.2}{0.9\cdot 0.2\ +\ 0.25\cdot 0.8}=0.47$$

Now the prior probability of being in state m (misunderstanding) is 0.8.

$$P\left(X_5\left(s\right)=m \mid B_5\left(s\right)=c\right)=\frac{0.25\cdot 0.8}{0.25\cdot 0.8\ +\ 0.9\cdot 0.2}=0.53$$

This example is interesting because it is counter-intuitive: in the case where the topic is known to be very difficult (20% 'good' states), there are less that 50% chances that a student who selected the correct answer among 4 possible ones, actually is in the state 'good'.

The diagnosis based on answer c is hence a vector of two probability values, one for the state "good" and one for the state "mis", that we represent as: $X_5(s)$~[.45 .53]. Let's imagine that 4 possible learner states have equal probabilities; namely, that the vector is $X_i(s)$~[0.25 0.25 0.25 0.25]. Uncertainty is then maximal since we have no clue about the actual state of the learner. Uncertainty is lower if the probability distribution is skewed; for instance with $X_i(s)$~[0.5 0.10 0.10 0.75]. We can therefore use the same measure of entropy of X_i (S) here as for any row of the transition matrix, as explained in Point 23.

The art of writing questions is to minimize this diagnostic entropy. Let's replace the statistics quiz (Figure 5.1) by a question "*Remove 3 numbers from the following list in order to minimize its variance*" and let the student answer by editing the list of numbers. As there are 504 possible answers, the effect of chance will be minimal.

Examples of entropy variations are given in Figure 5.3. The values in the vector $X_i(s)$ will vary with time and behavioral observations. Naturally, entropy increases with time; if a teacher writes on the blackboard for 3 minutes, his uncertainty increases, as he cannot visually estimate student attention for those 3 minutes. This is why one often says that a good teacher has 3 eyes, two at the front and one at the back. Conversely, a good question will reduce entropy. Asking students "*Is everything clear so far?*" marginally reduces entropy, as students tend to nod randomly. "*Invent an example that illustrates the same rule*" leads to lower entropy. In general terms, the difference between a good question and a bad question can indeed be expressed by the extent to which the diagnosis entropy is reduced by the learner's answer to the question; this reduction being hereafter referred to as the diagnosis power of an activity.

The diagnosis power of an activity a_i is the measure by which it reduces entropy: $H(X_i(s)) - H(X_{i-1}(s))$

Let's consider a fictitious example, with 4 possible states: $X_i(S)$={"*lost*," "*active*," "*fine*," "*brilliant*"}. For the sake of simplicity, let's consider that there is no prior information on the probability that the learner is in any of these states. The normalized entropy[1] at a_i is maximal: $Ho(X_1(s)= [.25\ .25\ .\ 25\ .25]) = 1$.

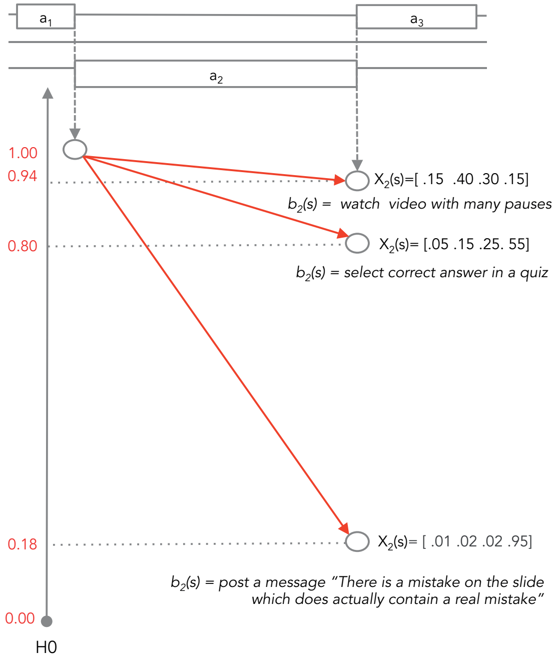

a_1
a_3
a_2
1.00
0.94
0.80
0.18
0.00
H0
$X_2(s)$=[.15 .40 .30 .15]
$b_2(s)$ = watch video with many pauses
$X_2(s)$= [.05 .15 .25. 55]
$b_2(s)$ = select correct answer in a quiz
$X_2(s)$= [.01 .02 .02 .95]
$b_2(s)$ = post a message "There is a mistake on the slide which does actually contain a real mistake"

Figure 5.3 Figure 30.2: Variations of entropy (number are fictitious, they have not been determined empirically):

- If the learner watches a video with many pauses, this may indicate that he is actively trying to understand ($x_2(s)$="*active*") or that he is taking notes ($x_2(s)$="*fine*"). This could lead to $X_2(s)$= [.15 .40 .30 .15], which corresponds to a minor decrease in entropy: $Ho(X_2(s))$ =0.94.
- If the student selects the correct answer in a quiz, the vector could be [0.05 0.15. 0.25 0.55], that is, the probability of the state "*fine*" or "*brilliant*" is not 1, since the right answer could be selected by chance.
- A message in the forum explaining a mistake in the slides would lead to a strong probability that the student has a deep understanding.

---

[1] As for the transition matrix entropy, the value of the diagnosis entropy depends upon the number of states in the diagnosis vector. In order to normalize it, this value can be normalized by dividing it by the maximal entropy, $log_2(p)$, where p is the number of states. The diagnosis entropy hence varies from 0 to 1.

Figure 5.3 shows how $Ho(X_2(s))$ would be reduced by three behaviors produced during a_2. The slope of the red arrows represents the diagnosis power of the activities.

If one computes the entropy vector for each of the q possible answer, considering each of p possible states, one obtains a diagnostic matrix. The entropy of this matrix (Figure 5.2) can be computed in the same way as for a transition matrix (Point 23). The computed value gives an indication of the intrisince quality of the question and proposed answers.

	State 1	State 2	..	State p
Answer 1				
Answer 2				
...				
Answer q				

Figure 5.2 Diagnosis matrix for multiple-choice questions.

The diagnostic power of activities is a key design point; diagnosis is not an issue to consider only at runtime, but also during the design phase. The craft of quiz design is to anticipate “distractors,” that is, answers that correspond to the most common misunderstandings. In other words, the design of activities has to satisfy two functions: maximizing learning effects and maximizing diagnostic power.

Figure 5.3 shows cases of entropy reduction, but sometimes entropy increases as well. It actually varies up and down with time along the graph activities. If, at some point, it is too high, the system should react by performing a new diagnostic act; for instance, triggering a new question. The threshold for deciding on an extra diagnostic act, that is, the maximal tolerable level of uncertainty, is discussed in Chapter 6.

Point 31 The diagnosis axis

In the previous chapter, the prediction of a state $x_i(s)$ was based on the previous state $x_{i-1}(s)$. As mentioned then, this prediction (based only on time) would neglect the main source of information—what the learner did during that activity, $b_i(s)$. Predictions would be more accurate if they could integrate both sources of information—the previous state $X_{i-1}(s)$ and the current behavior $B_i(s)$. As a basic example, let’s consider a transition matrix that predicts that if $x_{i-1}(S)$=”*fine*” then $x_i(S)$=”*fine*” with a probability of 0.96. Now, let’s consider the case of a particular student, Daniel, such as x_{i-1} (Daniel)=”*fine*.” If it occurs that b_i(Daniel)= 1/10 (he answered one question correctly out of 10), so Daniel probably does not belong to the 96% of students who experienced a smooth transition between a_i and a_j. In this example, the behavioral evidence is stronger than the time-based prediction.

However, the complementarity between time-based and behavioral-based inferences can work the other way around. Since behavioral inference is also uncertain, as explained in the previous point, the diagnosis uncertainty can sometimes be reduced by integrating time-based prediction. This complementarity appears in the following examples:

- If a learner provided the correct answer in a quiz and was previously in a "*good understanding*" state, the probability that this correct answer was due to chance is lower than if he were previously in a "*poor understanding*" state.
- If a learner rapidly explores different sections of a document, his state could be interpreted as being "*active*" or "*disoriented*." The first interpretation has a higher probability if the learner was previously in a "*good understanding*" state, since disorientation is more likely to appear for learners who don't have a good mental map to orient themselves in a complex space.
- If a learner is not contributing much to teamwork, as measured by the low number of messages in the shared workspace, there is a probability that he is in the "*free rider*" state ($X_i(s)$). However, as he was previously diagnosed as being in the "*leader*" state $X_{i-1}(s)$, the social loafing diagnosis loses its credibility. Sometimes a leader says few words, but important ones.
- If a learner makes many pauses while playing a MOOC video, it is uncertain whether he is struggling to understand it or whether he is simply pausing the video to take notes. If this learner rarely paused in previous videos, then the first hypothesis gains probability (why would he start taking notes only now?). Our recent results tend to show that video navigation patterns are difficult to interpret in absolute terms, but that a sudden change in pattern reveals a difficulty.

As mentioned in the previous chapter, hidden Markov models (HMMs) cope with these two sources of inference. Standard Markov chains capture the dynamics of a system that changes from one state to another; for instance, the variations of the stock market every day. These states are observable. In our case, the states are not directly observable, but are inferred from learner behavior. This inferred state is called a "hidden state." The relationship between hidden states (in our case, the cognitive state) and the observations (in our case, the behavioral traces) is often represented as in Figure 5.4.

In Figure 5.4, time is represented horizontally, from left to right, which is consistent with the graph representation used so far. To remain consistent with this representation, I prefer to keep the vertical axis for the social dimension, and I will apply a 3D rotation to this standard HMM representation. On Figure 5.5, the behavior is represented in the background while the

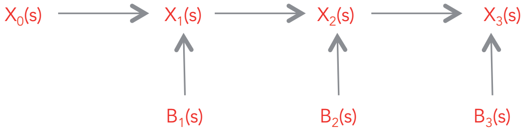


Figure 5.4 A standard representation of an HMM.

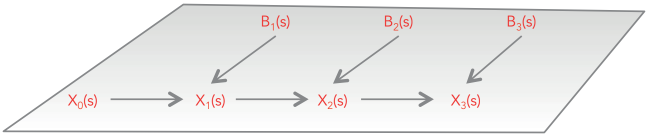


Figure 5.5 A rotated representation of an HMM.

hidden state is in the foreground. This is purely conventional; I only mention it because readers familiar with HMMs be mislead.

Whether a dimension is represented horizontally or vertically is not important, but what does matter in this figure is that the prediction of a cognitive state does not only depend on the behavioral evidence, but also on the previous states of the learner. This joint probability can be written as:

$$p\,(X_i(s) \mid X_{i-1}(s)\,,\, B_i(s))$$

Point 32 The modeling cube

So far, orchestration graphs have been represented with two dimensions; time captured by a sequence of activities (a_1, ..., a_n) and the social plane (π_1–π_6). In the previous point, I introduced a third dimension, the diagnosis axis, and proposed to combine these as time (horizontally) and diagnosis (depth). Figure 5.6 reintroduces the social dimension (vertical axis).

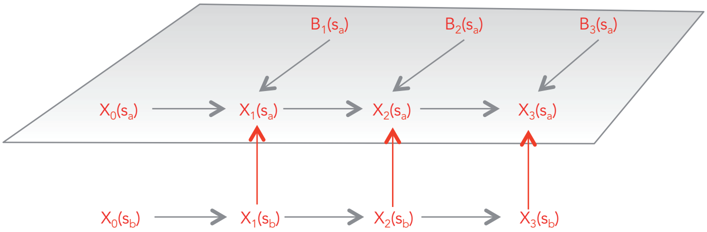


Figure 5.6 Reintroducing the vertical dimension (social axis) orthogonal to the horizontal dimension (time axis) and the depth dimension (diagnosis axis).

The social axis allows us to predict the state of a learner from the state of one other or many other learners. Here are some examples of situations where this axis introduces meaningful information:

- A teacher asks questions to five students, (selected more or less randomly from the class), and from their poor answers, he infers that the state of the whole class is probably unsatisfactory. This is not an optimal inference, but it's better than having no information at all.
- Even without knowing how Dinesh has behaved in the current activity, it can be guessed that he was successful, because every one else in the class succeeded, or because everyone with a similar profile succeeded.
- Even without knowing how Dinesh behaved in the activity, if everyone fails, there is a low probability than he will have succeeded.
- Even without knowing how a team collaborated on an activity, if all other teams failed to adopt the predefined roles, there is a low chance that this team will stick to the roles.

Building inferences along the social axis is especially relevant in the MOOC context; thousands of students engaged in the same graph give this axis its predictive power. However, a situation in which all students fail or all succeed rarely occurs. In most cases, inferring the state of a learner from the state of other learners requires taking into consideration the diversity of the class. The relative position of John in the current activity will be computed based on his relative position in a previous activity, which requires reasoning on both the vertical and the horizontal axes of the cube.

The meaning of the vertical axis is therefore somehow different to the way it has been used so far, that is, as an operational structure for orchestration. In stochastic models, the vertical axis uses social structures (e.g., two members of the same team) to compute probabilities. I hope the reader will forgive this breach of consistency in the book. Moreover, with a third dimension, orchestration graphs lose the simplicity I have been striving for, but this third dimension allows me to integrate three sources of information when modeling the learner's state:

Horizontally: John's state is predicted by knowing John's previous state:

$$p\ (x_i(s) \mid x_{i-1}(s))$$

Vertically: John's state is predicted by knowing the average class (S) state:

$$p\ (x_i(s) \ \mid x_i(S))$$

Vertically: John's state is predicted by knowing the state of a similar learner:

$$p\ (x_i(s_a) \mid x_i(s_b))$$

Depth: John's state is predicted from his behavior in the current activity:

$$p\ (x_i(s) \mid b_i(s))$$

A simple situation where multiple dimensions bring an advantage is, of course, when some sources are missing:

- If the goal is to make a prediction, that is, to model $X_i(s)$ before a_i has started for any learner, and only the horizontal axis is available.
- If the graph implements a new MOOC and neither the transition matrix nor data from other students is available, the diagnosis axis will be the only source of inference.
- If the state X_i(Mike) is unknown, because Mike has skipped the activity a_i. In a_j, Mike is watching the video, but since he did not answer the quiz at the end, we can't infer X_j(Mike) from B_j(Mike). However, if all the other students answered the quiz correctly in a_j, it is plausible that Mike is in a similar state. This situation may sound awkward, but orchestrating a pedagogical scenario includes the management of absences, late arrivals, unavailable data, and so on.

In summary, diagnosis can be enhanced by taking into account 3 sources of information: the individual history, the interpretation of behaviors, and social diversity. They contribute towards computing the probability that the learner is in a given state given his previous state, his behavior, and the states of the other learners. These 3 sources of information "converge" on the current state of the learner as represented in Figure 5.7.

$$p\,(x_i(s) \mid x_{i-1}(s)\,,\, x_i(S),\, b_i(s))$$

It is somehow misleading to call this space a cube and to represent it as a cube. It is a 3 dimensional space, but the dimensions are not truly orthogonal,

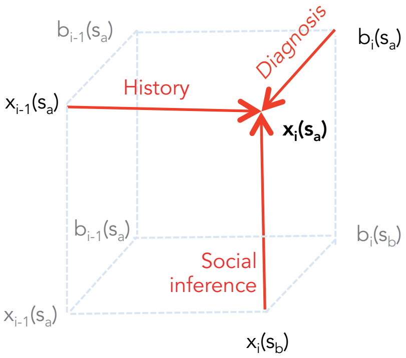


Figure 5.7 The modeling cube: Predicting the state of the learner combines information from 3 orthogonal axes—the diagnosis axis interprets behavioral traces, the time axis extrapolates from previous states of the same learner, and the social axis forms predictions based on the states of other learners.

since they are not totally independent from each other. Moreover, it won't probably be the case that position of the vertex is the same on each axis . So, it is not a cube mathematically speaking, but as you will have noticed by now that I am not a mathematician, please allow me to call it a conceptual cube. Instead, I will represent the space as in Figure 5.8, in which the position of $x_i(s)$ can vary on each axis. The position of $x_i(s)$ in this 3D space is determined by its position on each axis:

- On the horizontal axis (the history of the learner's state), the distance from the origin is the normalized entropy of the transition matrix $M^{i-1,i}$, as explained in Point 23.
- On the depth axis (diagnosis), the distance from the origin is the normalized diagnosis entropy, as explained in Point 30. The origin is the previous state, since to satisfy the Markov assumption, the previous state includes all previous states. This means that the origin shifts when moving to a new activity.
- On the vertical axis (social), $x_i(s)$ is predicted from the state of the class $x_i(S)$. The distance from the origin is the entropy of the distribution of states in the class. Let's consider the inference "*If all learners have failed, John will probably fail.*" If $Xi(S)$ has four possible states and "*failed*" is the first state, the class state $x_i(S)$="*all students failed*" can be represented by a vector [1 0 0 0], which has a null entropy. A class in which 25% of students end up in each of the four states would correspond to the vector [0.25 0.25 0.25 0.25], which has maximal entropy, normalized to 1. It is a bit awkward to use entropy as measure of class heterogeneity, the rationale being to use the same scale on each axis.

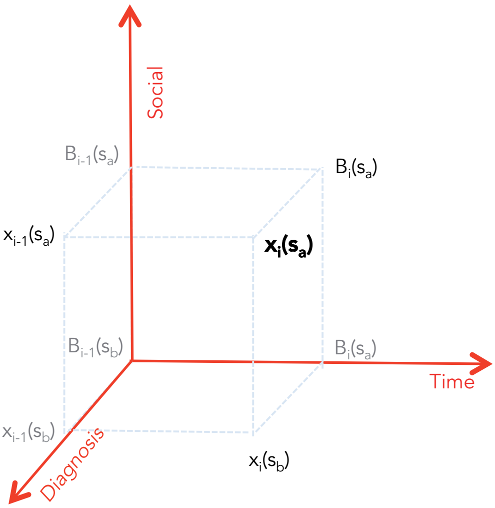


Figure 5.8 Figure 32.3: The *x,y,z* position of $X_i(s_a)$ is computed by measuring its normalized entropy according to each axis, that is, each source of the prediction.

At this point, I need to once again bring back some emotional values to this technical discussion.

> *One may question the ethical character of these predictions. If John is currently at* a_{10} *and failed the 9 previous ones, some optimistic teachers—I love them—will nonetheless believe, or at least hope, that John may succeed* a_{10}. *Determinism constitutes another tumor in schools, propagated by "reputation": a student with a reputation of being weak has low chances of succeeding because neither his teachers nor himself believe he will succeed. The "halo effect" (Foster & Ysseldyke, 1976) shows that this situation even crosses the limits of a discipline. However, the low variability of states is here considered as a statistical question, not as an ethical one.*

Point 33 Multidimensional predictions

The previous point explains that the cognitive state of a learner can be predicted by 3 sources of prediction: his previous state, his behavior, and the state of the other students. How to integrate these different sources of information into a single prediction? So far, I looked at a simple case where only one source is available. If several sources of information are available, are some sources more important than others? I do not ask this question in a general probabilistic context, but in the specific educational context of orchestration. A first principle is that, by default, **behavioral evidence is stronger than a prediction based on a transition matrix** (horizontal axis), since this matrix has been calculated based on many learners, while behavioral information is specific to one learner. For instance, even if a grammatical activity has almost always been successful, it may still be that it does not work at all for Christine because the sentences used as examples during a_i referred to a dog, and her dog has recently died. Humans are not fully predictable, fortunately. Now, this default rule—behavioral primacy—may sometimes not apply. In some cases, behavioral evidence can be less useful than the transition matrix. For instance, as I mentioned earlier, the interpretation of video navigation patterns is highly hypothetical. If the video in a_2 explains a new concept, and the learner had failed to learn the previous concept (a_1), which was a strict prerequisite ($\omega_{1,2} = 0.9$), there is a high probability that he will fail a_2, whatever his video navigation pattern turns out to be. In this example, the horizontal axis provides a better prediction than the depth axis. Finally, if $\omega_{1,2}$ is close to zero, the horizontal prediction may be less useful than the vertical prediction, for instance, the fact that other students have been successful at this activity. In summary, I propose the following principle for integrating 3 sources of prediction:

> Learner modeling integrates inferences from the 3 axes in a way that is inversely proportional to their uncertainty.

A challenge at this point is to turn this common sense principle into an equation. The evidence illustrated by each axis should have a weight inversely proportional to the normalized entropy. However, as I mentioned, the axes are not completely independent, and dependencies between variables have to be taken into account. I could not further elaborate this equation without empirical data. I hope that some statisticians will find it interesting to reframe the modeling cube within the context of Bayesian networks.

The cube model enables another way of integrating multiple sources of information, based on the geometry of the cube. So far, predictions rely on first order inferences: the general schema is “*What is* V1 *to* V2*?*” In our case, this question was instantiated by what is $x_i(s)$ to $x_j(s)$, what is $b_j(s)$ to $x_j(s)$, and what is $x_j(s)$ to $x_j(S)$? The cube geometry enables second order inferences such as “V1 *is to* V2 *what* V3 *is to* V4.” These inferences exploit the parallelism of edges, that is, the fact that the relationship between two variables can be compared to the relationship between two other connected variables. This relationship between relationships is represented by H-shaped patterns on the following figures.

Figure 5.9 illustrates a situation with two students Nina (s_a) and Mike (s_b). If we know Mike’s current state ($x_i(s_b)$), and if Mike and Nina have a similar behavior ($b_i(s_a) \approx b_i(s_a)$), one may infer that Nina’s state ($x_i(s_a)$) is probably similar to Mike’s ($x_i(s_b)$).

In more general terms, the difference between the states of the two learners (red vertical arrow Δx_i) is a function (red horizontal arrow) of the difference between their behaviors (red vertical arrow Δb_i). Since this function may not be known, the trick is to find “a Mike” that has the least behavioral

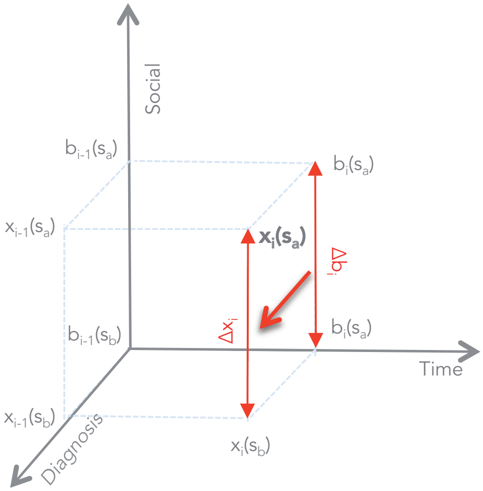


Figure 5.9 Inference based on the parallelism of twins—two learners with similar behavior must have a similar state.

difference with Nina at activity. I refer to **twins** as being two students who have very similar behaviors (or, later on, a very similar state). The notion of twin, and hence the inferences described in this point, necessitates using some similarity metrics. Metrics for quantitative data can be domain independent. For qualitative data (e.g., a small set of possible states), I am afraid we'll have to develop similarity measures that are specific to the activities included in the orchestration graph. This can be summarized by:

Diagnosis is parallel:

The difference between the states of two learners is related to the difference between their behaviors.

Figure 5.10 shows another form of second order inference, happening on the same face of the cube, but orthogonal to the former. The function (lower red horizontal arrow) that that has been applied to infer Mike's state $x_i(s_b)$ from Mike's behavior $b_i(s_b)$ can be re-applied (red vertical arrow) on Nina's behavior to infer her current state (upper red horizontal arrow).

This reasoning in Figure 5.10 makes especial sense for the quantitative processing of behavioral dust, since a general mathematical function or machine-learning algorithm may apply to different traces. For instance, the function that predicted Mike's degree of understanding from his response time or from his gaze patterns can be reused for Lena. This is the same for the previous second order inference example. These second order inferences would be useless if the diagnosis was a simple mapping function, such as in multiple-choice questions.

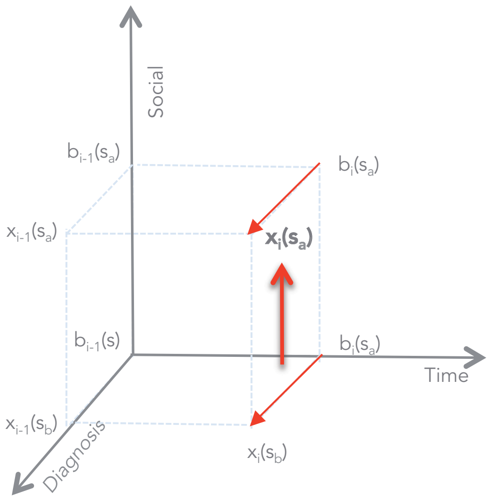


Figure 5.10 Inference based on the parallelism of diagnosis—a behavior is associated with a state if a similar behavior was associated with a similar state.

Continuing the same kind of reasoning, it can be reasonably postulated that students evolve in reasonably parallel ways, that is, their differences vary slightly, but not radically, from one activity to another. As illustrated in Figure 5.11, if Mike (s_b) and Nina (s_b) were in a similar state at a previous activity, we can infer the current state of Nina from the current state of Mike.

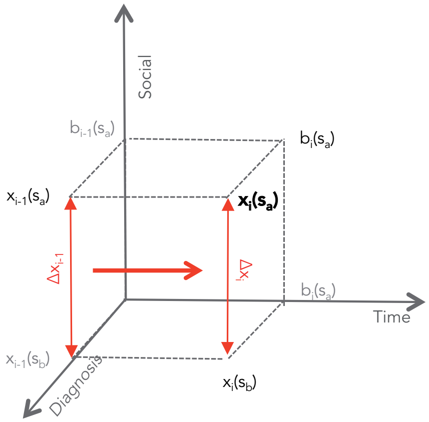


Figure 5.11 Inference based on the parallelism of individual evolution.

In more general terms, the difference between the states of two persons (Δx_i) is a function of the difference in their states at a previous activity (Δx_{i-1}). Since this function may be unknown, the trick is to find twins (i.e., s_a and s_b) that have the smallest state difference at the previous activity ($x_i(s_a) \approx x_i(s_b)$) and to use $x_j(s_a)$ as the most probable value for x_j (s_b). This can be summarized as follows:

The learners' volution is parallel:

The difference between the state of two learners is related to their difference in previous states.

The same type of second order inference can be made with respect to the whole class, S, instead of with respect to a twin; one can infer that the position of Mike within the class distribution at a_j (e.g., 2 σ above the mean of students' scores in the class) is probably similar to his position among the class at a_i, the previous activity. In this case, we don't compare $x_{i-1}(s_x)$ at $x_{i-1}(s_y)$ but at $x_{i-1}(S)$. Being "2 sigma below" constitutes an example of social ranking that is no more pleasant that the individual determinism I criticized earlier. A phenomenon can be statistically true, even if ethically questionable. Using standard deviation applies if the state is stored as a metric value. For ordinal values, that is, an ordered set of categories, this determinism will be translated as staying in the same category.

Evolution is parallel: The position of a learner in his class remains relatively stable over time

Figure 5.12 illustrates another type of second order inference: the diagnostic process is stable over time. If, in a previous activity, a diagnosis function predicted the learner's state from his behavior, we can reuse the same function to infer his current state. For instance, let's consider the two ways of interpreting an incorrect numerical value, as reflecting two possible misconceptions of the domain; if an interpretation is chosen at time *i*, the same interpretation should be selected at time *j*.

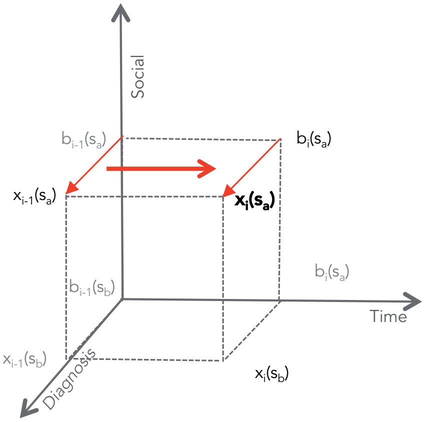


Figure 5.12 Inference based on the stability of diagnosis over time.

Conversely (Figure 5.13), the evolution of a learner's state is parallel to the evolution of his behavior; if we know how much his behavior has changed over time, we can apply the same transformation to his previous state in order to compute his current state.

In general terms, the difference between two successive states of a learner is a function of the difference between his two successive behaviors. Since this function may not be known, we could use the case where the learner had the same behavior before: "Find $b_i(s)$ that has the smallest difference with $b_j(s)$ ($i<j$), and use $x_i(s)$ as the most probable value for $x_j(s)$." This rule is not useful for behaviors that can't be compared across activities (such as the answers to different multiple-choice questionnaires), but some variables can be

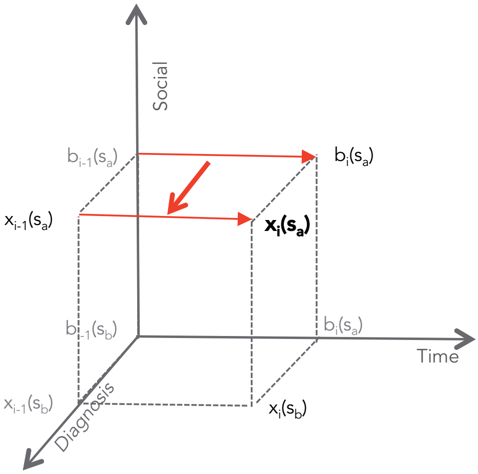


Figure 5.13 Inference based on the stability of diagnosis over time.

compared. For instance, if high gaze-movement frequency was interpreted as an indicator of engagement in an activity, a similar frequency in a later activity could lead to the same interpretation.

Finally, I could go further in complexity and use the 3 dimensions. Figure 5.14 can be read as follows: if the difference of behavior between s_a and s_n is the same now as it was before (this equivalence being represented by the H-shape on the back facet of the cube), then their difference in state should be the same now as the difference in state before (represented by the H on the

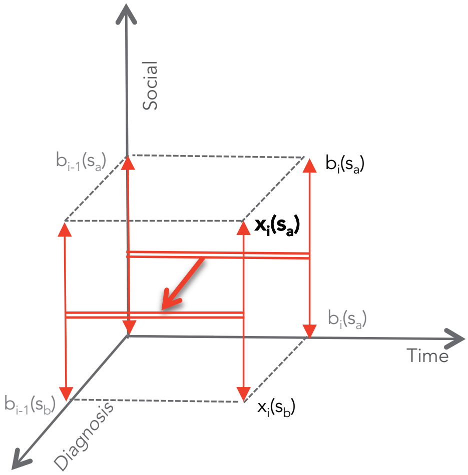


Figure 5.14 Third-level inferences.

front facet). Such an inference would be more meaningful for behavioral dust variables (such as navigation patterns or gaze patterns) than for quiz answers.

It could be argued that these 3 dimensions actually hide the many dimensions that can be portrayed in a generalized linear model. My point is that these 3 dimensions are ontologically different; they correspond to different forms of reasoning and should hence be treated distinctively.

Conclusions

This chapter enriched the orchestration graphs with some general mechanisms by which the cognitive state of the learner can be inferred or predicted. If the graph defines the pipes of analytics, these principles define the mechanics of the pump that circulates data, that is, the analytics engine. The proposed principles have to be refined and/or expanded, both from mathematical and from computational viewpoints. Now, analytics is not a goal, per se, but only a tool for improving the efficiency of the pedagogical scenario and for investigating human learning processes. Therefore, the last chapter is devoted to methods for adapting an orchestration graph, modifying it and learning from it.

Chapter 6
Orchestration

As mentioned in the previous chapter, analytics are not conducted for the sake of analytics, but because collecting information is necessary for making decisions with respect to a pedagogical scenario. These decisions range from a temporary adaptation of the activities to deeper changes in the scenario. I refer to changes in the graph with the term “evolution”; an orchestration graph evolves in order to maximize its fitness to the classroom ecosystem or the online education ecosystem.

This chapter adopts an engineering perspective; the role of analytics is to increase the current or future effectiveness of an orchestration graph. However, beyond engineering, analytics also enables research. Data are analyzed to extract generalizable knowledge; for instance, by measuring the effect of a design parameter on learning gains. The goal of research is not to optimize a particular system, but to enrich learning sciences. Even if this has been my day-to-day job for many years, I will only briefly mention research in this chapter, as it would deserve another entire book.

Point 34 Scope and space

The evolution of a graph includes four different types of modifications presented in Table 6.1.

- The scope of a modification defines its social and temporal amplitude. The two columns discriminate local versus global modification. An example of “local” modification is to select the difficulty of the next exercise for one learner in the current session. Consequently, a local modification concerns one or a few students, but not future sessions. An example of “global” modification is to replace an unclear question with a clearer one. Such a modification will not only concern students currently engaged in the graph, but also participants in future sessions.
- The two rows of Table 6.1 indicate whether the modification mechanism has been defined or planned, that is, included into the design of the graph, or whether it is performed manually, improvised, or created on the fly. In the row “defined,” the range of possible modifications is limited to those that have been anticipated by the designer; for instance, it is possible to

change the number of exercises, but not the number of examples. The row "open" includes any possible modification of the graph.

Table 6.1 Four mechanisms of modification.

Space of modification	Scope of modification	
	Local	Global
Defined	Adaptation	Optimization
Open	Repair	Redesign

The rationale for discriminating between these four types of modifications is that some confusion has emerged in the past between adaptive instruction and orchestration. The same general idea underlies each of processes described in Table 6.1—to modify the graph based on how effective it has been up to that moment. However, this general idea is operationalized in very different ways across the four categories of modifications, which I will now describe one by one.

Adaptations: This process is central to "adaptive instruction." This can be a somewhat misleading term, as it actually means a differential adaptation, that is, adaptation to individual differences. Of course, nothing prevents adapting the learning activities to the level of the whole class. However, the core idea is that all learners are different, and that the strength of digital education is the possibility of adapting instruction to individual needs. This principle, called "individualization," has been central to learning technologies, from the first drill and practice software in the 1960s to the MOOCs of today. Adaptation can be performed by the system, by the teacher, by the learner, or by any combination of these actors depending upon the distribution operator (Point 15). Individualization includes predicting the future state of the learner in order to preventively adapt activities. In digital learning environments, adaptations have been planned; the system has rules that detect specific learner states; for example, "$x_i(s)$= "low-level understanding," and that decide what needs to be modified; for instance, increasing or decreasing the level of difficulty. The adaptation mechanisms can be modeled as a set of if-then rules: if X happens, change Y. The set of events that trigger an adaptation (e.g., the specific learner states) and the set of possible modifications (what is changed and by whom) have been specified in the design stage: **the space for adaptations is predefined**. When the teacher or the learner decides on the adaptations, this space can be a list of menu items. When an algorithm decides on the adaptations, the space can be a set of predefined parameter values. In a classroom, the teacher may have prepared two sheets of exercises and distributes the easy or the difficult ones to students based on their state. Two types of adaptations are common in education graphs:

- Branching: selecting a_{i+1} among a set of predefined candidates, and in some cases, selecting a different a_{i+1} for different subsets of S.
- Parameterization: selecting a new value for one or more parameters of the next a_i, which will determine what students will actually do in $a_{i.}$

The space of predefined adaptations can be large, but it is finite. In adaptation, the learner path in the graph is modified; it is set to one of the paths integrated in the graph. The graph itself and the activity definitions do not change. This discriminates adaptations from the other categories of modifications defined in Table 6.1.

Repairs: The repair process copes with situations that have not been taken into consideration in the design phase. A scenario rarely unfolds as it was planned. Education is far from being an exact science. Unexpected events occur inside the didactic space of the scenario: the explanation was too difficult for all students: the student has to collect various samples in a field trip, but most of them brought the same sample back; two students started fighting with each other; and so on. In addition, many unexpected external events can perturb the learning process. Education does not happen in a vacuum. For instance, half of the students arrive after the phase where instructions were given, because of a train breakdown; the automatic grader (in a MOOC) has a bug; there is so much noise outside the classroom that the teacher does not hear the students' questions; students are not paying attention because they have an exam in another class right after the lesson, or because their soccer team won the night before. My favorite example of an external event is when a crane stops and starts to operate in front of the classroom windows, irresistibly drawing the attention of all the boys in the class. In other words, the set of events that require modification can be infinite. If one of these unexpected events occurs, the set of possible modifications can also be infinite; like those who use a screwdriver as a hammer, a creative teacher may invent any number of ways to fix the problem. For the previous example where half of the class

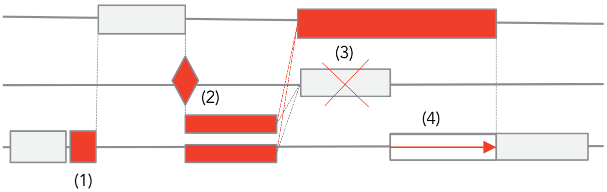


Figure 6.1 Examples of evolution acts. This is a math class attended by students from chemistry and from biology. (1) Repair: The teacher realizes students are not mastering the skill and decides to add 5 exercises for all students. (2) Adaptation: After a short lecture, the students can choose between exercises that concern math applications to biology or exercises about chemistry. (3) Repair: As these new exercises are not very successful, the teacher decides to skip the team activity and to replace it with a new step-by-step explanation, which leads him to shorten the time devoted to the final individual activity (4).

arrives late, he could make groups with 2 students who got the instruction and 2 latecomers. For the noise problem, he will ask students to save their questions and do the exercises in the meantime, that is, permute two activities. These "repairs" cope with unexpected events; teachers have to invent solutions on the fly.

The repair actions are central to the concept of **orchestration**. Orchestration is much more than just adaptation; it includes routine adaptation, but also pays attention to these improvised repairs without which no educational system would work. These changes made on the fly by the teacher somehow contradict the orchestra metaphor, since the conductor does not change the score during the performance. Jazz improvisation would be more appropriate; jazz players do not play completely freely—they also follow some kind of graph, but have the freedom to explore the space around that skeleton. In other words, the word "orchestration" can be somehow misleading when taken in a strict sense, depending upon the dictionary being referred to. Therefore, it is important to clarify that classroom management includes repairs, improvisation, mess management, or even chaos management. The key idea in orchestration could be captured by an oxymoron—**design for what cannot be anticipated**. As far as possible, the technology should enable the teacher to modify any design decision at any time. This is why I often refer to orchestration as "empowering the teacher," which means leaving him the possibility—as often as possible—to bypass any choice made during the design. This flexibility is mainly a technical issue. For instance, some readers may remember that it was easier to permute two slides during a lecture when we were using plastic transparencies than it is now with PowerPoint. The flexibility of a graph is hence a central concept of orchestration (Tchounikine & Dillenbourg, 2007); it depends on the way the operators and the workflow has been designed, as well as on the weight of the edges.

Optimization and **Redesign:** Let's now move to the second column of Table 6.1. It contains persistent improvements in the graph between two sessions, that is, modifications that should make the graph more effective in the next sessions. Improvements can be produced in two ways:

- Redesign: the teacher/designer uses analytics to find out what should be changed and then, as in repairs, manually performs these changes on the graph. The borderline between repair and redesign is shallow: a repair is a plaster that is not supposed to stay for long.
- Optimization: Learning traces are automatically processed by machine-learning algorithms in order to optimize the efficiency of the scenario. The application of machine learning to educational environments has led to the concept of self-improving systems developed in this chapter.

Actually, this chapter turns common practices in digital education up side down; while adaptation is often automated, I stress the possibility for manual repairs; conversely, while redesign is usually manual, I stress the importance of machine learning for improving a graph over time. In the following points, I describe the adaptation and repair processes at a certain level of abstraction; namely, in terms of constraints satisfaction. The design of an orchestration graph has to satisfy multiple constraints, presented in the next point. When the scenario is being conducted, some of these constraints may be violated. Orchestration consists in changing the graph in a way that restores satisfaction of the constraints, and this at the lowest cost. Alternatively, the teacher may also decide to relax some of the constraints and keep the graph as it stands.

Point 35 The constraints library

Before analyzing how a pedagogical scenario can be modified, one has to understand the constraints that shape its design and pave its daily use. I have mentioned the need for storing some elements of the design rationale in the graph data structures several times. Therefore, one slot of the activity description is devoted to design rationale. In other fields, such as architecture and engineering, design has been described as the search for a solution that satisfies multiple constraints. This can constitute an optimization issue, since constraints are often in conflict with each other, at least partially. For instance, the design of a cross-country shoe has to be flexible on the longitudinal axis, but nonetheless rigid on the lateral axis; warm, but nonetheless light; robust, but cheap. This constraint-satisfaction perspective is also relevant for instructional design; the pedagogical scenario has to go deep into content but not for too long, in order to be interesting for the best student, but still be feasible for weaker students; to provide both compiled procedural skills and higher-level thinking skills; and so on.

Now, let's look at a specific graph has that been designed in order to satisfy certain constraints. While the graph is running, some constraints may suddenly be violated: "*To stay within 50 minutes,*" "*To keep all students 'with me',*" may be true at time t, but not at time *t*+1. These violated constraints will require a repair action. In turn, some repair actions may actually violate other constraints. The graph evolution process will hence be described in terms of detecting violated constraints and repairing them. In this analysis, it is relevant to discriminate constraints that are intrinsic from those that are extrinsic to the learning processes (Dillenbourg & Tchounikine, 2007; Dillenbourg & Jermann, 2010). Intrinsic constraints come from the processes by which people learn. Extrinsic constraints come from the context in which education is conducted. As in previous chapters, I have structured the set of constraints as a library. The list of intrinsic constraints is limited to the four constraints in the library, but each of them can be decomposed into many constraints. The list of extrinsic constraints can certainly be expanded with many more examples.

Table 6.2 A library of constraints to be satisfied by orchestration graphs at design stage, but also while running.

Intrinsic constraints	Extrinsic constraints
Student profiles	Scale
How people learn	Time segmentation
Domain epistemology	Discipline
Learning time	Fairness
	Teacher's energy
	Finances
	Funding
	Producing grades
	Culture
	Safety
	Technology
	Learner satisfaction
	...

Firstly, instructional design aims to satisfy four **intrinsic constraints**, that is, constraints that are intrinsic to the learning process of human learners—namely, **who** is learning **what**, **how** and for **how long**. In principle, these constraints should mostly shape the design of the orchestration graph, but also have to be respected when the scenario is conducted.

Student profiles: *An orchestration graph has to maximize learning gains for the specific set of students* S. Civil engineers do not learn the statics of buildings in the same way that apprentice carpenters do. What are the specific features of the students (*S*) that will engage in the scenario? What are their prior knowledge and skills? What motivates them? Do they have specific learning styles? To which culture do they belong? How old are they? As mentioned above, this constraint actually includes many variables.

How people learn: *An orchestration graph has to take into consideration the cognitive processes by which* S *actually learn*. An example of this constraint is cognitive load. Cognitive load is not a learning theory, but a constraint on the performance of our learning machine; simply stated, human cognition is limited by the ability to store more than a few elements at the same time in its working memory.[1] Constructivist learning activities have been criticized for neglecting this constraint; a nice exploratory learning activity may

---

[1] To measure your cognitive load, try to answer the following question: if Mike is John's brother, John is Susan's father, Susan is married to Helmut, Irmi is Helmut's mother, Nathalie is Helmut's niece, and Pierre is Nathalie's brother, who is Pierre's grand-father? The difficulty of this question is not the understanding of relationships, but the temporary storage of each relationship into a buffer—called a working memory—to mentally constitute the genealogical tree of this family.

fail because it requires students to keep in mind too many pieces of information at the same time, and therefore they simply cannot carry out the learning activity. The solution is obviously not to abandon constructivist activities, but to design them in such a way that takes this constraint into consideration.

Domain epistemology: *An orchestration graph has to take into consideration the intrinsic features of the contents to be learned or skills to be acquired*. We do not learn history in the same way that we learn algebra. The design of any effective learning scenario requires a deep understanding of the knowledge structures that learners will have to elaborate. This does not mean that every teacher has to have a deep knowledge of the epistemology of the field, but rather a good grasp of the mesh described in the conclusion to Chapter 2. Actually, what matters more is not the scientific field, but the specific learning goals in the field. Many would agree with the claim "*You don't teach computer science in the same way that you teach Latin*." Actually, the difference between these two disciplines may be more cultural than cognitive. Are syntactic structures of Latin and XML so different from each other? There are more difference inside disciplines (e.g., between algebra and geometry) than between disciplines. There are differences between teaching a concept and a procedure, and between teaching facts versus systems. It remains nonetheless true that any good design has to rely on a deep understanding of what is to be taught.

Time: *An orchestration graph has to maximize learning gains within a certain time budget*. The time available for a specific lesson is an extrinsic constraint, but, at a more general level, time is actually intrinsic to any learning, since learning refers to a difference of knowledge or skills between two points in time. The person who decides to learn to play the guitar or to speak Farsi probably plans on devoting some time to it. If the learning outcomes achieved within this time budget are not satisfactory, he may either lose motivation and drop out, or decide to invest more time. Another instance is the "cost-in-time" of pedagogical scenarios; an intrinsic drawback of discovery learning is that this method needs much more learning time than, for instance, lecturing. A common solution is to use discovery methods for elements that students still need to remember in 10 years time, to conduct lectures for elements that are important, but not critical, and, finally, to cover the rest of the contents with optional readings. This segmentation can be rephrased as follows: the time devoted to learning a specific set of skills should be proportional to the importance of these skills for the students.

If education occurred in a vacuum, the designer would only pay attention to these four intrinsic constraints. The reality is, of course, different. In formal education, the institutional context generates multiple **extrinsic constraints**, that is, constraints that are remotely related to learning, but are more related to the social and physical environment in which learning occurs.

Scale: *An orchestration graph has to maximize learning gains for a given number of students*. This constraint has been presented in the introduction; a method that would be optimal for 10 students might be intractable for 100 of them. The scale changes the nature of interactions between the students and the teacher, and the type of assignments that can be graded within a reasonable workload

Time segmentation: *An orchestration graph has to maximize learning gains despite the time segmentation.* The global time budget for a scenario is an intrinsic constraint; it is defined by the curriculum to which the scenario belongs. However, the segmentation of scenarios into times slices (50 minutes, two hours or one 3-day workshop) is an extrinsic constraint generated by our institutions; namely, by the availability of teachers and teaching rooms. The segments of MOOC lectures in 8-to-15-minute videos could be considered as an intrinsic constraint, since it is justified by the learners' limited attention span—one constraint of human cognition.

Discipline: *An orchestration graph has to generate rich social interactions while maintaining a reasonable level of discipline.* In a physical classroom, there is no life without a certain amount of noise. Silence is not a condition for learning. However, if the students are out of control, the teacher will soon have problems within his classroom. I have seen examples of scenarios that rely on interactive tabletop environments that have to be operated in the dark. Try this with 30 children! In a MOOC, discipline issues mostly concern discussion forums where the teacher is somehow responsible for neutralizing insults, racist utterances, and so on. Several corporate actors that have discussed MOOCs with us were concerned by the risk of dissatisfied customers posting forum messages that would be detrimental to the image of their company. How to smoothly manage discipline in a class may not sound like a very interesting issue in learning sciences, but it remains a major concern for many teachers, especially those who are new to the job.

Fairness: *An orchestration graph has to adapt learning activities to individual needs, while maintaining fair conditions for success among all learners.* This constitutes an implicit, but ubiquitous constraint on educational practices. For instance, if a team loses a member, this may—depending upon the task—increase the workload for the remaining team members. Is this fair? It is genuinely expected that the requirements for a certain certificate or a certain number of credits are the same for all learners. It is accepted that the effort to reach these requirements varies across learners, but the requirements themselves should be stable across learners. Plagiarism and exam cheating are examples of unfair situations, since the cheater acquires the same certificate without meeting the requirements.

Teacher's energy: *An orchestration graph has to maximize learning gains without increasing the teacher's workload too much.* A graph may fail because it overestimates the time available for the teacher to prepare, conduct, or evaluate activities. We have witnessed pedagogical reforms, such as "problem-based learning," that have been applied to ambitious curricula, but that were stopped after 3 to 4 years, because the workload (which was fine during the novelty phase) became unbearable when the enthusiasm faded out. In empirical research, new pedagogical methods are often tested with teachers that are especially motivated and therefore invest an amount of time superior to what can be expected from the whole population of teachers, or even from the same teacher, over a longer period. This partly explains why some methods that have been proven effective in empirical studies fail to generalize across an educational system. This energy limitation does not only concern the teacher, but the whole teaching team, including assistants, as well as any other actor involved. This notion of "energy" does not only refer to the amount of time a teacher needs to prepare or grade, but also to the amount of energy required in the classroom to orchestrate the activities, which we refer to as **orchestration load** (Point 41).

Finances: *An orchestration graph has to maximize learning gains for a given budget*. No need to explain that the available budget is a constraint. The budget covers the number of teaching assistants (and therefore the global amount of a teacher's energy), as well as teaching resources such as books, scientific equipment, technologies, bandwidth, consumables, and travel for field trips.

Funding: *An orchestration graph has to take into consideration the origin of the funding that enables an educational action*. For instance, it is expected that when participants pay high fees, the rate of failure is expected to be reasonably low. From an ethical point of view, the rate of success should not vary between public and private schools, but the reality is different. Conversely, the high dropout rate for MOOCs was tolerated, because they were accessible for free. We recently compared the MOOC strategy developed at the EPFL and at the University of Edinburgh, two leading universities in the European MOOC landscape. We found out that most of the strategy differences related directly or indirectly to the amount of fees paid by different categories of students.

Producing grades: *Many orchestration graphs have to produce grades*. The need to evaluate (i.e., to determine the learner state) is intrinsic to learning and to teaching processes, such as giving feedback or adaptation. However, in many formal educational contexts, producing grades is also an extrinsic constraint; a teacher has to provide grades on a regular basis to comply with the school rules. For instance, some teachers reject collaborative learning, because their school requires an individual measure of a student's skills. In the same vein, parents, learners, and teachers are used to receiving physical traces of student activities; documents, notes, textbooks, and posters are among the many artifacts that make learning tangible, even though it is an invisible process.

Culture: *An orchestration graph has to be compatible with the specific culture of the educational context in which it is used*. This culture partly determines what can be done or not done in a pedagogical scenario. International comparison studies, such as PISA, have to carefully avoid instances that exclude some countries; the sentence "*For Christmas, he decided to build a snowman in the garden*" makes no sense for children from non-Christian countries, from hot countries, or from the southern hemisphere. The culture constrains the teaching style. For instance, starting a course with the sentence "*Let's define* S *as a non-empty sequence of integers*" belongs to the culture of an engineering school, but not of a vocational college. As another example, we encountered difficulties while using a scenario with collaborative activities in a medical school, because their first year was competitive (only the *N* best students pass) and so collaboration was not part of the students' culture. Conversely, we encountered success when training warehouse workers with a tangible interface, because the physicality of a tangible interface was part of their culture.

Safety: *An orchestration graph has to minimize the risks taken by learners*. In formal education, teachers are responsible for the safety of their students. Some British colleagues reported that they had to modify a great learning scenario, where students would explore a city in small teams before aggregating data (e.g., noise level, pollution level) at many points around town. The school rules prevented teachers from leaving teenagers alone in the city; so they explored the city together and the core idea of the pedagogical scenario vanished. In the case of MOOCs, safety mostly translates into privacy: what is stored, where it is stored, and what is accessible to whom. This currently raises major concerns.

Technology: *An orchestration graph has to cope with the constraints imposed by the technologies it uses*. Of course, many technological constraints exist. Some of them are trivial, such as the bandwidth or the size of a display (e.g., it may be difficult to do complex mathematic symbolic manipulations on a smartphone). Other technology constraints are less visible, such as the reliability of automatic grading software or the flexibility of a workflow.

Learner satisfaction: *An orchestration graph has to maintain a reasonable level of satisfaction among learners*. This constraint may sound awkward, but it is actually ubiquitous in education. Many universities use teaching evaluation surveys that measure student satisfaction. This process is often criticized as a symptom of consumerism, but my experience—I have been in charge of this for 10 years—is that student opinions are actually quite accurate. They focus on the quality of content more than on the scenic performance of the teacher, and their opinion is not correlated with course difficulty.[2] In elementary and secondary schools, user satisfaction includes parents' satisfaction, measured indirectly by the frequency of complaints to the school principal. This constraint is especially strong in private schools. In MOOCs, this is becoming a concern. Several top institutions envisage MOOCs as a way

---

[2] We found a correlation of r=0.14 (based on 600,000 pairs of data) between the grade that the student gave to the teacher through a student satisfaction questionnaire and the grade that the teacher gave to the student at the exam. Easier courses do not generate more student satisfaction.

to maintain their relationship with alumni. Alumni would then receive a kind of diploma with a maintenance clause, which means that their alma mater will offer a regular refreshment of their expertise. Since alumni are the main economic actors in some American university budgets, they have to make sure that their alumni complete their MOOC with a high level of satisfaction.

Some pedagogical methods, despite being poorly justified by intrinsic constraints, are nonetheless widespread in formal education because they satisfy extrinsic constraints. For instance, lecturing is ineffective from the intrinsic viewpoint, but very efficient from the extrinsic one. Another instance where extrinsic constraints override intrinsic constraints is team formation (π_2). On the one hand, intrinsic constraints suggest making teams of 2 to 4 students for a convergent task, because larger group sizes lead to phenomena such as the free rider effect (Point 21), which are detrimental to learning. On the other hand, if there are many students in the class, extrinsic constraints recommend forming teams of 5 or 6 in such a way that the teacher has to grade fewer assignments—one per team (teacher's energy constraint).

When educational researchers conduct experiments in real classrooms, it could be expected that their methods take these extrinsic constraints into account. However, extrinsic constraints are often partly waived in these field studies. For instance, it often occurs that, for the sake of research, the teacher agrees to spend more time on a topic than he normally would or to adapt the class schedule. This means that many field studies in authentic classrooms are still not 100% ecologically valid, that is, not fully representative of genuine classroom conditions.

My point is NOT that extrinsic constraints are more important than intrinsic ones; the goal of an orchestration graph is that students learn, not to please the institution. Let me repeat this: the goal is that students learn. However, extrinsic constraints determine the feasibility of the graph; an efficient unfeasible graph is as useless as an ineffective feasible graph. Simply put, both intrinsic and extrinsic constraints need to be satisfied. The reason I have spent more time describing extrinsic constraints is that in the field of learning sciences they do not have the epistemological status they deserve, compared to their role in the success of educational practices. Extrinsic constraints are often considered to be implementation details. I would instead suggest conceptualizing them at the theoretical level in order to integrate them into the design process, as well as into adaptation mechanisms. This is how we will come to the notion of "classroom usability" developed in Point 42.

Since I have expressed design as a constraint satisfaction problem, adaptation can be rephrased as attempting to **maintain or restore constraints satisfaction**, which can be done in two ways. The first way to cope with constraint violation is to *restore constraints satisfaction*, that is, to adapt the graph in such a way that constraints are satisfied again. The alternative is to **relax some of the constraints** that have been violated. For instance, if too many students fail the exercise, adding explanations restores the effectiveness constraint, while postponing the deadline relaxes the time constraint. The decision to restore or to relax constraints depends upon the nature of these constraints. If an intrinsic constraint is violated, the priority should be to restore it by adapting the graph. Conversely, if an extrinsic constraint is broken, consideration should first be made to relax it in order to avoid breaking down intrinsic constraints. Intrinsic and extrinsic constraints determine the flexibility of a graph in different ways, as illustrated in Dillenbourg & Tchounikine (2007).

Adaptation searches for the cheapest way to maintain or restore intrinsic constraints satisfaction, by modifying the graph or by relaxing constraints.

Figure 6.2 illustrates this process of constraints restoration/relaxation with ArgueGraph, the graph described in Point 13. Forming pairs of learners with opposite opinions (operator noted as 1 in Figure 6.2) is an intrinsic constraint and should not be changed, otherwise the whole pedagogical idea behind the graph collapses. Conversely, the final summary that students had to produce (denoted by 2 in Figure 6.2) was partly due to an extrinsic constraint (producing notes) and could be skipped; for instance, if time is too short. The flexibility of this graph is encompassed in the weight of the edges.

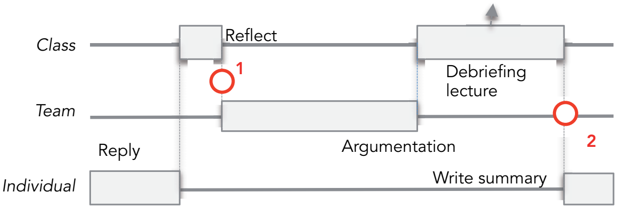


Figure 6.2 Adaptation and constraints in ArgueGraph.

How do we know which constraints can be more or less relaxed or which ones are broken if an activity is skipped? To anticipate this question, the graph needs to keep trace of the *design rationale* behind each activity: what are the set of intrinsic or extrinsic constraints associated with an activity? This design rationale is stored in the activity's metadata field (Point 4) and in the label field for edges (Point 5).

Now, if in the same scenario, the teacher feels that the conflicts emerging in the class are becoming too emotional, he may decide to form pairs with similar opinions. This means that he deliberately violates an intrinsic constraint (conflict generates learning) in order to satisfy an extrinsic constraint (safety); lesson failure is more acceptable than classroom violence. Teachers should be able to bypass any designer's decision, because they know the context. In terms of design, this means that any platform should encompass a “bypass” function by which the teacher may manually modify decisions made at design stage. This illustrates what I mean when I claim that learning technologies should empower teachers (Dillenbourg, 2013).

Point 36 The economy of evolution

Many forms of adaptation and repair exist; the teacher may change exercises because they are too difficult for all the students, he may reorganize activities because the class is too heterogeneous, or he may add an extra question, because he is not sure if the students have properly understood. These various examples do not have the same cost; for instance, in terms of teacher workload. To analyze these costs, I have further structured the space modifications applied to an orchestration graph. Table 6.3 shows that various modifications can be associated with the three axes of the modeling cube (Point 32), as well as with the 4 categories of evolution that were presented in Table 6.2.

Table 6.3 Classifying the modifications with respect to the modeling cube.

Cube Axes	Type of Modification	
	Defined (Adaptation/Optimization)	Open (Repair/Redesign)
Horizontal adaptation (time axis)	A student who reached a high score in a_1 may skip a_2 and go directly to a_3.	As the average team performance is low, the teacher decides to postpone the exercises and re-explain the principles with a new video.
Vertical adaptation (social axis)	Based on the answers to the video quiz, the class is split into two subclasses. Every subclass is assigned either standard or advanced exercises.	Given the difficulties observed by the teacher, he asks an assistant to work with the three students in trouble.
Depth adaptation (diagnosis axis)	If the learner state reaches a normalized entropy above 0.6, the system automatically generates 3 more questions.	If the analytics produced too much uncertainty, the teacher will modify the items for the next session

Adaptations or repairs are triggered if a specific learner state (Point 21) has been detected during the activities. The criteria or parameters that trigger evolution will be different on each axis of the modeling cube:

- **Drift (horizontal axis):** Adaptations or repairs are required if the state of the learner is too far from what it should be in order to maintain the effectiveness of the next activities. This increase in distance between the actual and desired state is referred to as "drift." If the drift is too large, activities must be adapted.

- **Heterogeneity (vertical axis):** Adaptations or repairs are required if the heterogeneity of the class is too high to guarantee the effectiveness of the next activities. In this case, the class could be split for the next activities or new activities that reduce heterogeneity have to be introduced.
- **Uncertainty (depth axis):** Adaptations or repairs are required if the teacher or the system has too much uncertainty with respect to the state of learners. If it does not know whether learners are in the appropriate state or not, the system should try to find out.

On each axis, the system or the teacher has to monitor the evolution of a parameter (respectively, drift, heterogeneity, and uncertainty) and to modify the graph if the value of any of these parameters passes a predefined threshold. The extent to which the current parameter is higher than the predefined threshold tells us how much adaptation is necessary. It is called the "necessity" hereafter. This mechanism applies to each axis and is represented in Figure 6.3.

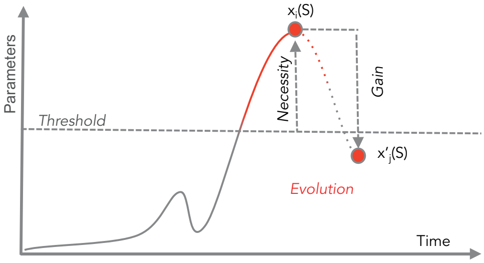


Figure 6.3 Adaptation is triggered if one of the monitored parameters represented by the vertical axis (i.e., drift, uncertainty, or heterogeneity) passes over a certain threshold.

The necessity establishes that some modification of the graph should be performed, but it does not determine which one. In order to choose among several possible modifications, two other considerations have to be taken into account: the predicted benefit and the cost of modifications, described hereafter.

The benefit expected from an evolution, hereafter referred to as the gain or the **predicted benefit**, measures how much the problem at hand would be reduced if this modification were performed. I call it "benefit" despite the fact that, in Table 6.3, it is represented as a reduction, because reducing a problem constitutes an improvement in the situation. This benefit will vary for each axis:

- **Horizontal axis:** Will the learner get closer to the desired state after the newly modified activity, that is, will drift decrease?
- **Vertical axis:** How much will class heterogeneity be reduced after a split into subclasses?
- **Depth axis:** How much will diagnosis entropy be reduced after the question is modified?

The system cannot know in advance what the state of the learner or the class after a modification would be. This is why I call it "predicted benefit." Actually, the benefit of an adaptation *can* be predicted if transition matrices are available. For instance, the transition matrix $M^{i,j}$ and M^{ik} can be compared to determine how much would be gained from replacing a_j by a_k. However, for a modification of type "repair," this benefit can only be guessed.

Now, the various possible modifications should not only be compared in terms of predicted benefit but also in terms of cost. Creating a new activity may have a higher benefit than re-explaining the current activity, but it represents much more work for the teacher. Every adaptation or repair has a **cost** in terms of the teacher's time, energy, and money, as well as learner time, energy, and money. The modification should be performed only if the necessity for adaptation justifies these costs. If the adaptation has been integrated in the graph as a distribution operator, the cost is minimal. If the teacher has to split the class and quickly build new activities for a subset of the class, the cost is higher. The cost is therefore more of a technical issue, related to the flexibility of the technology: if the teacher has to change a sentence in a document, the cost is much lower than if he has to edit a video.

In summary, the selection of a graph modification follows a cost/benefit model on each axis; if the parameter bypasses a certain threshold, it creates a need for evolution. If there is a modification that can bring the parameter back below the threshold and if the cost is affordable, then the adaptation or repair will be applied.

$$p(\text{modification }(G)) = \frac{\text{necessity (modification }(G)) \times \text{benefit (modification }(G))}{\text{cost (modification }(G))}$$

where G is the graph and P (modification(G)) is the probability of performing a modification.

This looks like an equation, but it is more conceptual than numerical. The equation means that for a serious problem, there are more opportunities to change the graph if there is an optimistic way of solving the problem, and if this solution is affordable. We will see that some of these elements can be measured. However, the 3 variables (necessity, benefit, costs) refer to the constraints described in the previous point, which were mostly described in qualitative terms. The intrinsic and extrinsic constraints influence both the necessity and the costs of adaptation. For instance, low participant scores (intrinsic constraint) or a copyright problem (extrinsic constraint) determine the necessity of modifying the graph.

In adaptive systems, the benefit can be predicted, but the costs are negligible, since all modifications have been predefined. The main cost is the risk of violating other constraints (time, curriculum). In repairs, the benefit cannot be predicted, and the cost can be much higher. Therefore, in the next two points, I will elaborate on the "economy of evolution" equation for adaptations (Point 37) and repairs (Point 38), respectively.

> *This "economy of evolution" equation reflects a very rational view of education. This is indeed the perspective on education that I have maintained throughout the book. It can inspire algorithms, but it does not faithfully model the reasoning of a human teacher, which is certainly more intuitive than analytical. Moreover, if this equation is intended to model the way teachers make decisions, it should be enriched with personality traits. An idealistic teacher would repair any problem, despite the costs. A perfectionist teacher would try to repair, even if the benefit is minimal. A disillusioned teacher would consider heterogeneity or failure as natural phenomena that cannot be compensated. For a lazy teacher, any cost could be too high. My goal with this fake equation is to isolate the rational elements in adaptation and repair, not to predict the behavior of teachers in a real context.*

Point 37 Graph adaptations

I will now elaborate the process of adaptation, that is, choosing among predefined activities or among predefined parameter values for a given activity. I will look at adaptation from the perspective of the three modeling axes, starting with the **horizontal axis**. A pedagogical scenario, despite having been skillfully designed, is rarely perfect. Even if this scenario has been successfully conducted 17 times, students vary, the context evolves, and accidents happen. Inescapably, minor problems (misunderstandings, delays, or difficulties) accumulate and the students, individually or collectively, drift away from the optimal path. This drift is defined as follows:

- $x^*_i(s)$ is the **desired learner state** of student s at the end of activity a_i. It can be defined as the optimal state among the set of states for a_i or as a "*reasonable expectation*" state (e.g., I expect all students to reach "*intermediate*" level, even if some may reach an "*advanced*" state). The expectation varies depending upon the context; the state "*being able to open one's parachute*" is clearly a desired state before jumping out of a plane. In many courses, the teacher would discriminate between fundamental skills, for which he expects all learners to reach full mastery, and optional skills, where a lower state of mastery would be tolerated. *In general,* $x^*_i(s)$ can be calculated as the state with the highest probability of being successful in the next activity, as predicted by the transition matrix. The best state in this next activity is itself the optimal state for the following activity, and so forth recursively till the final state, defined by the learning outcomes associated with the scenario.

- $\Delta\ (x_i(s), x^*_i(s))$, abbreviated as $\Delta x_i(s)$, is the **drift**, that is, the difference between the state in which the learner is after a_i and the state in which he was expected to be. This difference can be positive (i.e., the student's state is lower than the expected state), but it can also be negative, if the student reached a higher state than expected. Negative drifts are not a problem, but they are worth capturing. I use Δ instead of a subtraction[3] to refer to any measure of differences.
- The next element anticipates how a learner state would evolve if the next activity was modified in a specific way: $x_{j'}(s)$ is the **predicted learner state** after an activity $a_{j'}$, where $a_{j'}$ is the adapted version of a_j M. $\Delta\ (x_i(s), x_{j'}(s))$ is the **predicted benefit**, that is, the difference between the learner's current state and the state in which he could be after the modification of the graph.

In some cases, students reach the expected state much later than expected, which can be modeled (if all activities use the same set of states) as another type of drift $x_i(s) = x^*_j(s)$ but $i > j$. This **time drift** can also be negative if $i < j$. If this delay is shorter than the duration of a_i, this won't be modeled in the system, since the model only considers states at the end of an activity. Again, if time drift is a concern—actually a frequent issue in lecturing—the activity should be broken up into subactivities.

If drift passes above a certain threshold, the student or the class will eventually reach a point where activities are not appropriate anymore. Drift on the horizontal axis leads to adaptation on the same axis—changing the current or the next activity. A positive drift may trigger adaptation operators such as lowering the level of difficulty or re-explaining the prerequisites. A negative drift may also trigger adaptation; for instance, increasing the level of difficulty or skipping some learning activities.

The economy equation presented in the previous point has 3 parameters, necessity, benefit, and cost. Table 6.4 presents the values of these parameters for adaptations of the horizontal axis.

---

[3] A subtraction would only apply to metric states, while this book mostly referred to discrete states.

Table 6.4 Parameters of the economy of adaptations equation on the horizontal axis.

Parameter	Description of this parameter for the **horizontal axis**
Necessity	The necessity is the drift $\Delta x_i(s)$: the further the learner is from the desired state, the more it is necessary to change something.
Cost	The cost of adaptation is minimal, since, according to my definition of adaptation, these modifications have been predefined: the modification has to be decided, but applying it requires no effort. There can, however, be collateral costs due to the fact that the modification violates other constraints, such as the total time available.
Predicted benefit	The predicted benefit is the reduction of drift after the adaptation, if this adaptation is to replace a_j is by a_k as the next activity: $(\Delta x_i(s) - \Delta x_j(s)) - (\Delta x_i(s) - \Delta x_k(s))$

The predicted benefit can be calculated as a subtraction if the states are numerical values, such as a performance score. If modeling relies on a set of discrete states, it can be calculated in the way my following example outlines. I have looked at a simple case where the teacher has to choose between two activities that have already been used in the past, a_j and a_k, which means that two transition matrices are available, M^{ij} and M^{ik} (Figure 6.4). They have the same utopy value for the whole class (Point 24), but the data are distributed differently over rows. If a learner encountered difficulties in a_i, ($x_i(s)$=”*difficulties*”), his possible transitions are located in the second row of each matrix. The second row of M^{ij} is more pessimistic than the second row of M^{ik} in which this learner has a 40% chance of improvement ($p(x'_k(s)$=”*medium*”)=.3 + $p(x'_k(s)$=”*good*”)=.1). The teacher should therefore select a_k for a learner who has difficulties. Conversely, if the learner was fine in a_i ($x_i(s)$=”*good*”), the fourth row of M^{ij} is more optimistic than the second row of M^{ik} in which there is a probability of 30% that the learner will end up with a lower state ($p(x'_j(s)$=”*medium*”)=.3). This could happen if a_k was so easy that the best students in the class got bored. In summary, if transition matrices enable predictions, the rational decision is to choose the next activity that has the most optimistic prediction.

It could be argued that predicting the state of the learner in the next activity is not so important, since what matters is the final state, or whether the learner will reach the learning goals of the scenario. If the learner is in state $x_i(s)$, which according to the $M_{i,i+1}$ is related to $x_{i+1}(s)$ (and which is itself predictive of $x_{i+2}(s)$), the final learner state $x_{end}(s)$ can be recursively predicted and the necessity of changing the next activity can be decided on. It would—mathematically speaking—make sense to calculate predictions up to the final state, but as education is not an exact science, I am not optimistic that accumulating uncertainty over several transition matrices will produce accurate predictions.

M^{ij}	*Lost*	*Difficulties*	*Medium*	*Good*	*Brilliant*
Lost	0.8	0.2	0	0	0
Difficulties.	0.3	0.6	0.1	0	0
Medium	0	0.1	0.6	0.2	0.1
Good	0	0	0.1	0.6	0.3
Brilliant	0	0	0	0.1	0.9
			U(M)		0.04

M^{ik}	*Lost*	*Difficulties*	*Medium*	*Good*	*Brilliant*
Lost	0.6	0.3	0.1	0	0
Difficulties	0.1	0.5	0.3	0.1	0
Medium	0	0.1	0.5	0.3	0.1
Good	0	0	0.3	0.6	0.1
Brilliant	0	0	0.2	0.3	0.5
			U(M)		0.04

Figure 6.4 Predicting the future state of a learner in order to choose between two activities. $U(M)$ is the utopy index of the matrix (Point 24).

Let's now consider the **vertical dimension** of the modeling cube—the social heterogeneity. In rare cases, drift can be the same for all students; for instance, if nobody found the correct answer. However, in most cases, drift is not the same for all learners. So, drift also leads to an increase in class **heterogeneity** on the vertical axis: $\sigma\{x_1(s_i), \ldots, x_i(s_n)\}$ where n is the number of students in S and σ is any measure of dispersion on the set of individual states. If the differences among learners grow, they will eventually reach a point where what is suited for one subset of the class is no longer suited for the rest of the learners. In actual fact, a typical form of class heterogeneity is not a Gaussian distribution with a large variance, but rather a bimodal distribution. This corresponds to a frequent discourse among colleagues, such as, "*There are actually two subsets of students in my class; those who have no problem at all and those who struggle to pass.*" In many other cases, colleagues report a trimodal distribution; "*Those who don't need me, those who have no chance, and, in the middle, those for whom I work.*" The most obvious **adaptation** is to split the class into two or three subclasses and give them parallel independent activities, as explained in Point 8. For instance, in a language class, the adaptation could be to split the class into native and non-native speakers, which enables activities that suit the needs of both levels.

The economy equation presented in the previous point has 3 parameters, necessity, benefit, and cost. Table 6.6 presents the values of these parameters for adaptations of the vertical axis.

Now, reducing heterogeneity is not the only choice. In the language class example, the graph can form pairs of students—a native speaker with a non-native speaker—so that the one who is fluent will help the one who faces difficulties. Therefore, the social operators that adapt the graph are not only able to minimize heterogeneity, but can also optimize it, that is, find the level

Table 6.6 Parameters of the economy of adaptations equation on the vertical axis.

Parameter	Description of this parameter for the **vertical axis**
Necessity	The necessity is the dispersion of the class or the distance between modes in a bimodal or trimodal distribution.
Cost	This adaptation has a low cost if the graph includes subclass activities, and if there is a distribution operator (Point 15) that partitions class S into $\{S_i\}$ in such a way that reduces the heterogeneity of individual states within each S_i.
Predicted benefit	The predicted benefit is the difference in variance between the whole class and the split classes.

of heterogeneity that produces the richest interactions. I stress the words "optimize heterogeneity" instead of "maximize" because, beyond a certain threshold, heterogeneity may lead to the emergence of negative phenomena such as the free rider effect (Point 21).

Finally, adaptation also occurs in the **depth axis** of the cube. Table 6.5 presents the values of the 3 equation parameters for adaptations of the depth axis.

Table 6.5 Parameters of the economy of adaptations equation on the depth axis.

Parameter	Description of this parameter for the **depth axis**
Necessity	The necessity for adaptation is the uncertainty regarding the current state of the learner(s). If the teacher or the system has too much **uncertainty** to make a decision, he/it has to modify the diagnostic process; namely, by asking an additional question or rephrasing an existing question. In Point 31, the uncertainty was calculated as the normalized entropy of the vector made from all possible learner states. This applies both for an individual learner and for a class of learners. Now, what is the acceptable level of entropy between 0 and 1? This question cannot be answered in general terms, but only empirically. There are activities where entropy can be tolerated, because whatever the state of the learner is in a_i, it will not greatly affect a_j, according to M^{ij}.
Cost	The cost of adaptation is low if the adaptive system has a library of additional questions or tasks available, or if the system includes disambiguation mechanisms as in some computational models of dialogue (Traum, 1999).
Predicted benefit	The benefit is the diagnosis power of the new question (Point 31); for instance, "*Is this clear?*" does less to reduce uncertainty than "*Which of these examples contradict the definition?*" The benefit is therefore the decrease in uncertainty provided by an additional diagnostic act, that is, the difference in entropy after the adaptation, if this adaptation is to replace a_j by a_k as the next activity: $(H(X_i(S)) - H(X_j(S))) - (H(X_i(S)) - H(X_k(S)))$

Point 38 Graph repairs

Repairs are the modifications of the graph that were not anticipated at design time, but are performed on the fly by the teacher. Let's again consider the three variables of the economy equation, the cost/benefit ratio. The necessity to repair the graph will be the same as for adaptations, and this is the case for each of the 3 cube axes. The benefit that can be expected from a repair cannot be predicted, since the modified activity, by definition, is new and therefore no transition matrix is available. Intuitively though, an experienced teacher could anticipate some benefits. The key variable in the "economy of repair" is therefore the cost of a modification. The cost of a repair is determined by the flexibility of a graph, that is, the amount of effort necessary to modify it. This implies some pedagogical and technical flexibility.

- The pedagogical flexibility is determined by the intrinsic constraints; if an activity is changed, how will this increase or reduce the probability that students learn what they have to learn? The weight of an edge indicates how necessary one activity is for the next one: skipping a_i will be less detrimental if ω_{ij} is low. For instance, skipping a prerequisite activity is more risky than skipping a motivation activity. The cost can also be estimated by the extent to which extrinsic constraints are violated; for instance, increasing the duration of an activity can either reduce the time devoted to another activity or modify the total time budget of the scenario (what I called "relaxing the constraint").
- The technical flexibility of the graph refers to the energy required for changing it; even if the activity could be modified from a pedagogical viewpoint, it is sometimes difficult to change the technology. Some technical choices in the construction of a workflow can be antagonist to flexibility—workflows were invented for automating processes. Since operators transform data structures into other data structures, they can't be permuted randomly without a risk of breaking data consistency. If an aggregation operator collects the products of four learners, A, B, C and D, in order to compute A+B/C+D, what happens if D is missing? It is a challenge for computer scientists to develop operators with a level of abstraction sufficient for substantial flexibility.

Figure 6.5 illustrates how we increased the technical flexibility of a workflow operator, within an online CSCL environment called ManyScripts (Dillenbourg & Hong, 2008). This feature was part of the graph ConceptGrid (Point 19). The window shown in Figure 6.5 enables manual repair (by the teacher) of team composition. For instance, if 16 students have to form teams of 3, what should we do with the 16th student? What if a student drops out in the middle of the teamwork? ManyScripts allows the teacher to cope with these constraints with the buttons on the right hand side of the window in

Figure 6.5. If there are fewer students than roles, ManyScripts suggests the "spy" feature—if role 3 is missing in Team 33, this team may borrow the definitions produced by any student who is playing role 3 in another team. If there are more students than roles, ManyScripts allows the teacher to choose the "joker" feature—the extra student(s) will have the right to choose any of the already distributed roles in his team. If the team has 9 concepts to define, but includes 4 members instead of 3, they can freely decide how to distribute the workload among themselves. These solutions are not perfect, as because workload is somewhat uneven within or between groups, this may violate the fairness constraint. However, they enable the teacher to continue the scenario despite unexpected events (e.g., a student dropout). This is an example of the teacher's bypass function presented in the previous point.

On the diagnosis axis, the cost of a repair is the time spent by the teacher or the students for collecting the additional behavioral information required to reduce uncertainty. "*Is this clear?*" is often used despite its low benefit, because it takes 2 seconds, while "*Write a summary*" provides a fine diagnosis, but takes time both for the student and for the teacher who grades it. An interesting way

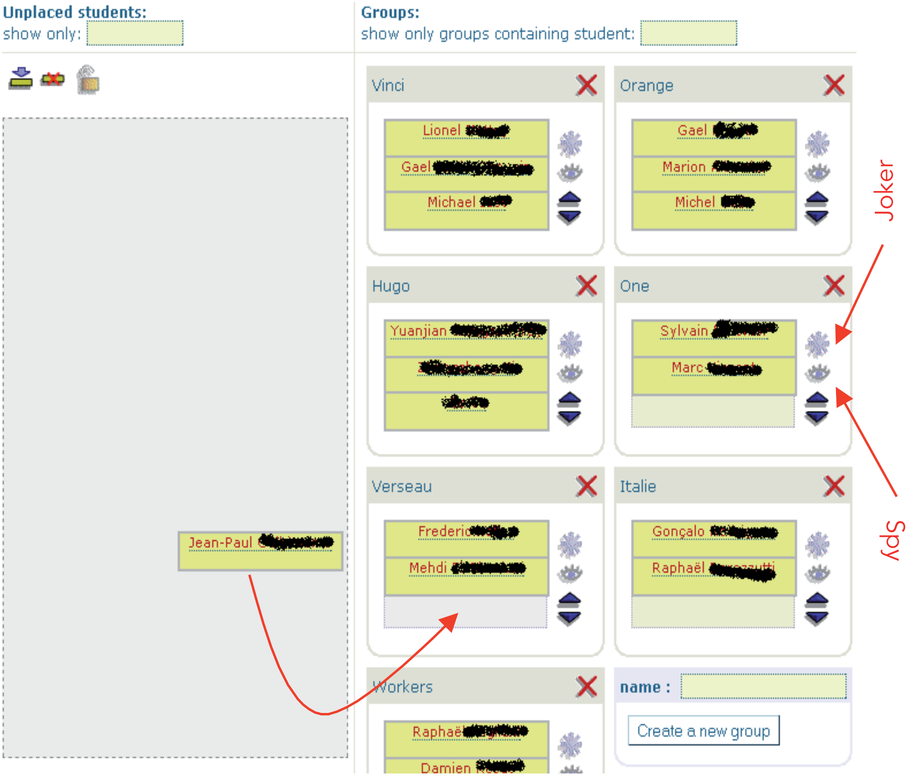


Figure 6.5 Examples of flexibility tricks for group formation in ManyScripts.

to implement diagnosis repair is the so-called **open learner model** (Bull & Kay, 2010), that is, the possibility for learners to access the information the system inferred from their behavior and to repair it—to modify it—if they consider it wrong. Moreover, inviting students to control how their own cognitive state is modeled by the system constitutes an opportunity for reflective activities. This self-repair is an interesting activity for the development of metacognitive skills. Actually, in proper communication settings, humans are quite good at detecting if they have been misunderstood and in repairing the conversation if that is the case. This phenomenon has been investigated in psycholinguistics, where models of mutual understanding discriminate two ways of repairing misunderstandings[4] (Clark, 1996). If A has too much uncertainty of what B meant, A asks a clarification question. Conversely, B may repair dialogue by rephrasing what he said. This repair requires B to build a representation of how A understood what he meant. This may sound sophisticated, but I have heard learners commenting on their interactions with the computer: "*It thinks that I believe this, but this is not what I believe.*" The possibility of detecting and repairing misunderstandings is related to the features of the communication medium; Clark & Brennan (1991) analyzed several medium features that influence the cost of grounding. One key feature is that the people who communicate with each other have a shared workspace that supports deictic gestures; the learner points to an object with a gesture or a mouse cursor to disambiguate what he meant by "*here*" or "*it*." This refers to the first key finding in CSCL: the critical design factor for an interface that supports collaboration is less the epistemic truth of the graphical representations than the ability for learners to refer to interface objects in order to detect and repair misunderstandings (Roschelle, 1992).

Point 39 Self-improving systems

As defined in Table 6.1, I call 'optimization' the process by which systems improve their effectiveness or their teaching resources. In the past, I developed and tested a self-improving system (Dillenbourg, 1989). The system taught geometrical concepts by systematically varying the learning activities—10 activities being available for teaching each concept. It recorded the student's profile before each activity and immediately measured the activity's effectiveness by testing the concept to be learned. The system used machine-learning algorithms to discover which student profile was predictive of the effectiveness of each method. At that time, symbolic machine-learning methods were rudimentary, and the sample size was too low to produce any robust results. This research direction faded out rapidly. However, three decades later, machine

---

[4] For instance, Naël indicates to Savita that he lives in Geneva, and Savita confuses Geneva with Genoa, in Italy. When she replies, "I love Italy," Naël understands that Savita misunderstood him. Naël has naturally built a representation of how Savita understood what he said. This triggers the dialogue repair.

learning has become very powerful, and MOOCs provide wonderful sample sizes. I expect this topic to receive renewed attention in the near future. The term "self-improving system" is, however, a bit misleading. Machine-learning algorithms have to be run many times, manually trying various parameters and thresholds, before getting interesting results. Moreover, these interesting results will have to be filtered by human judgment. In other words, the machine does not really operate the improvement process in an autonomous way.

The modeling language I proposed for orchestration graphs supports this optimisation, since the transition matrices store the information collected from every activity with every learner. As data accumulate in the transition matrices, the prediction based on these matrices will become more accurate, which should enhance the efficiency of the digital learning environment in a similar way to how a teacher accumulates experience.

My goal in formalizing pedagogical scenarios is to even go beyond this accumulation of experience. Evolutive systems should discover new states as clusters of learner states, and refine the decision process with new rules that associate a state with a learning activity. The formal description of states and transitions aims to connect education with machine learning. Supervised machine-learning algorithms may discover data patterns where the final output can be classified as being positive or negative. This is the case with orchestration graphs, for instance by dichotomizing post-test scores into "pass" or "fail."

Now, machine-learning methods find relationships between variables, but do not establish any causality between them. Correlation is not causality. Analytics won't contribute to educational theories (which require causal processes) without being combined with controlled experiments. The formalism of graphs could allow us to precisely describe the nature of the intervention (which elements of the graph are being manipulated) and to include experimental comparisons in the graph.

The basic idea behind the experimental methods used in educational research is to compare the learning gains of students who followed method A versus those of students using method B. This comparison can be integrated into the graph (Figure 6.6 and Figure 6.7). If A is the new method, it's called the "treatment" and the subjects who follow it are called the "experimental condition," while the group following the standard method is called the "control condition." Figure 6.6 and Figure 6.7 present two typical experimental designs. In both designs, students start with a pretest (a_1), and subjects are assigned to the experimental group or to the control group. The edge operator may assign them randomly to one of the two groups or structure the assignment with "stratified sampling" based on pretest results. For instance, subjects may be classified as "*good*," "*medium*," and "*low*," and the sampling method makes sure the ratio of each category is the same in the experimental group as in the control group (see distribution operators, Point 15). It is also possible to include a questionnaire in the pretest and to balance the two groups with respect to gender, age, background, and so on.

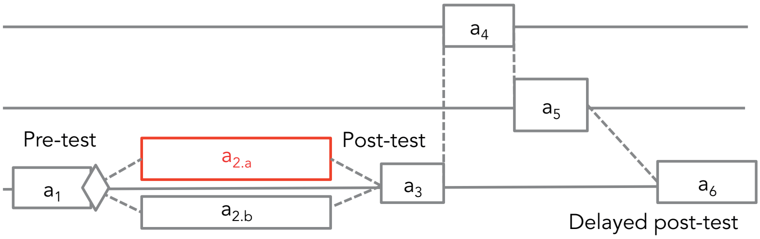


Figure 6.6 "Between subjects" experimental design.

In the "between subjects" experiment design (Figure 6.6), subjects from the control and the experimental groups perform different activities, respectively, $a_{2.1}$ and $a_{2.2}$. The differences between $a_{2.1}$ and $a_{2.2}$ are the "independent variables" or factors. Their effect on learners is measured through a post-test (a_3). The variables differentiating performance at this test are called "dependent variables." A delayed post-test is sometimes used to see how much is retained 1 hour, 1 week, or 6 months later.

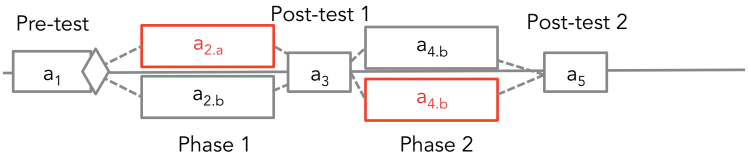


Figure 6.7 "Within subjects" experimental design.

The "within subjects" experiment design (Figure 6.7) follows the same approach for the first phase as the previous one, but students switch activities for the second phase, usually to task (a_4), which is as similar as possible to the first task (a_2). All students pass through both conditions, which implies that the differences, in effect, between methods A and B are not due to the differences that might emerge between two subsets of the class, despite random sampling. This design requires fewer subjects and is ethically more acceptable; if our hypothesis predicts that method A is more effective than method B, it's unfair to assign students to method B. This method guarantees that all subjects benefit from the most effective method, A. Half of the class does A before B and B before A in such a way to neutralize the order effect, but this often generates results that are complex to interpret. This within-subjects approach is very hard to apply when the experiment plan includes multiple factors or independent variables.

Point 40 Participatory graphs

This point develops a special case of self-improving systems in which the evolution does not result from a learning algorithm system, but instead comes from the students' contributions to the contents of the course or the MOOC, a socio-constructivist idea close to cMOOCs. MOOCs provide spectacular ways of collecting information in a structured and scalable way. The modeling language orchestrates this participation process. Graph operators enable the collecting of examples, pictures, or cases studies, which capture what the employees of a company think or the citizens of a country want. I illustrate participation mechanisms with a MOOC on geology (Figure 6.8).

After the introductory lecture (a_1), the 10,000 students are asked in a_2 to upload 2 pictures of erosion that they have taken in their surroundings. They also have to enter the coordinates of the place where the photo has been taken. Operator 1 collects the pictures, classifies them by location, and removes all pictures that are of bad quality, too fuzzy, or that contain human faces, pornographic elements, and so on. From the 20,000 collected pictures, let's assume that 15,000 are usable. Operator 2 randomly selects two pictures that have been taken in different part of the worlds and gives them to pairs of learners. They have to classify the pictures as examples of "geological erosion" versus "accelerated erosion." They also have to rate the quality of the image. Operator 3 aggregates pictures into 3 categories: those for which there is over 90% agreement that the picture shows geological erosion, those with the same unanimity for accelerated

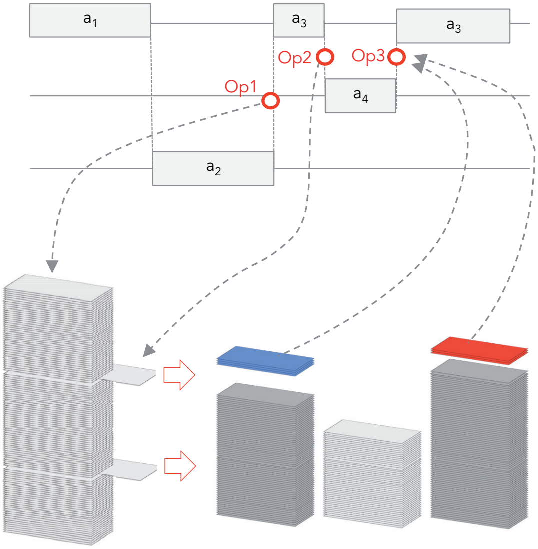


Figure 6.8 Collecting and filtering new teaching material.

erosion, and those that led to disagreement. In the newly formed subsets, each of which includes maybe 5,000 pictures, the third operator keeps the 10% of pictures with the highest quality ratings. The teacher finally obtains a database of 1,000 high quality pictures. He can also further analyze the examples that have been rejected due to a lack of consensus, since their analysis might reveal why students have difficulties discriminating between the two concepts.

This example has a pedagogical goal, but the process can go further. Imagine that the participants are not worldwide learners, but the employees of the company. The participants of a MOOC on management have to upload examples of suboptimal processes in the company. As the end, beyond the value in the course itself, it could be very worthwhile for the company management team to analyze these examples. MOOCs can become both a training tool and an information-collecting tool, provided that individual privacy is treated carefully. This participatory approach may appear slightly idealistic. Actually, from talking to many chief learning officers from large European corporations, I have noticed their considerable interest in these forms of training, where top-down delivery of contents is enriched with bottom-up collection of information. In addition, managers appreciate the benefit their company may expect, if its employees, sales networks, or even customers feel empowered as knowledge contributors. In this approach, similar to that of cMOOCs, training is no longer treated as an activity different to work; it removes the boundary between training and working. Orchestration graphs implement these participatory scenarios, making them more manageable.

If we extend the same data collection idea to a local, regional, or national community of citizens, these educational graphs could lead to new forms of democracy being invented. An intensive argumentation cannot occur in a forum with 30,000 participants. The easiest way to scale up is to simplify interactions; for example, voting, where citizens can only choose among existing proposals; they cannot propose new ideas unless they engage in politics. These graphs could support methods by which 200,000 citizens jointly build a proposal in the same way they processed the geology pictures.

Point 41 Orchestration load

Orchestration load is the sum of two sources, workload and cognitive load. The workload is the amount of energy necessary for running the scenario, monitoring the state of all learners, and adapting and repairing the graph. The teacher's cognitive load refers to the quantity of information the teacher needs to process and temporarily store; for instance, in order to think carefully about what he will say (e.g., lecturing on a complex topic), or to infer the state of the learners plus the costs of adaptations and repairs, which have been described in the previous point.

> *The concept of cognitive load is simple and intuitive; our working memory can only store a limited number of pieces of information at the same time. If humans were computers, we could say that our hard drive is unlimited, while our RAM is limited to a few elements. I mentioned in Point 35 my criticisms against scholars who use cognitive overload as an argument for pedagogical fast food. Simply stated, learning is the side effect of processing information, and any information processing inevitably implies some cognitive load. Writing the summary of a lecture is a more intense processing of information than listening to the lecture, hence a higher cognitive load, but it should lead to higher learning outcomes. Now, I consider the concept of cognitive load useful for describing the teacher's activities and understanding the acceptance of technologies in classrooms.*

Let's consider a teacher who is in front a class of 30 pupils. His load includes reasoning on the contents, following the graph activities, managing time constraints, managing discipline, monitoring the state of the students, adapting and repairing the graph, and so on. If any slot remains available in his limited working memory buffer, he might add some instances of humor. Let's consider a single task among these many parallel tasks: "*Monitoring the state of the students.*" If every learner state is defined by 3 variables such as "*attention: high*," "*motivation: low*," "*understanding: low*," this means that the teacher needs to have 90 information pieces permanently available to follow his whole class. This is far from what empirical studies have established as the maximal capacity of the working memory: 7 ±2 units. These two values, 9 versus 90 pieces of information, are actually not incompatible because the "unit" of the working memory is undefined. For an expert, a unit can be a rich structure such as a disease, its mechanisms, its symptoms, and its treatment. Experienced teachers probably develop similar rich structures that allow them to condense information. I will risk an analogy with methods for compressing images: how can we reduce the file size of an image without a major loss of quality? Here are three answers to this question:

- *Depth reduction:* The file size of an image can be reduced by minimizing the amount of information per pixel, or pixel depth: one bit of information allows 2 colors, 2 bits 4 colors, n bits 2^n colors. To minimize orchestration load, the multiple features used for describing a learner can be summarized by a single pattern (e.g., "*smart-lazy*," "*meticulous-but-limited*,"…). In large data sets, such as those generated by MOOCs, these patterns can be extracted from multiple variables by using techniques such as principal component analysis (PCA). In image compression, such a method is called "lossy" since some data are lost and the final image is less accurate. However, it may be that the difference in quality is not noticeable by the eye looking at the picture. Similarly, in the classroom, the teacher using this kind of pattern could lose details about individual differences, but this would have no impact on the next decision to be made.

- *Spatial compression*: If all pixels within an $n \times m$ pixel-wide area of a picture are the same color, the information about the $n \cdot m$ pixels can be replaced with the information about one pixel and the parameters that define the area. To minimize orchestration load, a teacher who is in front of a whole lecture theatre; for instance, could also do spatial compression by replacing n students by "*the last two rows*." Other categories of students could be used to reduce information about the class, such as the chemists versus the biologists, the first year students versus the second year ones, the boys versus the girls, or the native speakers versus the others.
- *Time compression:* In a movie, if two successive images are very similar, the system may store only the first one and then the differences between the first and the second one. This may apply to n images. To minimize orchestration load, the teacher may only keep the relevant changes of state in the classroom in his working memory.

Every human task requires some cognitive load, teaching being one of them. This is not the point. My point is that orchestration load is a key criterion to understand why technologies are accepted or not by teachers. For instance, while lecturing on a tablet whiteboard, it can be very demanding to use all the options (changing tools, pen color, pen size, ...), while concentrating on the discourse. If a teacher uses two interactive tables in his classroom, each with 5 learners, he may still have 15 learners left to manage while waiting. He will eventually find a way to orchestrate this class, but this requires an extra effort. I hypothesize that orchestration load partly explains the low rate of acceptance for many learning technologies. Despite 30 years of rich development, these technologies are still mostly underexploited in schools. Many scholars claim that this slow development is due to teacher's traditional resistance to new things. But why would a teacher not use the technology they use in their everyday life in their classroom? I believe instead that the design of these technologies often increases their orchestration load, to a point where the technology is simply too painful to use.

The corollary is that the design of a learning technology should strive for minimizing the teacher's orchestration load. I illustrate this with a device called Lantern (Alavi et al., 2009). We observed that, in our university, exercise sessions are often orchestrated in a suboptimal way. Students typically work in teams of 2 or 3, receive a list of exercises, start to do these exercises up to the moment where they need help from a teaching assistant (TA), and then raise their hand. While students are waiting for the TA, many don't continue searching for a solution. Instead, for 62% of their waiting time, they visually track the TA to make sure they grab him as soon as he is available. Therefore, we developed a very simple device (Figure 6.9) that each team puts on its desk. The color of the device indicates which exercise the team is working on. When a team moves to the next exercise, it rotates the cap of the Lantern and the color is updated. At a glance, the TA knows the position of each team within the series of exercises. Once a team starts a new exercise, only the lowest LED of the Lantern is on, but as they spend more time, the higher the LEDs will turn on. This allows the TA to let teams try to solve the problem by themselves before offering help. When a team needs help, it pushes on the Lantern top, which starts blinking, first slowly and then faster and faster. Alavi tested the Lantern in classrooms and observed that the time wasted while waiting for the TA, decreased from 62% to 6%.

The Lantern clearly illustrates the notion of orchestration technology; the tool does not change the learning activity, per se, as it remains the same, but instead optimizes the orchestration of the activity. In terms of orchestration load, the Lantern acted as an information buffer. Without a Lantern, a TA sees hands rising here and there, and he has to remember who was first (which he often forgets), who is late in the sequence of exercises, and so on. The Lantern actually gathers this information for each team, and by glancing over the classroom, the TA can perceive the state of the whole class. The Lantern can be described as an orchestration prosthesis, an extension of the TA's working

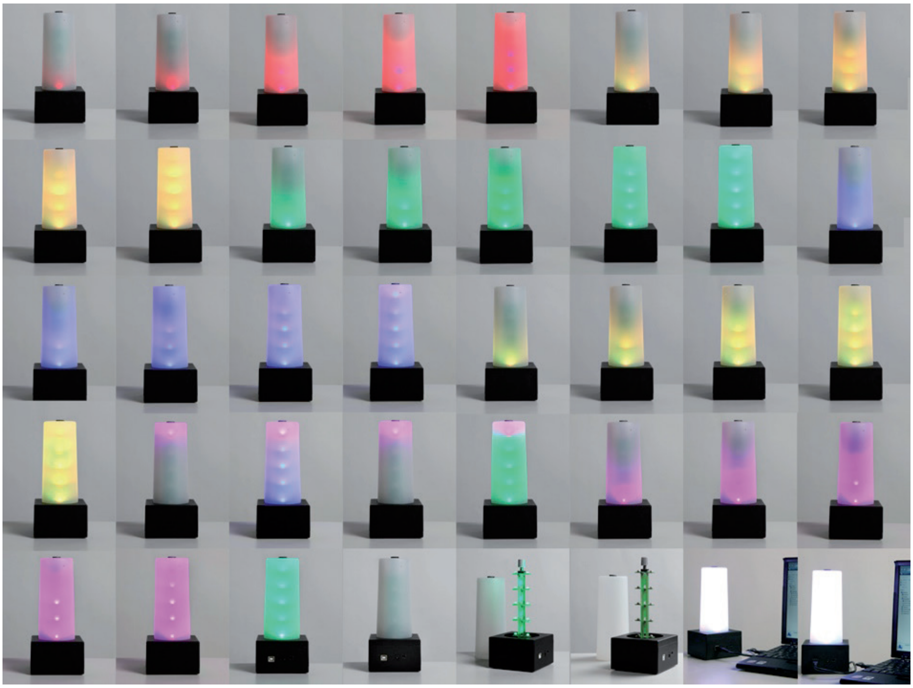


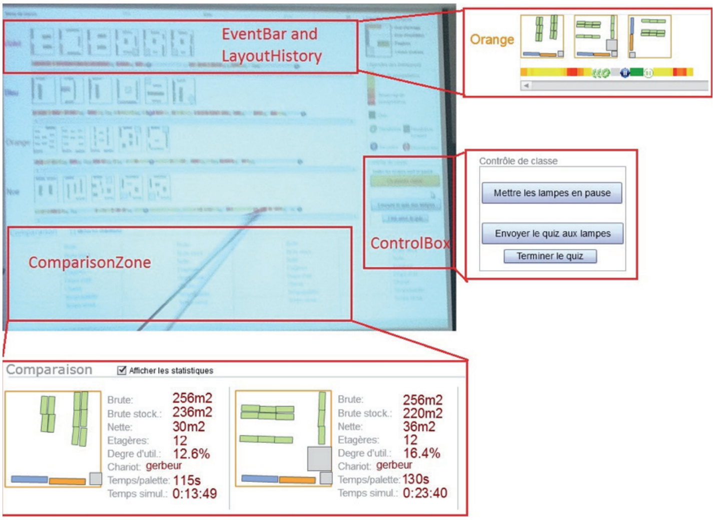
Figure 6.9 Various states of the Lantern. A color state corresponds to a learner cognitive state at π_2 (usually it corresponds to a team of 2 to 4 students). The Lantern displays 4 pieces of information that describe the team's state: the exercise they are currently working on, the time spent on this exercise, if they asked for help, and how long ago.

memory embedded into a physical device. Another orchestration prosthesis was developed by Raca & Dillenbourg (2013): the system uses computer vision methods to measure the attention of all learners in the class and provide the teacher with a synthetic account of their attention. As would many prostheses, an attention meter will probably be useless for an experienced teacher, but useful for those with lower skills, which could be progressively internalized.

The Lantern can be viewed as a dashboard that has been distributed over space. A dashboard is to teachers what a cockpit is to aircraft pilots: it summarizes all the necessary information so that the orchestrator, teacher, or pilot, can make decisions quickly. Several colleagues have developed dashboards for online education. Producing visualizations from analytics is a great domain for research. The same approach applies to physical classrooms.

Do Lenh et al. (2010) developed a real time orchestration tool for teachers in a vocational school. The learning activity was conducted with an augmented-reality environment. The students were logistics apprentices, studying how to optimize storage and the transportation of goods in a warehouse. Apprentices used an interactive lamp, called TinkerLamp (Figure 6.11), developed by Zufferey et al. (2009). They built a warehouse through a tangible interface, namely placing plastic shelves on the table. The camera situated in the lamp captured the position of the shelves, via two mirrors, and the beamer overlaid information (the simulation of the warehouse); the learner could then see forklifts operating between the shelves and the trucks that came in and out the warehouse. Typically, four TinkerLamps were used in the classroom, which made the orchestration pretty difficult. Therefore, Do Lenh developed a dashboard (Figure 6.10) in order to facilitate orchestration. The dashboard displayed the warehouse created by each team, as well as various parameters that summarized the performance of these warehouses. The teacher could use the dashboard to automatically compare the performance of two warehouses—comparisons that are useful for debriefing lectures, but difficult to calculate without a tool. This dashboard therefore included an aggregation operator (Point 13). The classroom workflow connected the four TinkerLamps and the teacher projection device.

Actually, a dashboard or any other orchestration tool provides more information to the teacher and invites him to perform more operations. Does it really offload the teacher or does it instead increase his orchestration load? The field of information visualization is a great space for creativity, but I must say the usefulness of the visualization is often neglected in favor of its sophistication or aesthetics. How can analytics visualization really facilitate orchestration? I illustrate the notion of relevant visualization with the TinkerLamp example. Through several empirical studies, Do Lenh et al. found that the warehouse simulation was almost too engaging: students built a specific warehouse, ran the simulation, changed the layout, ran the simulation again, and so on. Running many experiments does not generate learning outcomes without some kind of reflection activity, that is, a moment where students try to understand why one warehouse layout worked better than another. Therefore,



Brute: 256m2
Brute stock.: 236m2
Nette: 30m2
Etagères: 12
Degre d'util.: 12.6%
Chariot: gerbeur
Temps/palette: 115s
Temps simul.: 0:13:49

Brute: 256m2
Brute stock.: 220m2
Nette: 36m2
Etagères: 12
Degre d'util.: 16.4%
Chariot: gerbeur
Temps/palette: 130s
Temps simul.: 0:23:40

Figure 6.10 An orchestration dashboard for the TinkerLamp scenarios.

knowing how many changes students performed on their warehouse and how often they ran the simulation was precious information for the teacher. This information was represented on the dashboard by the yellow to red color scale (Figure 6.10). It was directly relevant for any important orchestration act, engaging students into reflective activities (e.g., explaining, comparing, predicting). Actually, classroom response systems (or clickers) also constitute good examples of simple, but useful, dashboard systems.

Point 42 Classroom usability

Usability is a key concept in human-computer interaction. It describes how easy it is to learn and to use a technology. Usability is not simply about designing nice looking interfaces or increasing user satisfaction. It also aims to understand why some designs trigger, for instance, many user mistakes, while other designs quickly make the user effective at his task. A piece of software that generates a high cognitive load for the user will be difficult to learn, and will generate stress and mistakes, which will eventually lead to low acceptance. The analogy with orchestration load is obvious. A piece of learning technology that generates a high orchestration load will also generate stress and errors, and end up with low acceptance. This analogy prompted the notion of **classroom usability** (Dillenbourg et al., 2012); it is a measure of the extent by which a learning technology facilitates the orchestration of pedagogical scenarios by a teacher. For instance, we noticed that paper-based interfaces made pedagogical scenarios easy to orchestrate, for reasons that are explained below.

Let's consider tablet computers. For the learner who interacts with a tablet, this device has a great usability; it's easy to learn and to use, providing that no long text entry is required. Conversely, for the teacher, the same tablets make orchestration more difficult. If the learners are using the devices, and the teacher wants their attention before giving a short collective explanation, it may take up to 2 minutes to have all the students "with him." If this episode occurs 5 times during a 50-minute-long lesson, it represents 20% of the learning time. In other words, the same technology may have high usability from an individual viewpoint, but a low usability from the classroom perspective. This explains why we distinguish three planes of usability (Dillenbourg et al., 2011): the individual, the team, and the classroom. They match the three lower planes of our orchestration graphs. We defined classroom usability as the third plane of usability (π_3).

These three planes of usability differ in terms of who is considered as "the user": it can be an individual person at π_1, a team at π_2, and the classroom at π_3. Each plane can also be defined in terms of constraints.

- At π_1, the constraints are the individual's cognitive load, his background knowledge, experience, or motivation.
- At π_2, the constraints are related to the team's need to build enough shared understanding to carry out the task at hand, the peers' degree of interdependence, or the possibility or not of having synchronous interactions.
- At π_3, teachers have to cope with the constraints listed in Point 35. The classroom usability of a learning environment is inversely proportional to the orchestration load it generates.

I illustrate this third plane of usability with three examples. The first picture (Figure 6.11) describes a classroom where 23 apprentices are working on four TinkerLamps (see Point 41). One peculiarity of this picture is that each lamp has a different color. Why would the color of a computer have any role

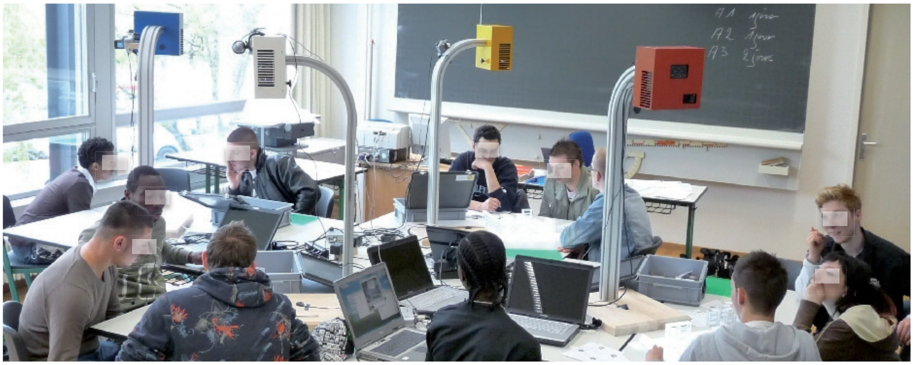


Figure 6.11 Why does the color of computers matter?

beyond marketing and sales? At π_1, the color of the device has no influence on individual interactions. At π_2, the color is no more relevant for teams interacting with the TinkerLamp. It is only at π_3 that the color plays a role; it allows the teacher to easily refer to any team during the lesson, by mentioning the color of its device. This helps the teacher to manage the learning processes, to intervene rapidly, and to control his class (which correspond to extrinsic constraints, such as minimizing teacher's workload and managing time). This is not a significant breakthrough in design, only a simple example of what is meant by the third plane of usability.

The second picture (Figure 6.12) compares two designs of the TinkerLamp hardware. Functionally, both designs are equivalent: they include a beamer, a camera, some mirrors and a computer that runs the same software. The learning activities are the same at the planes π_1 and π_2, but a difference appears at π_3, in terms of orchestrating the class. In a classroom made up of 18-year-old apprentices, the left-hand-side model violates an extrinsic constraint, maintaining discipline, because it interrupts the teacher's line of sight. A good teacher has to visually scan the class on a regular basis to monitor learner states, and this is difficult if some apprentices are hidden behind the lamp. Conversely, in Swiss elementary classrooms, there is usually a small corner, with a book shelf, a sofa, and a computer, where from time to time the teacher can send one or two pupils to carry out individual exercises or teamwork on the computer. In this context, where discipline is less of an issue, the black model creates an intimacy that allows the pupils to concentrate on a task distinct from the activities of the rest of the class. In other words, the shape of this device mainly affects the third plane of usability.

The third example is also related to the TinkerLamp. As I mentioned earlier, the apprentices tended to manipulate the simulation too often, without much

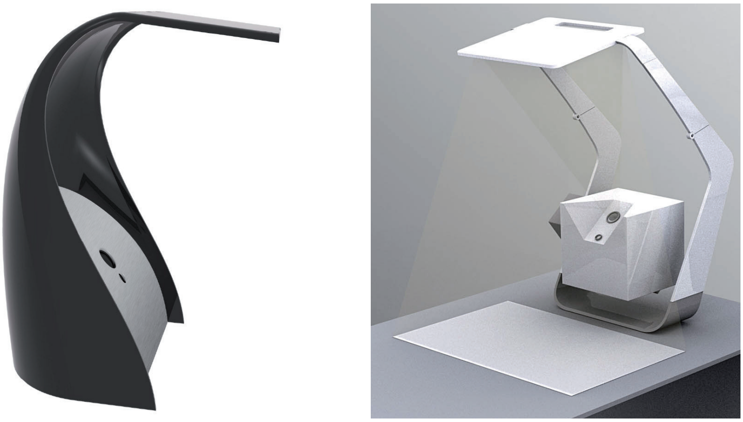


Figure 6.12 Two designs of TinkerLamp that differ at the 3rd plane of usability. Left lamp design by D'Esposito & Gaillard; right lamp design by Y. Guibinelli.

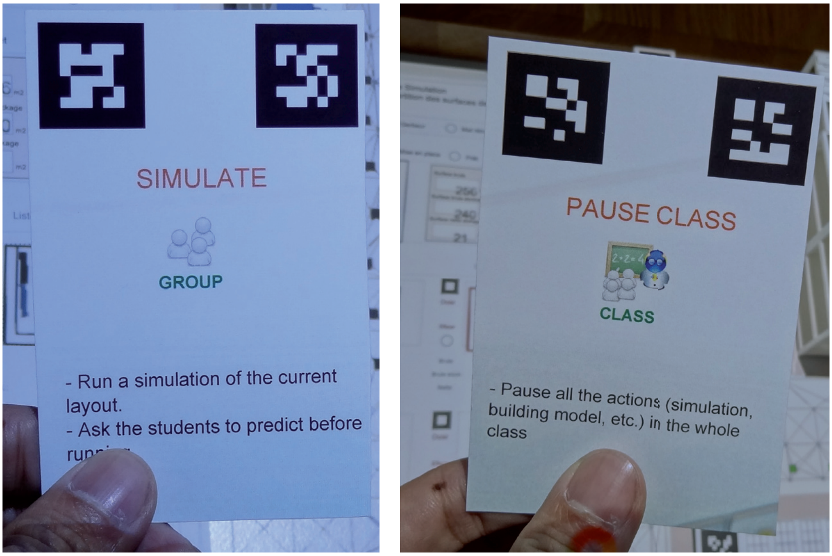

SIMULATE
GROUP
- Run a simulation of the current layout.
- Ask the students to predict before run
PAUSE CLASS
CLASS
- Pause all the actions (simulation, building model, etc.) in the whole class

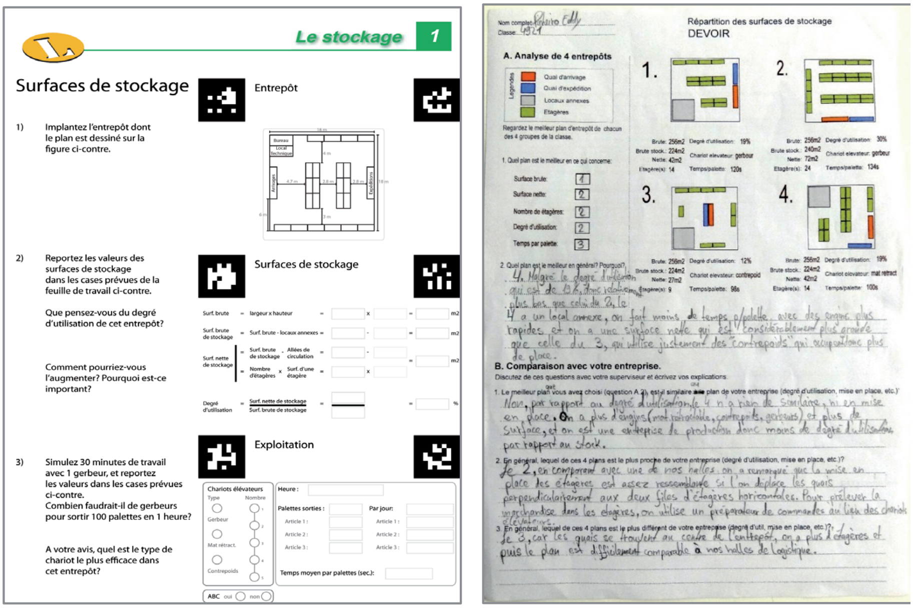
Figure 6.13 Two paper orchestration cards for the TinkerLamp: π_2 on the left and π_3 on the right.

reflection. Apprentices did not spontaneously analyze the performance of their warehouse layouts. The role of the teacher was to engage them into reflective activities. In terms of orchestration, this implied monitoring the progress of every team and jumping into their work when relevant for prompting reflection. Therefore, Do Lenh developed orchestration cards (Figure 6.13). The left one has two sides: when the teacher shows one side to a TinkerLamp, the team cannot run the simulation anymore. They may build a new layout, but, when they are ready to run the simulation, they have to call the teacher. He will then prompt reflection by asking team members to justify their design and to qualitatively predict the results of the next simulation (e.g., "*Will this warehouse operate faster or slower than the previous one?*"). Once the teacher is satisfied with the team's answer, he shows the other side of the orchestration card to the TinkerLamp, and the apprentices can then run the simulation. The right card in Figure 6.13 allows the teacher to pause all TinkerLamps; when the teacher shows this card to any of the four TinkerLamps in the classroom, none of the lamps project information, just white light. This "pause" provides the teacher with the attention he needs for a short, collective explanation. Teachers appreciate the ease of use of orchestration cards; they simply walk across the classroom with 5 to 10 cards in their hands.

Moreover, to launch and run a particular pedagogical scenario on the TinkerLamps, teachers simply had to select from a binder the A4 paper sheet that corresponded to the scenario (left side of Figure 6.14) and place the right half of that sheet below the TinkerLamp. The left part of the orchestration sheet includes instructions for human users (teachers and learners), while the right part serves as interaction area; the user selects choices by moving a token on the paper "buttons," the system displays parameters that describe the layout being constructed below the lamp. Distributing a sheet of paper to each student corresponds to a lower orchestration load than the traditional process of login, followed by navigating along a sequence of menus and submenus.

The right side of Figure 6.14 shows homework sheets; when the apprentices had designed several warehouse layouts, they were asked to save their best designs. The system then prepared and printed a homework sheet that asked them to compare the warehouse they designed at school with the warehouse where they work the rest of the week. They were also invited to discuss this comparison with their workplace supervisor. Teachers warned us that apprentices never usually do their homework. As it turned out, 90% of them filled in their homework, and 82% discussed it with their workplace supervisor. This success can be explained by the novelty effect, but my hypothesis again refers to the minimal orchestration load of paper; it took one minute for apprentices during a coffee pause to take the homework sheet out of their pocket and show it to their work supervisor. In comparison, if the same activity had been computer-based, the apprentice would have had to ask his supervisor to sit together in front of a computer, to log in, to access the right document… an eternity compared to the paper version.

The positive experience with paper-based interfaces can be understood in the light of orchestration. If, in schools, paper has resisted the digital push


Bureau
Local Technique
Arrivages
Expéditions
18 m
4 m
4.7 m
2.8 m
2.8 m
18 m
6 m
3 m
1.
2.
3.
4.

Le stockage 1

Surfaces de stockage

1) Implantez l'entrepôt dont le plan est dessiné sur la figure ci-contre.

Entrepôt

2) Reportez les valeurs des surfaces de stockage dans les cases prévues de la feuille de travail ci-contre.

Que pensez-vous du degré d'utilisation de cet entrepôt?

Comment pourriez-vous l'augmenter? Pourquoi est-ce important?

Surfaces de stockage

Surf. brute = largeur x hauteur = ☐ x ☐ = ☐ m2

Surf. brute de stockage = Surf. brute - locaux annexes = ☐ - ☐ = ☐ m2

Surf. nette de stockage = Surf. brute de stockage - Allées de circulation = ☐ - ☐

= Nombre d'étagères x Surf. d'une étagère = ☐ x ☐ = ☐ m2

Degré d'utilisation = Surf. nette de stockage / Surf. brute de stockage = ☐ / ☐ = ☐ %

3) Simulez 30 minutes de travail avec 1 gerbeur, et reportez les valeurs dans les cases prévues ci-contre.
Combien faudrait-il de gerbeurs pour sortir 100 palettes en 1 heure?

A votre avis, quel est le type de chariot le plus efficace dans cet entrepôt?

Exploitation

Chariots élévateurs

Type	Nombre
○ Gerbeur	○ 1
	○ 2
○ Mat rétract.	○ 3
	○ 4
○ Contrepoids	○ 5

Heure : ☐

Palettes sorties :	☐	Par jour:	☐
Article 1 :	☐	Article 1 :	☐
Article 2 :	☐	Article 2 :	☐
Article 3 :	☐	Article 3 :	☐

Temps moyen par palettes (sec.): ☐

ABC oui ○ non ○

Nom complet: Pinheiro Eddy
Classe: 4921

Répartition des surfaces de stockage
DEVOIR

A. Analyse de 4 entrepôts

Legendes
🟥 Quai d'arrivage
🟦 Quai d'expédition
⬜ Locaux annexes
🟩 Etagères

Regardez le meilleur plan d'entrepôt de chacun des 4 groupes de la classe.

	1.	2.	3.	4.
Brute:	256m2	256m2	256m2	256m2
Brute stock.:	224m2	240m2	224m2	224m2
Nette:	42m2	72m2	27m2	42m2
Etagère(s):	14	24	9	14
Degré d'utilisation:	19%	30%	12%	19%
Chariot elevateur:	gerbeur	gerbeur	contrepoid	mat retract
Temps/palette:	120s	134s	98s	100s

1. Quel plan est le meilleur en ce qui concerne:

Surface brute: 1
Surface nette: 2
Nombre de étagères: 2
Degré d'utilisation: 2
Temps par palette: 3

2. Quel plan est le meilleur en général? Pourquoi?
4. Malgré le degré d'utilisation qui est de 19%, donc relativement plus bas que celui du 2, le 4 a un local annexe, on fait moins de temps p/palette avec des engins plus rapides et on a une surface nette qui est considérablement plus grande que celle du 3, qui utilise justement des "contrepoids" qui occupent donc plus de place.

B. Comparaison avec votre entreprise.

Discutez de ces questions avec votre superviseur et écrivez vos explications.

1. Le meilleur plan que vous avez choisi (question A.2), est-il similaire ~~aux~~ au plan de votre entreprise (degré d'utilisation, mise en place, etc.)
Non, par rapport au degré d'utilisation, le 4 n'a rien de similaire, ni en mise en place. On a plus d'engins (mat rétractable, contrepoids, gerbeurs) et plus de surface, et on est une entreprise de production donc moins de degré d'utilisation par rapport au stock.

2. En général, lequel de ces 4 plans est le plus proche de votre entreprise (degré d'utilisation, mise en place, etc.)?
Le 2, en comparant avec une de nos halles on a remarqué que la mise en place des étagères est assez ressemblante si l'on déplace les quais perpendiculairement aux deux files d'étagères horizontales. Pour prélever la marchandise dans les étagères, on utilise un préparateur de commandes au lieu des chariots élévateurs.

3. En général, lequel de ces 4 plans est le plus différent de votre entreprise (degré d'util, mise en place, etc.)?
Le 3, car les quais se trouvent au centre de l'entrepôt, on a plus d'étagères et puis le plan est difficilement comparable à nos halles de logistique.

Figure 6.14 Two orchestration sheets for TinkerLamp—to set up a scenario on the left and for homework on the right.

so well, it may not be simply due to nostalgia or fear of change, but because it is a very efficient technology from an orchestration viewpoint, that is, at the third plane of usability. Darwin would say that paper is well adapted to the classroom ecosystem, through generations of paper forms. I do not claim that every paper interface, per se, is efficient, simply because it is paper based. Instead, paper has affordances that the design both captures and fails to take advantage of. For instance, the orchestration cards were appreciated by logistic teachers, who walked around the class with 5 to 10 cards in hand or in their pocket. We designed another augmented reality for training carpenters where apprentices had to manipulate around 25 paper cards. The experiment quickly revealed that they lost a lot of time and concentration in searching for the right card (Cuendet et al, 2013).

The functionality of paper in the classroom reminds me of a seminal paper by Hutchins (1995): "*How a cockpit remember it speeds*." Hutchins described an aircraft cockpit in which paper cards were still in use and explained their relevance. He modeled the aircraft cockpit as a distributed information system within which information flows between artifacts (e.g., altitude meters), pilots, tools, and these paper cards. A very similar analysis is relevant in the classroom context, where information flows across multiple channels and buffers (the blackboard, the notebooks, or devices like the Lantern).

In summary, technologies acquire high usability in a classroom if they satisfy the many constraints of this ecosystem. This requires the technology to be considered not as a monolithic device, but as a distributed system in which heterogeneous elements find their relevance in the classroom ecosystem.

Conclusions

I often use the slogan "no more log-in" to illustrate the pragmatic viewpoint of orchestration. Asking students to log into the system can take from 2 to 3 minutes, because some of them may have forgotten their password, while others have the caps lock on. This process consumes 5% of lesson time without any guarantee of bringing 5% additional learning gains. Log-in is a practical point, not a pedagogical one, but it has an impact on teaching. By also paying attention to practical aspects of classroom life, I wanted this book to end up with what a teacher's common sense would call "a method that works well." The term "works well" does not necessarily refer to the learning results of students (the intrinsic constraints), but very much to the extrinsic constraints: teachers' workload, the difficulty in managing students' participation, the stress related to time, and so on. The beginning of this book, because of its formalism, may appear far from classroom practices. I hope that this chapter shows why the modeling language is, in fact, so compatible with the practical considerations that arise while running a pedagogical scenario in a classroom or online.

The last two points overemphasize classroom practices to the detriment of online education. However, orchestration load and the third circle of usability can be translated into the MOOC space. For instance, the reason why online forums “work well,” despite large participant numbers, is that smart workflow operators have been introduced on forum platforms, such as the possibility of voting for someone else’s question, the possibility for TAs to filter the messages that the course teacher should personally reply to, and so on. I would say that MOOC forums are well-orchestrated technologies. The success of MOOCs can be understood by analyzing the orchestration load and usability of MOOCs.

Discussion

Every step in this book could be further developed, and every formula should be substantiated with authentic empirical data. Remaining at a high level has allowed me to use the modeling language across all aspects of a pedagogical scenario: the design, the learning processes, the analytics, and the adaptation/evolution process. The next steps are to collect the empirical evidence for some points and to implement the whole framework. This book can be understood as the presentation of a research program for the next decade. It is a proposal.

Design for analytics
This slogan appeared in the introduction. I hope the 42 points have clarified that the slogan does not mean that the goal of instructional design is to produce data. My claim is that a formal description of a MOOC makes analytics more powerful. This is not a book on statistics or machine learning, but it aims to describe educational challenges in a way that is meaningful for experts in these fields. The formal description of MOOCs is also motivated by the expectation that this could facilitate the sharing of data produced by running different MOOCs.

Pedagogy inside technology
Learning technologies is an interdisciplinary field. For the last 30 years, I have heard colleagues defending the preponderance of pedagogy over technology, as if there was a clear boundary between them. Of course, the goal of learning technologies is that students learn; the technology is not the goal. However, I hope that this book demonstrates that the craft of learning technologies is to understand the computational mechanisms that implement pedagogical ideas. The devil lies in the detail. The field of learning technologies is more than the union of pedagogical and computational pieces of knowledge. At its heart are these in-depth links that explain, for instance, how the operator associated with the edge of a graph will trigger different verbal interactions. This is why I have stressed the notion of workflow; even if it may sound like a terribly technical or even “bureaucratic” concept, it in fact encompasses the dynamics of rich pedagogical scenarios.

Cognitive tool for teachers
The proposed modeling language constitutes a tool for thinking about teaching. A tool is not a theory, and this book does not explain why people learn.

A model does not aim to be true, but rather to be useful for tackling complex situations such as the new rich integrated learning scenarios described in this book. However, this tool is not limited to MOOCs; it can apply to education with or without technologies. It can be applied to a pedagogical scenario in a class of 20 students that does not include any computational device. I believe that a professional teacher would be more comfortable when driving a lesson if he had a clear vision of inductive mechanisms, the role of positive and negative instance, near-miss examples, and if he orchestrated his lesson by thinking in terms of undergeneralization and overgeneralization. A formalization of the inductive process may help him to cope with the complexity of the induction processes of 20 heads. Orchestrating a class requires thinking in terms of planes and operators, with or without the use of computers. Actually, I have run several sessions of ArgueGraph without computers, by handling paper sheets, and it worked very well. Some technology is necessary to run the operators of ArgueGraph with 1,000 students, but the model, per se, is a guide for running this graph in technology-free classes.

Flipped Classes

This model applies to both classroom and online activities, especially when they are integrated into a consistent graph, as in “blended learning” or flipped classes. A class is said to be flipped when students get the theory before the class-contact hours; for instance, the students watch the lecture videos or read some texts before the lesson, and then do more interactive activities with the teacher, such as exercises, case studies, or real-world examples. What orchestration graphs describe is how these activities can be integrated within a consistent pedagogical scenario.

Implementation

This book does not address specific implementations of the modeling language, but I hope it will inspire developers. The ontologies used in current MOOC platforms are not as rich as one could hope for. Hence, this model could inspire the evolution of platforms or the design of new ones. A central challenge is to develop flexible workflows, that is, to invent architectures that take advantages of the power of workflow, but still allow substantial flexibility. I expect that blackboard architectures or “publish and subscribe” architectures (where distributed agents read and write in a central information space) could be one direction. Another challenge is to describe the workflow operators with a sufficient level of abstraction, so that they can be applied to a variety of data structures. Beyond a new platform, my dream would be that this model could be used for dialogue between platforms. I anticipate the emergence of an ecosystem of MOOC services. Some services (e.g., translation, coaching, proctoring) are already available, if we consider “service” in the business sense of the word. More services—in the computational sense—will emerge. For instance, a platform could send a list of students, plus some team formation criteria, to a specific service and would receive a list of teams that match these criteria in return.

XMOOCs versus cMOOCs

A model is only valid within some boundaries. This model focuses on formal education with no claim regarding its validity in informal/community learning. In all examples, a teacher has designed or at least prepared the learning activities. What if a student organizes his own learning activities or if a community constructs its own collective knowledge as in cMOOCs? In these cases, the activity graph has not been "designed," but emerges progressively through interactions. However, the way the community designs its own graph is shaped by the technology it uses, which has itself been designed by someone—this is what Fischer (2003) called "meta-design."

It may be useful at this point to discriminate between constructivism and connectivism (Siemens, 2005). The latter gives a holistic learning experience by which learners integrate learning, work, democracy, culture, and so on. As Fischer put it (2004), this approach is relevant "when the answer is not known." When the answer is well known by the teacher, a constructivist teacher may still desire to set up activities through which the learner elaborates knowledge, as described in this book. In other words, the model I presented is more relevant for xMOOCs than cMOOCs, that is, when there is a clear didactic plan, but it also allows socio-constructivist methods to be brought into the realm of xMOOCs.

Learners are more than just brains

Another limitation of this book is that it provides a very partial account of education. Like any model, it cuts out a slice in the multidimensional complexity of reality. It emphasizes the rational aspects of education, but does not consider the rich emotional aspects, the importance of the teacher's personality, the relationship the teacher builds with the learner, or the social dynamics among learners. My focus in no way implies that these factors are not important. Once, when I was an elementary school teacher, a father said to me "*Thank you. Since my daughter joined your class, she has started smiling again.*" What else is there? This moment was more important than the acquisition of any cognitive skill. The graphs I present focus on knowledge and skills, that is, the cognitive facet of education, but that does not dismiss the role of teachers as people, the importance of sharing their personality and enthusiasm, or how important it is that they believe in what learners will achieve. I hope that a teacher who is at ease with the mechanics of orchestration graphs, that is, the rational side of education, could more easily express his social and emotional talent, to the same extent that a musician needs a high technical mastery to express his personality.

Acknowledgments

Thanks to Xavier Silva, Louis D’Hainaut, and John Self, who shaped my rational stance on education. Thanks to my former colleagues Christian Depover, Peter Goodyear, Daniel Schneider, and Patrick Mendelsohn who influenced my work. Thanks to my team in the EPFL who developed the technologies that inspired this book: Patrick Jermann, Guillaume Zufferey, Son Do-Lenh, Sébastien Cuendet, Frédéric Kaplan and Hamed Alavi. Many of these team members also commented on earlier drafts of this book, as did Luis P. Prieto, Beat Schwendimann, Daniela Caballero, Lorenzo Lucignagno, Mina Shrivani, Ayberk Özgür, Mirko Raca, Severin Lemaignan, Wen Chen, Julia Fink. Thanks to Miguel Nussbaum, Erik Duval, Dan Schwartz, Olivier Levêque, Gerhard Fischer, Roland Tormey and Jeremy Roschelle who also commented earlier versions. Special thanks to Lukasz Kidzinski ,Valerie Chavez and Pierre Tchounikine who read this book more than once. Thanks to Patrick Aebisher, Martin Vetterli, Philippe Gillet and Jean-Marc Tassetto (Coorp Academy) who very early on realized the potential of MOOCs. Thanks to my friends from Waterloo, Honnelles, Lancaster, Igloolik, Prémanon, Genève, and Lausanne who know that 42 does not refer to any galaxy, but to our gravity-sensitive world. Thanks to EPFL Press for their support during this long journey. This book was written in Givrins, Villars-sur-Ollon, Haute-Nendaz, Château d’Oex, Les Diablerets, Skellefteå, Al Ain, Palo Alto, Angres, Waterloo, Bordeaux, Leiden, Pavia, Barbaggio, Edinburgh, Fort Williams, and Ullapool, as well as in many trains and airports.

References

Ainsworth, S. (1999). "The functions of multiple representations." *Computers & Education* 33 (2): 131–152.

Akshay, N., Deepu, S., Rahul, E.S., Ranjith, R., Jose, J., Unnikrishnan, R., Rao, R. & Bhavani, (2013)."Design and evaluation of a haptic simulator for vocational skill training and assessment." *39th Annual Conference of the IEEE Industrial Electronics Society*. Vienna.

Alavi, H. S., Dillenbourg, P. & Kaplan, F. (2009). "Distributed awareness for class orchestration." In *Learning in the Synergy of Multiple Disciplines*, 211–25. Berlin Heidelberg: Springer-Verlag.

Aronson, E., Blaney, N., Sikes, J., Stephan, G. & Snapp, M. (1978). *The Jigsaw Classroom*. Beverly Hills, CA: Sage Publication.

Ausubel, D. P. (1960). "The use of advance organizers in the learning and retention of meaningful verbal material." *Journal of Educational Psychology* 51 (5): 267.

Bachour, K., Kaplan, F. & Dillenbourg, P. (2010). "An interactive table for supporting participation balance in face-to-face collaborative learning." *IEEE Transactions on Learning Technologies* 3 (3): 203–13.

Bargh, J. A. & Schul, Y. (1980). "On the cognitive effects of teaching." *Journal of Educational Psychology* 72: 593–604.

Beattie IV, V., Collins, B. & McInnes, B. (1997). "Deep and surface learning: A simple or simplistic dichotomy?" *Accounting Education* 6 (1): 1–12.

Berger, A., Moretti, R., Chastonay, P., Dillenbourg, P., Bchir, A., Baddoura, R., Bengondo, C., Scherly, D., Ndumbe, P., Farah, P., & Kayser, B. (2001). "Teaching community health by exploiting international socio-cultural and economical differences." In *Proceedings of the First European Conference on Computer Supported Collaborative Learning*, ed. P. Dillenbourg, A. Eurelings & K. Hakkarainen, 97–105. Maastricht: Maastricht McLuhan Institute.

Blaye, A. (1988). "Confrontation socio-cognitive et résolution de problèmes." PhD dissertation. Centre de Recherche en Psychologie Cognitive, University of Provence.

Bloom, B. S. (1984). "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring." *Educational Researcher* 13 (6): 4–16.

Bloom, B. S. & Carroll, J. B. (1971). *Mastery Learning: Theory and Practice*, ed. J. H. Block. New York: Holt, Rinehart, and Winston.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H. & Krathwohl, D. R. (1956). *Taxonomy of educational objectives. Handbook I: Cognitive domain*. New York: David McKay.

Borgatti, Stephen P. & Everett, Martin G. (2006). “A graph-theoretic perspective on centrality.” *Social Networks*, 28: 466–84.

Brown, A. L. & Palincsar, A. S. (1987). *Reciprocal Teaching of Comprehension Strategies: A natural history of one program for enhancing learning*. Westport, CT: Ablex Publishing.

Bull, S. & Kay, J. (2010). “Open learner models.” In *Advances in Intelligent Tutoring Systems*, 301–22. Berlin Heidelberg: Springer-Verlag.

Chan, T. W. & Baskin, A. B. (1988). “Studying with the prince: The computer as a learning companion.” In *Proceedings of the First International Conference on Intelligent Tutoring Systems*, Vol. 194200.

Chase, C. S., Chin, D. B., Oppezzo, M. A. & Schwartz, D. L. (2009). “Teachable agents and the protégé effect: Increasing the effort towards learning.” *Journal of Science and Educational Technology*, 18: 334–52.

Clark, H. H. (1996). *Using language*, (vol. 1996, p. 92). Cambridge: Cambridge University Press.

Clark, H. H. & Brennan, S. E. (1991). “Grounding in communication.” In Resnick, L. B.; Levine, J. M.; Teasley, J. S. D., *Perspectives on socially shared cognition*, American Psychological Association, 13 (1991), 127–49.

Clark, H. H. & Wilkes-Gibbs, D. (1986). “Referring as a collaborative process.” *Cognition*, 22: 1–39.

Crouch, C. H. & Mazur, E. (2001). “Peer instruction: Ten years of experience and results.” *American Journal of Physics*, 69: 970–77.

Cuendet, S. & Dillenbourg, P. (2013). “The benefits and limitations of distributing a tangible interface in a classroom.” In *Proceedings of the Tenth Computer Supported Collaborative Learning Conference*, 137–44. Madison, WI.

d’Hainaut, L. (1983). *Des fins aux objectifs de l’éducation*. Paris: Nathan.

de Jong, T. & van Joolingen, W. (1998). “Scientific discovery learning with computer simulations of conceptual domains.” *Review of Educational Research* 68: 179–202.

de Jong, T. (2006). “Scaffolds for scientific discovery learning.” *Handling Complexity in Learning Environments: Research and theory*, 107–28.

Dewey, J. (1938). *Experience and Education*. New York: Macmillan.

Dickson, W. P. & Vereen, M. A. (1983). “Two students at one microcomputer.” *Theory into Practice*, 22 (4), Special Issue: *Microcomputers: A Revolution in Learning*: 296–300.

Diestel, R. (2005). *Graph Theory*. Electronic EditionHeidelberg: Springer.

Dillenbourg, P. (1989). “Designing a self-improving tutor: PROTO-TEG.” *Instructional Science*, 18 (3): 193–216.

Dillenbourg, P. (2002). “Over-scripting CSCL: The risks of blending collaborative learning with instructional design.” In *Three worlds of CSCL.*

Can we support CSCL?, ed. P. A. Kirschner, 61–91. Heerlen: Open Universiteit.

Dillenbourg, P. (2013). "Design for classroom orchestration." *Computers & Education*, 69: 485–92.

Dillenbourg, P., & Hong, F. (2008). "The mechanics of CSCL macro scripts." *International Journal of Computer-Supported Collaborative Learning*, 3 (1): 5–23.

Dillenbourg, P. & Jermann, P. (2010). "Technology for classroom orchestration." In *New Science of Learning: Cognition, Computers and Collaboration in Education*, ed. M. S. Khine & I. M. Saleh, 525–52). Dordrecht: Springer-Verlag.

Dillenbourg, P. & Self J. A. (1992). "A framework for learner modelling." *Interactive Learning Environments*, 2 (2): 111–37.

Dillenbourg, P. & Tchounikine, P. (2007). "Flexibility in macro-scripts for computer-supported collaborative learning." *Journal of Computer Assisted Learning*, 23 (1): 1–13.

Dillenbourg, P. & Traum, P. (1999). "Does a shared screen make a shared understanding?" *Proceedings of the Third Computer-Supported Collaborative Learning Conference*, ed. C. Hoadley and J. Roschelle, Stanford, CA: 127–35.

Dillenbourg, P., Zufferey, G., Alavi, H., Jermann, P., Do-Lenh, S., Bonnard, Q., Cuendet, S. & Kaplan, F. (2011). "Classroom orchestration: The third circle of usability." *Proceedings of the Ninth Computer-Supported Collaborative Learning Conference*. Hong Kong.

Do-Lenh, S., Zufferey, G., Jermann, P. & Dillenbourg, P. (2010). "TinkerSheets and Reflection: Design augmented papers to facilitate student's reflection." In *ACM International Conference on Ubiquitous Computing (UBICOMP 2010): Workshop on Paper Computing*, PaperComp'10.

Engeli, M. (2000). *Digital stories*. Berlin: Springer-Verlag.

Fischer, G. (2003). "Meta-design: Beyond user-centered and participatory design." In *Proceedings of HCI International*, 88–92.

Fischer, G. (2004). "Social creativity: Turning barriers into opportunities for collaborative design." In *Proceedings of the Eighth Conference on Participatory Design. Artful Integration: Interweaving Media, Materials and Practices*, 1: 152–61. ACM.

Foster, G. & Ysseldyke, J. (1976). "Expectancy and halo effects as a result of artificially induced teacher bias." *Contemporary Educational Psychology*, 1 (1): 37–45.

Freinet, C. (1966). *Essai de psychologie sensible appliquée à l'éducation*. Neuchâtel: Delachaux & Niestlé.

Gijlers, H. & de Jong, T. (2005). "Confronting ideas in collaborative scientific discovery learning." Paper presented at *AERA 2005*, Montreal.

Guilford, J. P. (1956). "The structure of intellect." *Psychological Bulletin*, 53 (4): 267–293.

Hmelo-Silver, C. E. (2004). "Problem-based learning: What and how do students learn?" *Educational Psychology Review*, 16 (3): 235–66.

Hoppe, H. U. (1995). Using multiple student modeling to parameterize group learning. In J. Greer (Ed.), *Proceedings of the Seventh World Conference on Artificial Intelligence in Education* (pp. 234-241). Charlottesville, VA: Association for the Advancement of Computing in Education.

Hoppe, H. U. & Ploetzner, R. (1999). "Can analytic models support learning in groups." In P. Dillenbourg, *Collaborative Learning: Cognitive and Computational Approaches*, 147–168. Oxford: Elsevier.

Hutchins, E. (1995). "How a cockpit remembers its speeds." *Cognitive science*, 19 (3): 265–88.

Infante, C., Hidalgo, P., Nussbaum, M., Alarcón, R. & Gottlieb, A. (2009). "Multiple mice based collaborative one-to-one learning." *Computers & Education*, 53 (2): 393–401.

Jermann, P. & Dillenbourg, P. (1999). "An analysis of learner arguments in a collective learning environment." In *Proceedings of the 1999 Conference on Computer Support for Collaborative Learning*, 33. International Society of the Learning Sciences.

Jermann, P. & Dillenbourg, P. (2008). "Group mirrors to support interaction regulation in collaborative problem solving." *Computers & Education*, 51 (1): 279–96.

Jermann, P., Mullins, D., Nüssli, M.-A. & Dillenbourg, P. (2011). "Collaborative gaze footprints: Correlates of interaction quality." In *Proceedings of the Int. Conference on Computer Supported Collaborative Learning*. Hong Kong.

Jermann, P. & Nüssli, M. A. (2012). "Effects of sharing text selections on gaze cross-recurrence and interaction quality in a pair programming task." In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, 1125–34. ACM.

Kadushin, C. (2012). *Understanding Social Networks: Theories, concepts, and findings*. Oxford: Oxford University Press.

Li, N., Verma, H., Skevi, A., Zufferey G., Blom, J., Dillenbourg P. et al. (2014) "Watching MOOCs together: Investigating co-located MOOC study groups." *Distance Education*, 1–17.

Liu, Y., Hsueh, P. Y., Lai, J., Sangin, M., Nussli, M. A. & Dillenbourg, P. (2009). "Who is the expert? Analyzing gaze data to predict expertise level in collaborative applications." In *IEEE International Conference on Multimedia and Expo, 2009*, 898–901). IEEE.

McPherson, N., Smith-Lovin, L., Cook, J. M. (2001). "Birds of a feather: Homophily in social networks." *Annual Review of Sociology,* 27: 415–44.

Miyake, N. (1986). "Constructive interaction and the iterative process of understanding." *Cognitive Science*, 10: 151–77.

Mugny, G. & Doise, W. (1978). "Socio-cognitive conflict and structure of individual and collective performances." *European Journal of Social Psychology*, 8 (2): 181–92.

Mulholland, P., Anastopoulou, S., Collins, T., Feisst, M., Gaved, M., Kerawalla, L., Paxton, M., Scanlon, E., Sharples, M. & Wright, M. (2012). “nQuire: Technological support for personal inquiry learning.” *IEEE Transactions on Learning Technologies*, 5 (2): 157–69.

Muukkonen, H., Hakkarainen, K. & Lakkala, M. (1999). “Collaborative technology for facilitating progressive inquiry: Future learning environment tools.” In *Proceedings of the 1999 Conference on Computer Support for Collaborative Learning* (CSCL ‹99), eds. Christopher M. Hoadley and Jeremy Roschelle (p. 51). International Society of the Learning Sciences.

Paas, F., Renkl, A. & Sweller, J. (2003). “Cognitive load theory and instructional design: Recent developments.” *Educational Psychologist*, 38 (1): 1–4.

Parriaux, A. (2009). *Géologie: Bases pour l’ingénieur*. Lausanne: Presses polytechniques et universitaires romandes.

Perkins, D. N. & Salomon, G. (1992). “Transfer of learning.” In T. Husen and N. Postlethwaite (EdS), *International Encyclopedia of Education*, Second edition, pp. 6452-6457). Oxford, UK: Pergamon Press.

Prieto, L. P., Asensio-Perez, J.-I., Dimitriadis, Y., Gomez-Sanchez, E. & Munoz-Cristobal, J.-A. (2011). “GLUE!-PS: A Multi-language architecture and data model to deploy TEL designs to multiple learning environments.” In *EC-TEL 2011* ed. C. Delgado Kloos et al., LNCS 6964: 285–298, Berlin Heidelberg: Springer-Verlag.

Prieto, L. P., Dlab, M. H., Gutiérrez, I., Abdulwahed, M. & Balid, W. (2011). “Orchestrating technology enhanced learning: A literature review and a conceptual framework.” *International Journal of Technology Enhanced Learning*, 3 (6): 583–98.

Prieto, L. P., Villagrá-Sobrino, S., Jorrín-Abellán, I. M. , Martínez-Monés, A & Dimitriadis, Y. (2011). “Recurrent routines: Analyzing and supporting orchestration in technology-enhanced primary classrooms.” *Computers & Education*, 57 (1): 1214–27.

Raca, M. & Dillenbourg, P. (2013). “System for assessing classroom attention.” In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*, 265–69. ACM.

Roschelle, J. (1992). “Learning by collaborating: Convergent conceptual change.” *The Journal of the Learning Sciences*, 2 (3): 235–76.

Roschelle, J. & Teasley S. D. (1995). “The construction of shared knowledge in collaborative problem solving.” In *Computer-Supported Collaborative Learning*, ed. C.E. O’Malley, 69–197. Berlin: Springer-Verlag.

Rouet, J. F. (1992). “Cognitive processing of hyperdocuments: When does nonlinearity help?” In *Proceedings of the ACM conference on Hypertext*, 131–40. ACM.

Salomon, G. & Globerson, T. (1989). “When teams do not function the way they ought to.” *International Journal of Educational Research*, 13 (1): 89–100.

Schellens, T., Van Keer, H. & Valcke, M. (2005). "The impact of role assignment on knowledge construction in asynchronous discussion groups: A multilevel analysis." *Small Group Research*, 36 (6): 704–45.

Schoenfeld, A. H. (1988). "When good teaching leads to bad results: The disasters of 'well-taught' mathematics courses." *Educational Psychologist*, 23 (2): 145–66.

Schwartz, D.L. (1995). "The emergence of abstract dyad representations in dyad problem solving." *The Journal of the Learning Sciences*, 4 (3): 321–54.

Schwartz, D. L. & Bransford, J. D. (1998). "A time for telling." *Cognition and instruction*, 16 (4): 475-5223.

Self, J. A. (1990). "Bypassing the intractable problem of student modelling." *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, 107–23.aC. Frasson and G. Gauthier, eds., Ablex Publishing Company, Norwood, NJ.

Self, J. A. (1992). "Computational orchestrationsmathetics: The missinglLink in intelligent tutoring systems research?" In *New Directions for Intelligent Tutoring Systems*, ed. E. Costa, NATO ASI Series, 91: 38–56.

Siemens, G. (2005). "Connectivism: A learning theory for the digital age." *International Journal of Instructional Technology and Distance Learning*, 2 (1): 3–10.

Steiner, D. D. & Rain, J. S. (1989). "Immediate and delayed primacy and recency effects in performance evaluation." *Journal of Applied Psychology*, 74 (1): 136.

Suthers, D. D., Dwyer, N., Medina, R. & Vatrapu, R. (2010). "A framework for conceptualizing, representing, and analyzing distributed interaction." *International Journal of Computer-Supported Collaborative Learning*, 5 (1): 5–42.

Szewkis, E., Nussbaum, M., Rosen, T., Abalos, J., Denardin, F., Caballero, D. & Alcoholado, C. (2011). "Collaboration within large groups in the classroom." *International Journal of Computer-Supported Collaborative Learning*, 6 (4): 561–75.

Tourneur, Y. (1975). "Effets des objectifs dans l'apprentissage. Etude expérimentale." In *Recherche en Education, n°6*. Bruxelles: Direction générale de l'organisation des études.

Traum, D. R. (1999). "Computational models of grounding in collaborative systems." In *Psychological Models of Communication in Collaborative Systems—Papers from the AAAI Fall Symposium*, 124–31.

Van Lehn, K. (1988). *Toward a theory of impasse-driven learning*, 19–41. New York (NY), Springer US.

Vinciarelli, A., Pantic, M. & Bourlard, H. (2009). "Social signal processing: Survey of an emerging domain." *Image and Vision Computing*, 27 (12): 1743–59.

Vosniadou, S. (1994). "Capturing and modeling the process of conceptual change." *Learning and Instruction*, 4: 45–69.

Vygotsky, L. S. (1962). *Thought and Language*. Cambridge, MA: MIT Press.

Vygotsky, L. S. (1994). "Extracts from thought and language and mind in society." *Language, literacy and learning in educational practice*, 45–58. Clevedon: Multilingual Matters.

Webb, N. M. (1989). "Peer interaction and learning in small groups." *International Journal of Educational Research*, 13 (1): 21–39.

Webb, N. M. (1991). "Task related verbal interaction and mathematics learning in small groups." *Journal for Research in Mathematics Education*, 22 (5): 366–89.

Weinberger, A. & Fischer, F. (2006). "A framework to analyze argumentative knowledge construction in computer-supported collaborative learning." *Computers & Education*, 46 (1): 71–95.

Weinberger, A., Stegmann, K., Fischer, F. & Mandl, H. (2007). "Scripting argumentative knowledge construction in computer-supported learning environments." In *Scripting computer-supported collaborative learning*, 191–211. Springer US.In F. Fischer, H. Mandl, J. Haake & I. Kollar (Eds.), Scripting computer-supported communication of knowledge - cognitive, computational and educational perspectives (pp. 191-211). New York: Springer.

Wenger, E. (2010). "Communities of practice and social learning systems: The career of a concept." In *Social Learning Systems and Communities of Practice*, 179–98. London: Springer-Verlag.

Whyte, Jr., W. H. (1952). «Groupthink». *Fortune*. March, 114–117.

Wichmann, A., Hoppe, U., Spikol, D., Milrad, M., Anastopoulou, S., Sharples, M., Pea, R., Maldonado, H. & de Jong, T. (2010). "Three perspectives on technology support in inquiry learning: Personal inquiry, mobile collaboratories and emerging learning objects." In *Proceedings of the 9th International Conference of the Learning Sciences*, (ICLS '10), ed. Kimberly Gomez, Leilah Lyons, and Joshua Radinsky, 2: 499–500. International Society of the Learning Sciences.

Xu, Guandong et al. (2010). *Web Mining and Social Networking: Techniques and Applications*, p. 25. New York (NY), Springer.

Zufferey, G., Jermann, P., Lucchi, A. & Dillenbourg, P. (2009). "TinkerSheets: Using paper forms to control and visualize tangible simulations." In *Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pp. 377–84. TEI '09. ACM, New York, NY.

By modeling pedagogical scenarios as directed geometrical graphs and proposing an associated modeling language, this book describes how rich learning activities, often designed for small classes, can be scaled up for use with thousands of participants.

With the vertices of these graphs representing learning activities and the edges capturing the pedagogical relationship between activities, individual, team, and class-wide activities are integrated into a consistent whole.

The workflow mechanisms modeled in the graphs enable the construction of scenarios that are richer than those currently implemented in MOOCs. The cognitive states of learners in two consecutive activities feed a transition matrix, which encapsulates the probability of succeeding in the second activity, based on success in the former. This transition matrix is summarized by a numerical value, which is used as the weight of the edge.

This pedagogical framework is connected to stochastic models, with the goal of making learning analytics more appealing for data scientists. However, the proposed modeling language is not only useful in learning technologies, it also allows researchers in learning sciences to formally describe the structure of any lesson, from an elementary school lesson with 20 students to an online course with 20,000 participants.

Pierre Dillenbourg is a professor of learning technologies at the EPFL and the academic director of the EPFL Center for Digital Education. This center produces MOOCs that involve nearly one million participants. Pierre has conducted research in learning technologies since 1985. He currently leads a lab (chilli.epfl.ch) that among other things explores augmented reality, tangible interfaces, paper-based computing, and eye tracking. He started as a schoolteacher in Brussels, graduated in educational sciences at the University of Mons (Belgium) and obtained a PhD in computer science from the University of Lancaster (UK).

ISBN: 978-2-940222-84-1

9 782940 222841 >

EPFL Press