A Formal Method to Generate Games with Multi-level and Multi-user using Hierarchical Coloured Petri Nets

Vanessa Viana S. Carvalho
Universidade Federal do Ceará (UFC)
Joel Cruz Soares
Universidade Federal do Ceará (UFC)
Prof. Dr. José Marques Soares
Universidade Federal do Ceará (UFC)
Prof. Msc. Carlos Hairon R. Gonçalves
Instituto Federal do Ceará (IFCE)
Felipe Mota Barreto
Universidade Federal do Ceará (UFC)
Prof. Dr. Giovanni C. Barroso
Universidade Federal do Ceará (UFC)

Abstract

This paper presents a formal method to the generation of multi-level and multi-user games. Our approach models a multi-user game containing multiple navigation environments (multi-level) using Hierarchical Coloured Petri Nets, in which are specified all of the rules, properties and structures. The created model can be formally analysed allowing us to detect deadlocks or dead transitions (invalid paths) and others design problems. The formal analysis can be made using the CPN Tools. It is presented a tool, called *CPN Games*, and a new game in order to validate our method. This tool allows the user to create, fast and dynamically, simple conception games only using Hierarchical Coloured Petri Nets. Moreover, this tool is able to interpret and execute Hierarchical Coloured Petri Nets made in CPN Tools.

Keywords: Hierarchical Coloured Petri Nets, Games, CPN Tools

Author's Contact:

vanessaviana@great.ufc.br
hairon@ifce.edu.br
joelcruz007@gmail.com
felipemota0301@gmail.com
marques@ufc.br
gcb@fisica.ufc.br

1 Introduction

The difficulties inherent in the design of electronic games are the same that have been discussed in the context of software engineering since the early 1970s. Some of the problems found in the development of this particular type of software are discussed in [Petrillo et al. 2009].

According to [Blow 2004], programmers do not develop games as they used to do in the 90's because the process of a game development was funny and turned into a complex and difficult thing to implement (he shows an example of the increasing number of coding components). [Hamann 2003] says that the reason for that is because people tend to accumulate a lot of work to do in the deadline of the project (which he calls *Crunch Time*), and he also claims this is not a particularity of game projects.

Another related problem is the rapid obsolescence due to high recycling of the vanguard technologies. The lack of skilled labor for new technologies and their instability increase the challenges [Petrillo et al. 2009]. It concludes that many of the problems found could be minimized if the designers choose to use specialized tools for creating games.

Moreover, despite the large variety existing, the productivity tools to games do not include everything that a game designer needs [Blow 2004]. In general, they are slow to compile, debug and test, making many game elements have to be designed and built from scratch. So, it is understood that the development of tools that support the development of games is open to new investigations and propositions.

As a contribution to this area, in this paper is presented a tool that enables the construction of simple conception games using models in Hierarchical Coloured Petri Nets. It is provided to the designer an environment in which is possible to create new multi-level and multi-user games by modelling. The file coding a model of Coloured Petri Net (CPN) is processed, and then is generated an instance of a game, without any coding after the model creation. The locations, paths and rules are established for the game derived from the CPN, which is modelled using CPN Tools[1]. The XML code from a CPN, that is made by CPN Tools, is interpreted by a Java application that builds the game graphical interface to the user. The user interaction with the graphical interface is obtained following the rules established during the modelling using the CPN formalisms. Besides the possibility of formal representation of the dynamic and static aspects of a game, the CPN allows the verification and validation of the designed model, using the analysis simulation tools available on CPN Tools.

This paper is organized as follows: in Section 2, it is discussed the use of Petri Nets in the game development and its correlated works. In Section 3, it is exposed the rules to the game modelling in our context. In Section 4, it is presented the simulation and analysis by a CPN model. In Section 5, it is described an example of a game modelled using the developed tool and how the created model turned into the game. Finally, in Section 6 is presented the final considerations.

2 Using Petri Nets to Support the Devlopment of Electronic Games

The Petri Nets (PN) is a mathematical tool to model systems that have a graphic representation which consists of arcs and nodes [Murata 1989]. Logical and conceptual errors can be detected using formal analysis of the structural and behavioural properties, even in the early steps of the PN modelling. A PN is an abstraction of an original system and can be formally simulated and analysed, allowing the validations and verification of the structural and behavioural aspects of the net before the implementation of the real solution [Marranghello 2005].

The PN structure is represented by a set of places, a set of transitions, an input function and an output function. Graphically, the basic structure of a PN are places, arcs and transitions, that are represented by ellipses, directional arcs and rectangles, respectively (Figure 1). The dynamic of a PN is given by the firing of a transition that removes tokens of the input place and put the tokens in the corresponding output place. The number of tokens removed or put depends of the arc weight.

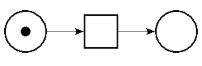

Figure 1: *Basic structure of a Petri Net*

The representation power of a PN is increased with the high level programming language features, that is called High Level Petri Nets, in which includes the Coloured Petri Nets (CPN) [Jensen and Kristensen 2009]. The CPN allows the representation of many abstractions, one of them is a entire net in a single transition, which is called substitution transition.

Both ordinary PN and High Level PN have been used to support the development of electronic games and their multiple features.

[1]Tool avaiable in http://cpntools.org/

In [Lee and Cho 2011] is proposed a method based in PN to generate little missions or tasks plots to the main characters. The same authors present in [Lee and Cho 2012] a tool, called *PlotWizard*, that gives a support to the development of the game plot. The formalism based in PN supports only how the characters interact in the game, not interfering in the graphic properties and the game controlling. The game states mapped to PN are static and quality restricted, which decreases the abstraction power of the PN.

In [Araújo and Roque 2009] is presented the modelling of a game flow in PN, as an alternative more robust to UML and Flowchat modelling. The authors use as a case study the navigation and mapping of the coast and sea routes. However, in this approach, there is no generation of automatic coding and of the nets that represent it.

The work of [Santos and Roque 2010] presents an automatic implementation of PN models in order to perform dynamic changes in the game to stimulate the player. The control of each player is accomplished through a PN and artificial intelligence. PN is adjusted according to the agent's behavior in each session. For agents that are not controlled by the player, i.e. for the Non-Player Characters (NPCs), the system checks which of the PN arches had more token passes. If the agent succeeds, the section arches over walkways are kept and the other arcs are replaced by arcs chosen in some sort of lottery or are removed. This study presents the modelling of the behaviour of the elements of the game by RP, but does not allow the creation or automatic generation of game structure.

Two limitations can be enumarated in the works mentioned above. First, they do not allow representations of multi-level games through hierarchical Petri nets. To the best of our knowledge, there is no studies to allow the generation of games in an automated manner from models designed in PN.

Among the several tools that allow modelling by Petri Nets, it has been adopted in this work the CPN Tools. The integration of this tool allows the modelling of the structure and behavior of the game, it is possible to perform simulations and formal analysis of the network. In the abstraction used to build games using CPN, the records deposited in places represent the game characters and the places represent the regions where the characters can move. Transitions constitute the ways on which a character can move from one place to another, provided they meet the rules of displacement specified in the modelling. Transitions substitutions are used to represent the level crossings. Using these rules of representation, you can use the CPN model, encoded in XML by CPN Tools, to directly generate the game. Therefore, it is only necessary to configure the media that are associated with the network elements during the construction of its graphical interface. Further details of the use of Colored Petri Nets for the automatic generation of games are described in next Sections.

3 Rules to Game Modelling

The characters are defined by the tokens configuration of a place and represent the sprites (characters in movement). Each token represents one sprite. The sprites are associated with a graphic representation through the numeric identifiers set in the tokens. The Figure 2 shows an example when two sprites are represented with the values 1 and 2, identified with the marking 1‘1 ++ 1‘2 (which means that there is a token with the value 1 and other with value 2). The sprites can move through the planets (abstraction to the places in this game) according to the rules modelled in the CPN. For example, the CPN of the Figure 2 specifies that the sprites can move in the paths obeying the arrows directions.

The hierarchies from a CPN are specified by substitution transitions. These transitions are represented by rectangles with double edge, as the transition *t5* of the Figure 2. Substitution transitions represents an abstraction level to a sub-net. In the Figure 2, the transition *t5* is the abstraction of the sub-net of the Figure 3.

In this paper is adopted the substitution transitions to configure multiple levels in the game. Thus, when a sprite moves to a substitution transition, this sprite is passed to a different domain. This new domain has a structure which consists of new places and transitions in which are associated distinct rules from the superior net. In the graphical user interface side, when a sprite collides in a substitution transition, means that the phase of the game will change. This change of phase automatically change the game scenario image.

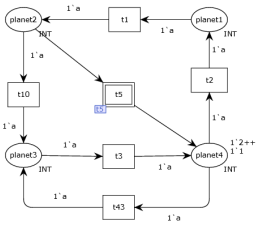

Figure 2: *Example of a Hierarchical Coloured Petri Net*

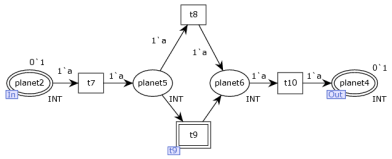

Figure 3: *Sub-net t5 of the transition t5 from the net of the Figure 2*

4 Simulation and Analysis by the CPN model

The formal analysis of a CPN model allows the validation through the structural and behavioural properties. The analysis realized with the CPN Tools can be made by validation (i.e. interactive simulation), by verification (i.e. state space analysis), and by performance analysis (i.e. monitors simulation) [van der Aalst and Stahl 2011]. Furthermore, the analysis report generated by CPN Tools allows the verification of the standard properties of a PN. These report properties can describe if the net is bounded, if the net has any dead marks (deadlock), if all the transitions are enabled from any mark (liveness, deadlock free), if all marking is reachable from any other marking (home-marking), and if from any marking is possible to return to the home marking (reversible net).

The CPN Tools generates an analysis report, in which is possible to identify some important properties of the game. For example, it is possible to check the existence of deadlock in a game plot. Using the example of the Figure 2, if the transition *t1* is removed, the elements that arrive in the place planet1 will not come back again, which means a deadlock (Figure 4).

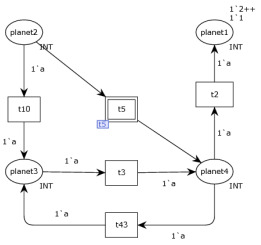

Figure 4: *Petri Net from the Figure 2, without the transition t1*

As the other properties, the deadlock can be viewed by simulation and can be verified by state space analysis of the CPN using the analysis report generated in the CPN Tools. For the example in the Figure 4, the CPN Tools generates an analysis report that is partially shown in the Figure 5. The generated report considers the initial mark established in the model represented in the Figure 2 (1‘2 ++ 1‘1 in the place *planet4*). In this example, the report shows a deadlock in the marking ”27” (1‘2 ++ 1‘1 in the place *planet1* in the Figure 4). It means that, if the sprites (1‘2 ++ 1‘1) reach the place *planet1*, they could never reach other places again.

The Figure 6 shows a state space generated by CPN Tools from the net of the Figure 4. In this state space is possible to identify the deadlock state which is highlighted with blue color.

```
Home Properties
------------------------------------------------

 Home Markings
    [27]

Liveness Properties
------------------------------------------------

 Dead Markings
    [27]

 Dead Transition Instances
    GameNet'tl0 1
    t5'tl0 1
    t5'tl7 1
    t5'tl8 1
    t9'tl1 1
    t9'tl2 1
    t9'tl3 1
    t9'tl4 1

 Live Transition Instances
    None
```

Figure 5: *Part of the report generated by CPN Tools*

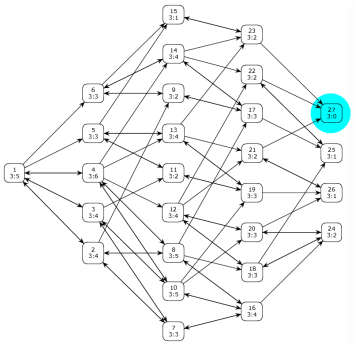

Figure 6: *State Space graph generated by CPN Tools from the net of the Figure 4*

5 Multi-level and Multi-user Game Project

The possibility to use hierarchical CPN can help the designer with more options to build the game model. The hierarchy can be modelled so that can represent many phases or even a passage to a fulfilment of an extra task.

The Figure 7 shows an interface of the game generated from the CPN of the Figure 2. The model defines the game structure and rules to the sprite movements through the planets. For the examples of the Figure 7, Figure 9, Figure 13 and the Figure 11, it has been used used the images from the free repository OpenGameArt[2].

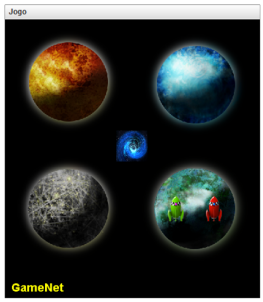

Figure 7: *Generated game from the net of the Figure 2*

The sub-nets are represented in a superior level of a net by substitution transitions. The sprites can move through the multiple levels of the game using these transitions. In the game example of the Figure 7, these transitions are represented by a black hole image. The game unfolds while the sprites move through the places of the net by the user interaction with the graphical interface. Furthermore, when a sprite collides with a substitution transition, this sprite migrates to a different game level, then it is displayed the corresponding layout to that level. Rules can be established by modelling, so only the sprites that had achieved certain properties could change the level.

The Figure 9 presents a *CPN Games* vision of two players in the same machine and in different levels of the game. On the left side, there is a sprite that just came through the black hole of the Figure 7 to the following level, that is structured according to the model presented in the net *t5* of the Figure 3. On the right side, there is a sprite that went through the black hole of the *t5* net and arrived on the *t9* net, whose model is represented on the Figure 8.

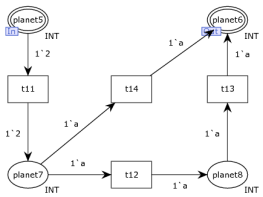

Figure 8: *Sub-net t9 of the transition t9 from the net of the Figure 3*

Figure 9: *Sprites in two different levels of a game, based on the sub-nets from the Figure 3 and Figure 4*

5.1 Changing of the Rules and Structure of the Game by Changing the Model

The model whose the rules and structures were defined can be changed only using CPN Tools. The new model can be loaded and interpreted by *CPN Games*. To illustrate this procedure, when the place *planet3* is removed from the net of the Figure 2, it is obtained the net of the Figure 10.

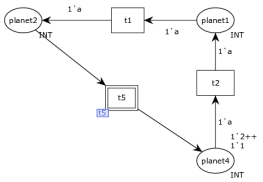

Figure 10: *Net from the Figure 2 without the place planet3.*

Then, the modified model is loaded in the *CPN Games* tool and the image that represents the *planet3* will not be shown anymore, as seen in the Figure 11. This procedure can be made without any code programming.

After realize the modelling or change any existing model, it must re-execute all the analysis and simulation procedures using the CPN Tools in order to validate the new model.

5.2 CPN XML Parsing

The CPN interpretation is realized by a model created in the CPN Tools. The generated file by this tool has the extension .cpn, coded following the XML syntactic rules. This file can be analysed by any syntactic analyser (parser) developed to this language.

The fundamental structures are represented by the elements (tags) *place*, *trans* and *arc* in the generated code of the CPN Tools. In the *CPN Games*, these structures are implemented in the classes *Place*, *Transition* and *Arc*, respectively. So, in the analysis process, all the instances of these elements are created and the properties are initiated according to the attributes extracted from the model.

To identify a sub-net and set different sub-levels of the game, it is necessary identify the occurrences of the substitution transitions. This is possible when the element <trans> has a <subst> child. Besides, an attribute subpage from the element <subst> has the sub-net identifier that substitutes the transition.

[2]Images obtained at http://opengameart.org/

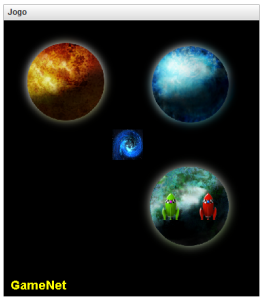

Figure 11: *CPN Games of the resulting net of the Figure 10.*

In the *CPN Games*, the *Reader* class is responsible to the syntactic analysis of the XML code and to the instantiation of all the elements that composes the modelled net to the game.

5.3 Graphical representation of the elements of a CPN model

Besides the building of the model with CPN Tools, in order that a game can be effectively executed by *CPN Games*, it is necessary associate each visible element to a file that contains the representative image. A configuration file with the same name of the file that contains the CPN code must be created with the extension *.properties* (see Figure 12). In this file there must have the relative path of all the images used in the game. It can have also the relative paths of the sounds that will be activated in each fired transition and the representation images of the substitution transitions. After the configuration of the *.properties* file, the game can be loaded and initiated.

rede5.properties

```
1 New Page.musica = resource_rede5/Adventure_Island_Level_1.mp3
2 New Page.transicao_subs_vertical = resource_rede5/buraco_negro_vertical.png
3 New Page.transicao_subs_horizontal = resource_rede5/buraco_negro_horizontal.png
4 t5.musica = resource_rede5/Adventure_Island_Level_2.mp3
5 t9.musica = resource_rede5/1943-lev1.mp3
6 imagens = resource_rede5/images
```

Figure 12: *A file example that defines the graphic and sound properties of the game*

5.4 User Interface

To the desktop environment, it was implemented an interface that contains all the necessary elements to the graphical representation of the game and the sprites movements by user interaction.

Besides the desktop implementation, it was developed an interface to the Android platform. Some aspects about mobile platforms must deserve some attention, for instance, the display size. In the desktop version, the camera is fixed and the sprite moves in the scenario that is completely rendered into the graphical interface. In this version, only a scenario part is rendered, and the camera must follow the sprite during its movement, as shown in Figure 13. Other aspect that deserve some attention is that in the mobile version the user can make moves with only one sprite.

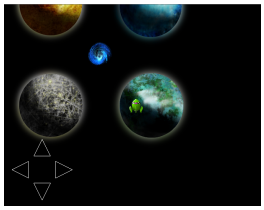

Figure 13: *Image of the game in a mobile device*

It is important to mention the lack of link between the model, that is independent of platforms, and the interface implementation, that depends of a platform. In both cases, different models generate different games, without any code implementation, making easier the portability of the games in different devices.

Furthermore, the *CPN Games* allows the gamers play the same game in local net. In the future, this communication can be extended to long distance networks. The distribution model used in this tool is the Server-Client, in which a server centralizes the global state of the game. This server can be executed in a mobile device or in a desktop. A localization service can identify the servers in a local network through broadcast messages.

6 Conclusion

This paper presents a method that allows the development of simple conception games with multiple levels and multiple users only with Coloured Petri Net modelling. The model allows the designers to perform tests and simulations before the software implementation. To validate this method, it was developed a tool, called *CPN Games*. This tool allows the user develop simple conception games, like the game described in this paper. The *CPN Games* was implemented in two versions of graphical user interface. One is the desktop version, where the user interaction is made by the keyboard, and the other version is to mobile (Android), where the user interaction can be made by touchscreen or by accelerometer. Regardless of the kind of user interface, it is not necessary any additional code programming after the building and analysis of the model. This method allows people with little programming knowledge to develop their own games. With the tools described in this paper, it is possible easily develop a range of simple conception games with low cost and high productive.

References

ARAÚJO, M., AND ROQUE, L. 2009. Modeling games with petri nets. *Breaking New Ground: Innovation in Games, Play, Practice and Theory. DIGRA2009. Londres, Royaume Uni*.

BLOW, J. 2004. Game development: Harder than you think. *Queue 1*, 10, 28.

HAMANN, W., 2003. Goodbye postmortems, hello critical stage analysis.

JENSEN, K., AND KRISTENSEN, L. M. 2009. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer.

LEE, Y.-S., AND CHO, S.-B. 2011. Context-aware petri net for dynamic procedural content generation in role-playing game. *Computational Intelligence Magazine, IEEE 6*, 2, 16–25.

LEE, Y.-S., AND CHO, S.-B. 2012. Dynamic quest plot generation using petri net planning. In *Proceedings of the Workshop at SIGGRAPH Asia*, ACM, 47–52.

MARRANGHELLO, N. 2005. Redes de petri: Conceitos e aplicaçoes. *São Paulo: DCCE/IBILCE/UNESP*.

MURATA, T. 1989. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE 77*, 4, 541–580.

PETRILLO, F., PIMENTA, M., TRINDADE, F., AND DIETRICH, C. 2009. What went wrong? a survey of problems in game development. *Computers in Entertainment (CIE) 7*, 1, 13.

SANTOS, S. D., AND ROQUE, L. G. 2010. Ensaio de reescrita de comportamentos em videojogos com base no ajuste e computao de modelos de petri net. *IX SBGames*.

VAN DER AALST, W., AND STAHL, C. 2011. *Modeling Business Processes: A Petri Net-Oriented Approach*. MIT press.