Modeling Lessons in Classical Ballet through Petri Nets

Adriano Baratè[1], Luca A. Ludovico[1], and Andrea Piermattei[2]

[1] Laboratorio di Informatica Musicale
Dipartimento di Informatica, Università degli Studi di Milano
{adriano.barate, luca.ludovico}@unimi.it
[2] Corpo di Ballo del Teatro alla Scala
andrea.piermattei@balletto.net

Abstract. Petri nets are a mathematical modeling language for the description of distributed systems. They are particularly effective to provide a synthetic, multi-layer graphical representation of processes, unveiling aspects such as iterative structures and concurrency. This paper aims to apply Petri nets to the formalization of a typical lesson in classical ballet. After analyzing the basic principles and the commonly accepted rules to structure a ballet class, we will define a net model that can be conveniently modified by changing either its initial marking or the network topology in order to take into account factors such as lesson duration, the age and skill level of dancers, etc. Such a model can be adopted also to automatically produce a lesson structure with customized features, as demonstrated by the final case study.

Keywords. Ballet class, modeling, Petri nets

1 Introduction

Modeling a complex activity such as the design, structuring and execution of a generic ballet class is a challenging matter. First, there is not one commonly-accepted lesson structure, but rather a number of methods and schools, each with their own set of habits, practices and accepted rules. In addition, a ballet class can address a heterogeneous target, ranging from the young student to the professional dancer. Finally, there are requirements and restrictions that have a practical origin: the duration of the lesson, the number of participants, the type of skills to be developed, etc.

In this extremely heterogeneous field, we have to focus on some practices that may be considered standard, thanks to the endorsement of experienced teachers and professional dancers. The goal is to determine a generic reference model, as close as possible to commonly-accepted practices, and to identify a set of variables able to adapt the model to different use cases.

A major issue is the choice of a suitable modeling tool, so that the resulting models can be descriptively effective, flexible enough, and understandable also by non-experts. As explained in the following, a good solution is the adoption of Petri nets, a formal tool to describe processes through a graphical representation easy to produce,

read and modify.
Before getting to the heart of the subject, it is worth explaining the meaning of this proposal. Models are typically approximations of real situations, often presenting variations and nuances to be preserved and emphasized rather than flattened. In this case, the modeling goal is even harder, since there is no agreement about a commonly-accepted structure for a ballet class, as illustrated in some detail in the next section. Nevertheless, efforts in this sense can produce interesting results:

- It is possible to discover similarities among different methods and schools;
- Thanks to the characteristics of Petri nets and their graphical representation, it is easy to modify models and study their behavior;
- The resulting models can be employed in ballet-oriented recommender systems and training applications, in order to automatically build user-tailored exercises and lessons.

This paper is structured as follows: Section 2 illustrates the main training systems in use in classical ballet and provides some links to literature; Section 3 discusses the characteristics of the typical structure of a ballet lesson; Section 4 presents a short overview about Petri nets, giving details about terminology, syntactic elements, and evolution rules; Section 5 shows a multi-layer model for a ballet class; finally, Section 6 addresses the issue of model customization and provides some clarifying examples.

2 Classical Ballet Training Systems

There are several standardized, widespread, classical ballet training systems, often named after their creators, and called *methods* or *schools*. The two prevailing systems from Russia are the *Vaganova method* by Agrippina Vaganova [1] and the *Legat method* by Nikolai Legat [2]. In Italy a widespread training system was developed by – and named after – the Italian dancer Enrico Cecchetti [3]. Another relevant example is the *Royal Academy of Dance (RAD) method*, collectively created by a group of ballet professionals and currently used in more than 70 Countries [4,5]. French ballet has no standard method: the major ballet schools employ their own training system, as for the Paris Opera Ballet School, Conservatoire National Supérieur de Musique et de Danse, and Académie de Danse Classique Princesse Grace. Finally, as it regards the United States, it is worth mentioning the School of American Ballet created by George Balanchine, who significantly contributed to the modernization of ballet [6]. In any case, the strong geographical and cultural roots of the mentioned methods have not prevented them to overcome national borders and become internationally-recognized teaching standards.
Quite surprisingly, even if each training system has been designed to produce unique aesthetic qualities and technical skills from its students, there is a general agreement about the type and sequence of exercises characterizing the typical ballet class. In the following, we will adopt the lesson structure currently in use at La Scala Theatre Ballet School, briefly described in Section 3.
As it regards glossary, since ballet became formalized in France, a significant part of terminology is in the French language. The most relevant terms for the comprehen-

sion of this work will be defined and shortly explained the first time they appear in the text. For further details, such as the meaning of other terms, cross-references and translations into other languages, please refer to [7], [8], and [9].

3 Lesson Structure

In this section we will describe the structure of a ballet class according to the training method in use at Teatro alla Scala of Milan, Italy. Even if this approach is substantially rooted in the Vaganova method, most exercises can be considered standard and are commonly performed also in other schools, in the same or a similar order.
We aim to provide a multi-layer description of such a structure, adopting a top-down approach, namely moving from the highest level of abstraction (the whole lesson) to the lowest one (how single pieces can be built as sequences and repetitions of music fragments). The resulting structure is made of multiple layers, and each layer can be seen as a detailing of the one immediately above.
As it regards the highest level, a dance lesson can be basically divided into two main parts. At first, dancers are attached to the *barre*, namely a sturdy horizontal bar – approximately waist height – used for warm-up and stretching activities. A common sequence of exercises proposed in the first phase is: *deux mains à la barre*, *pliés*, *tendus*, *jetés*, *ronds de jambe*, *fondus*, *frappés*, *ronds de jambe en l'air*, *adages*, *grands battements*, and finally stretching exercises.
The second stage takes place with dancers detached from the bar. This part can be further divided into:

- Center – Training activities are performed in the middle of the rehearsal room. The typical sequence of exercises is: *petits adages*, *tendus*, *jetés*, *ronds de jambe*, *fondus*, *grands adages*, *grands battements*, *pirouettes*, *grands pirouettes*;
- Jumps – This part of the lesson includes *petits sauts*, *moyens sauts*, and *grands sauts* (i.e. small, medium and great jumps respectively);
- Great turns – The *coda*, namely the concluding segment of the lesson, often contains virtuosity exercises, such as *manege de jettés*, *manege de piquet*, *fouettés*, *pirouettes a la seconde*, etc.

All the mentioned exercises are usually performed for each leg *en dehors* (implying a clockwise circle for a right working leg and a counter-clockwise circle for the left one) and *en dedans* (i.e. a clockwise circle for the left leg, and a counter-clockwise circle for the right one). This can be achieved either using two short pieces, or a single one with a double number of beats.
Each exercise type could be described in detail as it regards its technical and aesthetic goals, but a complete discussion of this subject would go clearly beyond the scope of the present work.
In summary, the division of a lesson in successive stages can be seen as the top level of abstraction, whereas the inner composition of each stage – namely the sequence of exercises and their repetitions – represents the intermediate one.

Now, let us describe the lower level of abstraction. Exercises can be customized by the dance teacher to adjust their length according to the training goal. The number of beats is typically a power of 2 (e.g., 16, 32, 64, 128, etc.) or a sum of powers of 2 (e.g., 48, 96, etc.). A common way to build a variable-length music piece is the aggregation and repetition of shorter cells, called *fragments*, properly organized in a musical continuum. The inner structure of a music piece is the last level that we want to unveil and describe in our multi-layer model.

A trivial example of a variable-length exercise could be composed by an initial 16-beats *intro* followed by a variable number of repetitions of the same 16-beats fragment, called *A*. The resulting sequences are:

- *intro, A* (32 beats);
- *intro, A, A* (48 beats);
- *intro, A, A, A* (64 beats);
- …

A more complex example can be produced with an initial 16-beats *intro* followed by other 16-beats fragments named *A*, *B*, and *C*, variously arranged to improve the musical result and to avoid a sense of repetitiveness:

- *intro, A* (32 beats);
- *intro, A, B* (48 beats);
- *intro, A, B, C* (64 beats);
- *intro, A, B, C, A* (80 beats);
- *intro, A, B, C, A, B* (96 beats);
- *intro, A, B, C, A, B, C* (112 beats);
- *intro, A, B, C, A, B, C, A* (128 beats).

Modular structures like the ones described above allow the dance teacher to select the most appropriate length for each exercise. For example, in the latter case – with the metronome set to 120 BPM – it is possible to generate a piece whose duration changes noticeably, ranging between 16 seconds and 1 minute and 4 seconds. It is worth noting that the revelation of the inner structure of a piece could provide also an effective method to compress information.

For instance, if the music for the ballet class is in the form of already-available audio tracks instead of being performed on the fly by a piano player, then the availability of independent fragments will produce a double benefit, in terms of both modularity and space occupation: it is sufficient to have a unique audio track equipped with ad hoc markers – or alternatively the recording of each fragment alone – to compose a sequence of any kind and duration. Some clarifying examples will be presented in Section 5.

4 A Brief Introduction to Petri Nets

A Petri net is an abstract and formal model to represent the dynamic behavior of a system with asynchronous and concurrent activities [10].

Petri Nets consist in a set of basic objects: *places*, *transitions* and *arcs*, whose graphical representations are circles, rectangles, and oriented lines respectively. Both places and transitions are called *nodes*.

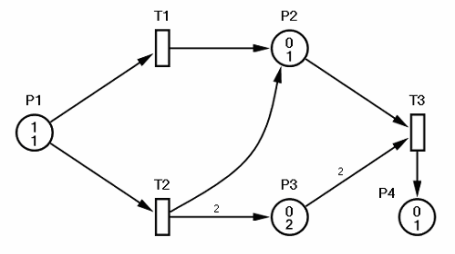

Fig. 1. An example of Petri net.

In Figure 1, an elementary Petri Net is shown. P1, P2, P3, P4 are places, T1, T2, T3 are transitions, and the oriented lines represent arcs. An arc can connect only nodes of different kind, i.e. places to transitions and vice versa. The number possibly associated to arcs is called *arc weight*, and its meaning will be explained soon. When not specified, the default value for arc weight is 1.

A key concept for Petri nets is the idea of *marking*, implemented through the metaphor of *tokens*. At a given time, any place holds a non-negative number of tokens, graphically indicated by the upper value inside the circle. The lower value indicates place capacity, i.e. the maximum number of housed tokens.

Tokens can be transferred from place to place according to policies known as *firing rules*. The dynamic evolution of a Petri net is determined by the following rules:

- A transition is enabled when all the incoming places of that transition present a number of tokens greater or equal to the weights of the corresponding incoming arcs, and – after the fire of the transition – the marking of all the output places will be less than or equal to their capacities;
- When a transition is enabled, the fire drops from the incoming places a number of tokens equal to the weights of the incoming arcs and adds to each outgoing place a number of tokens equal to the weights of the corresponding outgoing arc.

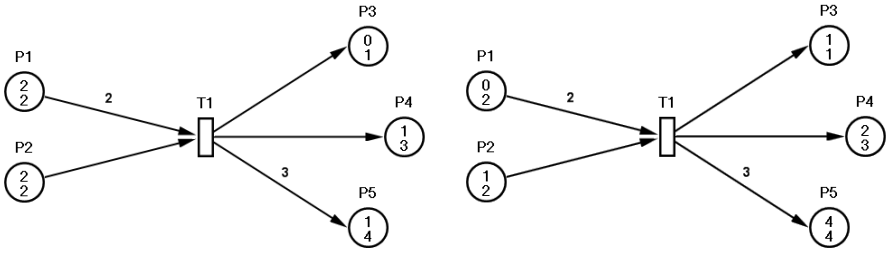

Fig. 2. The left part shows a net where transition T1 is enabled, i.e. it is ready to fire; the right part illustrates the new marking of the net after the fire. Please note that tokens have not been transferred, rather consumed in input places and created in output places in accordance with arc weights.

Another important concept is *refinement*, a simple morphism that allows the decomposition of complex Petri Nets into simpler ones thanks to the adoption of subnets. Subnets are very important for our work, since a multi-layer structure like the one described in Section 3 can be described in a compact and effective way through the use of multiple refinements, as shown in the following figures.
For the goals of this work, the need arises to introduce some standard extensions of Petri nets. Since they are well known in literature, please refer to the provided references for a formal description of such extensions and their properties.
First, the original theory of Petri nets intentionally does not model time; consequently, when a transition is enabled, there is no time constraint that forces it to fire immediately. Conversely, in the approach presented below we need to describe not only the structure, but also the timing of the model. In order to manage these cases, timed Petri nets have evolved, where there are places and transitions that are timed and others which are not [11]. In our model, places will be timed.
Another extension, known as stochastic Petri nets, lets us introduce non-determinism through adjustable randomness of the transitions [12]. Our model uses a "light" version of this feature, mainly to sort concurrent enabled transitions. To this purpose, we define the concept of *probabilistic weight* associated with an arc. Our definition states that, when many transitions are enabled to fire, the probability of choosing one specific transition can be calculated as follows:

- If all the enabled transitions have all the input/output arcs with probabilistic weights equal to 0, the first transition to fire is chosen randomly;
- If at least one of the enabled transitions has a probabilistic weight of the input/output arcs greater than 0, the probability of choosing one specific transition to fire is the sum of its input/output probabilistic weights divided by the sum of all the input/output probabilistic weights of all the enabled transitions.

The probabilistic weight is used in this work when: i) two or more transitions are in alternative, but ii) we want to follow a specific path until a particular event occurs. To accomplish this, the last transition to fire must have all the input/output arcs with probabilistic weights equal to 0. The transition with a probabilistic weight set to 0 will fire only when all the other transitions are no more enabled due to their marking. In the figures below, probabilistic weights will be enclosed in square brackets.

5 The Multi-Layer Model

In this section we present the 3-layer model of a standard ballet class through timed and probabilistic Petri nets. The concept of refinement provides a clear and intuitive way to describe the relationship among layers and to clarify their different degree of abstraction.
In the model we will assign two different roles to places: places that are associated to actions to be performed (e.g., "start the lesson", "play a music piece", "insert a break", etc.), and places whose aim is only to properly enable/disable transitions (e.g., "count the repetitions"). In the figures below, action places are represented through

white circles and counter places through smaller grayed circles.
Places of the former kind correspond to timed slots of a lesson. The meaning of a slot changes from layer to layer: at the lowest level, it can be a subpart of a music piece – and in this case it takes the time required to play the loop – as well as the break between the execution of two exercises; at the highest level, a slot corresponds to the total timing of a group of exercises. When a transition fires and puts tokens into this kind of places, the execution of the associated action starts immediately and goes on for the whole duration of the slot. Only after the slot's time has run out, the conditions to trigger the subsequent transitions are evaluated, and in case of success enabled transitions fire immediately.
Even if not associated to any practical action, the role of the latter kind of places is fundamental to confer a given structure to the model. Counter places are used to properly enable/disable transitions, and their marking – i.e. the number of contained tokes – can be modified also on the fly to modify the lesson structure, as detailed in Section 6.
Now let us describe the model in detail using a top-down approach. The top layer, shown in Figure 3, is a trivial Petri net with a linear structure. This macro-segmentation of the lesson contains only action places. The stage when dancers are detached from the bar (Middle) could be refined into the 3 subparts described in Section 3 – i.e. center, jumps, and great turns – but it would be only a different division of the top layer that does not introduce a different degree of abstraction in the model.

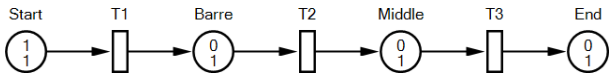

Fig. 3. The Petri net describing the top layer of the model.

The middle layer, shown in Figure 4, provides an insight on the sequence of exercises that compose each macro-section of the lesson. Once again the resulting Petri net is basically a linear sequence of action places, in this case associated with the performance of a complete exercise (or potentially to a break between exercises), but the presence of counter places allows to set the number of repetitions for each exercise, thus providing a first aspect of flexibility to the model.

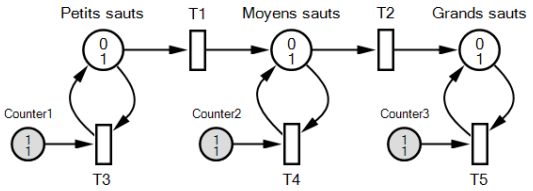

Fig. 4. The Petri net describing a part of the middle layer, namely a refinement of the Middle place.

Finally, the lowest layer addresses the detailed description of an exercise in terms of music fragments to mount in order to produce a target piece. Each music piece presents its own characteristics, and only a thorough knowledge of the repertoire allows

to suitably segment a score so as to obtain a satisfactory result as it regards both the musicality and the technical goals of the exercise. This is a typical ability of a skilled piano player used to accompany ballet training.

In Figure 5 we present the models of some standard sequences strongly rooted in the piano repertoire for ballet classes. Such models are the formalization in terms of Petri nets of the examples mentioned in Section 3. At this level of detail, probabilistic weights are used to manage repetitions. In Figure 5 we have used generic weights, labelled [w*n*], initialized to 0 before the first firing. During the net execution, these values can be set to a positive value and the weight of concurrent arcs to 0, thus stopping the loop before its natural end determined by the consumption of tokens in counter places.

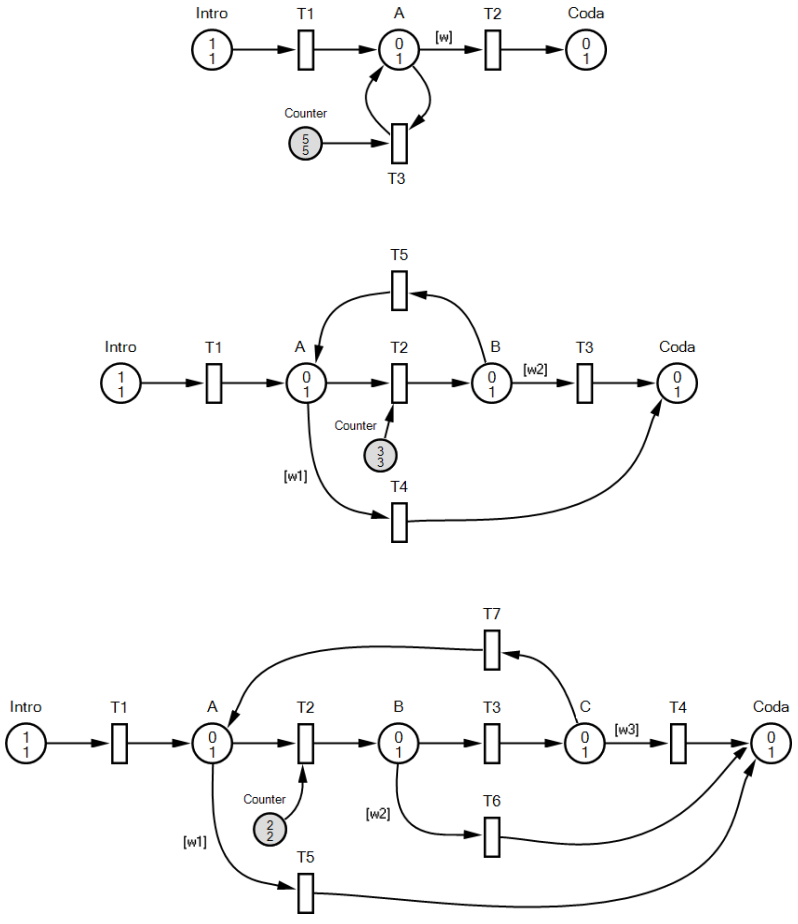

Fig. 5. The Petri nets describing the models of some pieces.

The upper structure shown in Figure 5 allows the generation of pieces organized as follows:

- *intro, A, coda*;
- *intro, A, A, coda*;
- *intro, A, A, A, coda*;
- *intro, A, A, A, A, coda*;
- *intro, A, A, A, A, A, coda*;
- *intro, A, A, A, A, A, A, coda.*

If no value is set for the probabilistic weight [w], T2 and T3 have the same probability to fire, until the last token in the counter has been consumed. Vice versa, if [w] is initially set to 0, place A will be executed 6 times. If [w] is changed on the fly, the number of iterations is changed accordingly.

Similarly, the other structures shown in Figure 5 present additional arc weights that allow to jump to *coda* on demand. The second model allows the generation of the following combinations:

- *intro, A, coda*;
- *intro, A, B, coda*;
- *intro, A, B, A, coda*;
- *intro, A, B, A, B, coda;*
- *intro, A, B, A, B, A, coda;*
- *intro, A, B, A, B, A, B, coda;*
- *intro, A, B, A, B, A, B, A, coda.*

Finally, the lower part of Figure 5 provides the following combinations:

- *intro, A, coda*;
- *intro, A, B, coda*;
- *intro, A, B, C, coda*;
- *intro, A, B, C, A, coda;*
- *intro, A, B, C, A, B, coda;*
- *intro, A, B, C, A, B, C, coda;*
- *intro, A, B, C, A, B, C, A, coda.*

For the sake of clarity, you can refer to the music pieces offered by Ballet Class, a free app for individual ballet training available on iOS and Android marketplaces. Thanks to the cooperation with experienced pianists from La Scala theater, in Ballet Class audio tracks are not saved in all their variants, rather music pieces are built on the fly by joining shorter audio fragments in accordance with the number of beats selected by the user. Even if Ballet Class engine does not parse Petri nets, it intrinsically adopts the same approach. Specifically, the first model in Figure 5 corresponds to the structure of the piece “Tendus d”, the second model to “Jetés C”, and the third model to “Frappés c”.

6 Customizing the Model

In this section we briefly outline the possibilities offered by a customization of the model. In Section 1 we have underlined the importance of flexibility in order to adapt the model to real situations and specific educational targets. To achieve this result, Petri nets can be modified to a certain extent also after the formalization of the model.
The initial marking – i.e. the number and position of tokens before the very first occurrence of firing – is a key factor that influences the evolution of the net. For example, it would be sufficient to add tokens inside counter places and/or properly set probabilistic weights to change the number of repetitions of a given action (e.g., multiple executions of the exercise, additional fragments inside a music piece, etc.). Modifications of net marking can occur on the fly too, thus providing the user with real-time control over the execution of the net [13]. For instance, a teacher could realize that the time is running out and the lesson has to be shortened, or conversely that the current exercise has to be repeated.
A more complex type of modification regards net topology: in this case, the structure itself can be modified by adding, removing or readjusting elements such as places, transitions, arcs, weights, etc. By changing net topology, on one side the user could skip unwanted exercises, insert or delete breaks between pieces, improve the way an audio track is built; but on the other side, he/she could invert lesson sections, play multiple audio tracks simultaneously, and erroneously join music fragments belonging to different pieces. Even if this approach is extremely powerful and flexible, it is also likely to overturn the identified model, and consequently it should be carefully evaluated.
As a further example of net modification, a certain amount of uncertainty can be introduced, in order to make the lesson experience more varied and less predictable. This result can be achieved by modifying or ignoring probabilistic weights. For the iterative structures of our model – where the next-step transition has probabilistic weight set to 0, so it can fire only after the consumption of all the n tokens in the counter place – without probabilistic weights the number of repetitions would unpredictably get a value $m \in [1 \dots n+1]$.
So far, modifications have been demanded to the user's arbitrary choices. But if the model was embedded into a recommender system, an expert system or any other computer-based device for ballet training, net modifications could respond to well-defined algorithms and adaptive strategies. For example, by considering the duration of audio fragments (values known in advance), computing breaks between exercises (an aspect customizable by the trainee), and taking into account model's adjustment strategies (teacher's expertise embedded into the software environment), the system could produce a complete ballet class of the desired duration. Similarly, by associating an average calorie consumption to each audio fragment, it would be possible to prepare a lesson that involves a given amount of energy consumption. Once again, modifications could occur even in real time, for example on the base of heart rate or other parameters read by activity trackers and fitness bands.
In any case, the role of an expert teacher is fundamental both to design the base model and to plan modification strategies. For example: Which types of exercise can you

suppress if needed? Which pieces can you consider as interchangeable? How can you shorten a lesson? And so on.

7 Conclusion

In this paper we have outlined an approach for the formalization of a ballet class through Petri nets. The intrinsic problems due to the lack of teaching and training standards, the variable structure of a lesson, the different goals each lesson can present have been addressed on one side by exploiting the know-how of skilled ballet teachers and the available literature, and on the other by explaining how the model can be provided with a certain degree of flexibility.

References

[1] A. Vaganova, *Basic Principles of Classical Ballet*. Dover Publications, Inc., 1969.

[2] J. Gregory, *The Legat Saga: Nicolai Gustavovitch Legat, 1869-1937*. Princeton Book Co Pub, 1994.

[3] C. Beaumont, and S. Idzikowski. *The Cecchetti method of classical ballet: Theory and technique*. Courier Corporation, 2003.

[4] Royal Academy of Dance. *The Foundations of Classical Ballet Technique*. Royal Academy of Dance Enterprises Ltd., 1997.

[5] Royal Academy of Dance. *The Progressions of Classical Ballet Technique*. Royal Academy of Dance Enterprises Ltd., 2012.

[6] T. Scholl, *From Petipa to Balanchine: Classical Revival and the Modernisation of Ballet*. Routledge, 2003.

[7] G. Grant, *Technical manual and dictionary of classical ballet*. Dover Publications, Inc., 1982.

[8] G. W. Warren, *Classical Ballet Technique*. University Press of Florida, 1989.

[9] R. Ryman, *Dictionary of Classical Ballet Terminology*. Dover Publications, Inc., 1998.

[10] C. A. Petri, "Introduction to general net theory," *Net theory and applications*, pp. 1-19. Springer Berlin Heidelberg, 1980.

[11] J. Wang, *Timed Petri nets: Theory and application*. Springer Science & Business Media, 2012.

[12] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, Inc., 1994.

[13] A. Baratè, G. Haus, and L. A. Ludovico, *Real-time Music Composition through P-timed Petri Nets*, ICMC|SMC|2014 Proceedings, Athens 14-20 September 2014, pp. 408-415, 2014.

Petri nets are a mathematical modeling language for the description of distributed systems. They are particularly effective to provide a synthetic, multi-layer graphical representation of processes, unveiling aspects such as iterative structures and concurrency. This paper aims to apply Petri nets to the formalization of a typical lesson in classical ballet. After analyzing the basic principles and the commonly accepted rules to structure a ballet class, we will define a net model that can be conveniently modified by changing either its initial marking or the network topology in order to take