Using Timed and Coloured Petri nets for Modeling, Simulation, and Analysis of Integration Solutions

Francine Freddo, Sandro Sawicki, Rafael Z. Frantz, Fabricia Roos-Frantz

Department of Exact Sciences and Engineering, Unijuí University.
Rua do Comércio, 3000. Ijuí 98700-000, RS. Brazil.

Abstract

Enterprise application integration (EAI) is a research field that seeks to develop methodologies, techniques and tools for the design and development of integration solutions. The software ecosystems of companies are composed of several heterogeneous applications, usually obtained from third parties or developed internally and adapted for their business processes. In this context, the main challenge faced by companies is that most of their applications were not designed considering integration with other applications. In this work, we propose to develop a mathematical model to simulate integration solutions in the design phase, i.e., before the implementation and testing stages. The aim of this simulation is to analyse the behaviour and identify the possible performance bottlenecks of the application integration solutions. In this way, it is possible to identify problems prior to the implementation and testing stages, in order to reduce costs, risks and time. A conceptual model of the integration solution designed using Guaraná technology will be translated into a mathematical model of the simulation using timed and coloured Petri nets. This equivalence is generic and can be translated into any application integration technology. A real-life problem in the area of marketing was used as a case study. In addition, simulation scenarios similar to an actual operating process were defined. The computational simulations were performed using CPN Tools, which allows the use of coloured and timed Petri nets. Finally, verification of the equivalence of the formal simulation

*Corresponding author

Email addresses: francinefreddo@unipampa.edu.br (Francine Freddo), sawicki@unijui.edu.br (Sandro Sawicki), rzfrantz@unijui.edu.br (Rafael Z. Frantz), fabriciar@unijui.edu.br (Fabricia Roos-Frantz)

model using timed and coloured Petri nets and the conceptual model, was performed using formal verification techniques widely found in the literature.

Keywords: Enterprise Application Integration, Coloured Petri nets, Timed Petri nets, Discrete-event Simulation.

1. Introduction

Usually, companies need to use their software ecosystems (Messerschmitt and Szyperski, 2003) to support and improve their business processes. Ecosystems are composed of many applications, usually designed without taking into account their possible integration. Within the area of software engineering, the field of study known as enterprise application integration (Hohpe and Woolf, 2004) seeks to provide methodologies, techniques and tools for designing and implementing integration solutions. In general, an integration solution aims to orchestrate a series of applications, in order to keep them synchronized or to provide new functionalities that can be formed from existing ones. An integration solution is created by processes which contain integration logic and communication ports that connect process-ecosystem applications or the integration solution.

Generally, the development of an integration solution is organized in traditional phases, such as analysis, design, implementation and testing. Although an integration solution follows implementation standards and techniques, errors may occur in its structure. In addition, it may present performance bottlenecks in its components, in situations of high computational demand. The analysis of integration solutions to predict behaviour and to find possible performance bottlenecks is an important activity for increasing the quality of integration solutions. The approach usually adopted by software engineers is to build and execute the integration solution. However, developing the solution involves costs and risks that can often be high.

The integration solution is a piece of software that is composed of one orchestration process that exogenously coordinates the heterogeneous applications involved in the software ecosystem. An application integration solution can be classified as stochastic, dynamic and discrete. Thus, it is possible to use computational and mathematical models, as well as discrete-event simulation techniques, while still in the design phase. In this way, it is possible to find the performance bottlenecks that may arise when an application integration solution encounters a critical operating scenario, before its implementation. Rezai et al. (1995) indicate that Petri nets have been proven to be an excellent modelling and analysis tool for discrete events or asynchronous systems. Alla and Ghomri (2012) report that the basic concept of Petri nets and the classes of derived models can be used for dynamic modelling of systems. The authors also report that autonomous Petri nets enable any kind of discrete event whatsoever to be modelled.

This work aims to analyse the behaviour and identify the possible performance bottlenecks of the application integration solution, through the analysis of its slots in the design phase. The variables analysed are: (i) the time permanence of messages and (ii) the number of messages accumulated. The conceptual model of the integration solution will be translated into a mathematical simulation model using Petri nets. Our work translates a static conceptual model into a simulation model. This equivalence is generic and can be translated into any application integration technology. A real-life problem in the area of marketing was used as a case study. In addition, simulation scenarios similar to those of an actual operating process were defined. The experimental results were collected after 25 executions for each scenario, following the law of large numbers (Grinstead and Snell, 2012). The computational simulations were performed using CPN Tools, which allows the use of coloured and timed Petri nets. Finally, the verification of the equivalence of the formal simulation model with Petri nets and the conceptual model, was performed using formal verification techniques widely found in the literature. The results obtained in the simulation were also used by the verification techniques to evaluate the accuracy of the model created.

The rest of this article is organized as follows. Section 2 provides an overview of the stochastic Petri nets, coloured nets and timed nets used to develop the formal simulation model, section 3 discusses related work that also uses timed and coloured Petri nets in discrete-event simulations to analyse systems, section 4 presents the study's proposal of translating integration solution models into timed and coloured Petri nets and section 5 discusses the analysis results obtained from the simulation of the formal model and its verification. Finally, section 6 presents the main conclusions.

2. Background

In this section, we provide a brief overview of the stochastic Petri nets, timed Petri nets and coloured Petri nets that were used as mathematical techniques to develop the formal simulation model. The enterprise application integration technology from Guaraná, used as a case study, is also discussed in this section.

2.1. Stochastic Petri Nets

Petri nets were invented in 1962 by Carl Adam Petri as a mathematical formalism to describe concurrence and dynamic synchronization in dis-

tributed systems. Petri nets are graphs composed of nodes that are places and transitions and arcs. Places are represented by circles and transitions by rectangles. Nodes are connected by arcs. Places can contain tokens, represented by points within the place, and the number of tokens in one place is called a marking. An input arc connects a place to a transition, and an output arc connects a transition to a place. When a transition is fired, the tokens in the places connected by the input arcs are removed and then added to the places connected by the output arcs. A transition is active and can be fired if the number of tokens in its places satisfies the weight determined by the input arcs. The weight of the output arcs is not necessarily the same as the weight of the input arcs. When a transition is triggered, the Petri net changes its state. This concept is called the "set of markings", with markings representing the number of tokens in each place after a finite sequence of transitions has been fired.

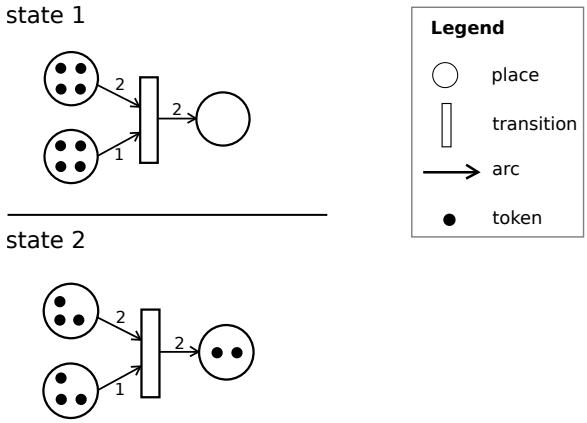

Figure 1: Example of firing a transition

Petri nets are defined as bipartite graphs (or bigraphs), because they present two types of nodes, places and transitions. Mathematically, Petri nets are defined as follows.

Definition: A Petri net is a sextuple:

$$RP = (P, T, Ar, K, W, M_0), \text{ where:} \tag{1}$$

- $P = p_1, p_2, ...p_m$ is the finite set of places,

- $T = t_1, t_2, ...t_n$ is the finite set of transitions,
- $A_r \subseteq (P \times T) \cup (T \times P)$ is the set of arcs,
- $K : P \to \mathbb{N} \cup \{\infty\}$ is the capacity function,
- $W : A_r \to \mathbb{N}^+$ is the weighting function,
- $I : T \to (R \cup \{0\})$, where the minimum and maximum delay is represented by $(d_{\min}, d_{\max}), d_{\max} \geq d_{\min}$,
- $M_0 \to N$ is the initial function tag, where $\forall p \in P : M_0(p) \leq K(p)$.

According to the Petri net definitions, the following conditions must be satisfied.

$$P \cap T = \{\} \tag{2}$$

and

$$P \cup T \neq \{\} \tag{3}$$

The equation in 1 represents a stochastic Petri net. According to the condition of Equation 2, the places and the transitions are distinct nodes, thus justifying the bipartite term. According to the condition of Equation 3, it is understood that in a Petri net, there is at least one place or transition. A Petri net is formed by a structure composed of places and transitions connected by arcs and an initial marking.

2.2. Coloured Petri Nets

Coloured Petri nets aim to reduce the size of the model, and they have considerable importance in the modelling of complex systems. This type of Petri net allows the tokens to be individualized by assigning colours to them, so that different processes or resources can be represented on the same network. Colours do not just mean patterns; they can represent complex data types. The colour nomenclature is used only to distinguish between tokens. Figure 2 presents a coloured network with the original representation,

where colours are used for the tokens and the arcs are labelled with the colours (x, y, w, z). Though simple, the original coloured Petri net provides mechanisms for making a deterministic choice.

A qualified choice leads to an evolution of the coloured Petri net, but this is aided by non-deterministic representations. Coloured Petri nets are composed of the following parts: structure, declarations and inscriptions. The structure is a graph driven by two vertices (places and transitions), which can store marks of different types in each place and represent values associated with more complex data types. Declarations are the specifications of the colour sets and variables, and the inscriptions change according to the component of the network. Places have three types of entries: names, a set of colours and an initialization expression. Transitions have two types of inscriptions: names and saved expressions. Arcs have only one type, which is given by the expression. To differentiate entries, names are written with normal letters, colours are written in italics, initialization expressions are underlined and storage expressions are enclosed in square brackets, as shown in Figure 2.

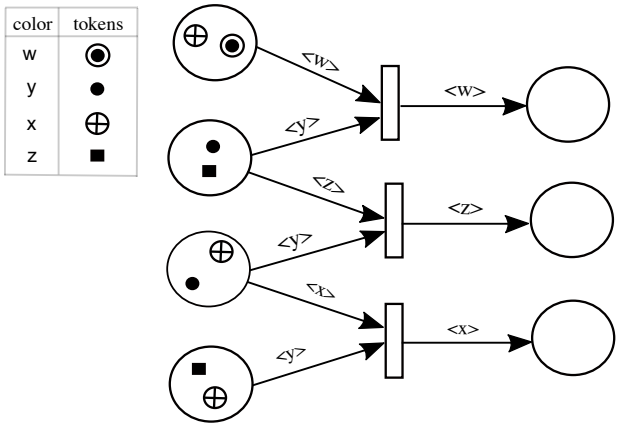

Figure 2: Coloured Petri net elements

2.3. Timed Petri Nets

Timed Petri nets have emerged for modelling dynamic systems. There is a logic associated with their transitions, such as the enable time, firing time and relative firing frequency. Enable time refers to the time for which the transition must remain enabled before firing. Firing time is the time the

transition needs in order to fire. This is used to represent the time consumed in the execution of events in a modelled system. Firing time can be a constant or a function. If it is a constant, it can be zero or a positive integer value. If it is a function, it can assume random values according to the input distribution chosen. The relative firing frequency applies to the transitions belonging to a Petri net conflict set, i.e., situations in which more than one transition is able to fire and there is a need for choice. The structure of a timed Petri net is the same as that of a Petri net. The real change is that tokens, places and transitions are associated with time values. If RdPT is a timed Petri net, then:

$$RdPT = (P, T, A_r, K, W, I, M_0) \tag{4}$$

An $RdPT$ (Equation 4) is made up of m places, n transitions, A arc sets, K functions for maximum token capacity in places and W weight or arc weight functions. These weights can be constant or a function I of time over a transition T_j, which takes a time value belonging to a set T for the two coordinates of minimum delay and maximum delay and an initial marking function $M0$.

A transition t_j is fired if it is enabled according to the following rules.

$$\forall p_i \in P : M[t_j > M(p_i) \geq I(p_i, t_j)] \tag{5}$$

That is, for every place p_i belonging to the set of places P, the place marking p_i must be greater than or equal to the weight of the arc connecting the place p_i to the transition t_j. During the interval $I(t_j) = (dj_{\min}, dj_{\max})$, a t_j will only be fired if $t(t_j) \geq dj_{\min}$, that is, if the transition time t_j is greater than or equal to the time of its minimum delay (transition rules of Equation 5).

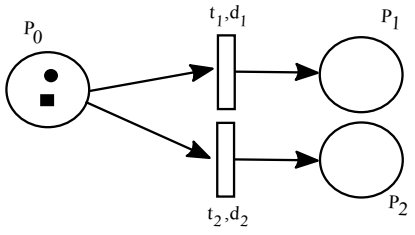

Figure 3: Semantics of firing in a deterministic timed Petri net

An example of a deterministic timed Petri net is shown in Figure 3, where transitions t_1 and t_2 have different associated times, which means that one transition will be fired before the other. Thus, assuming that $d_1 < d_2$, then the token will arrive first in place P_1. In a deterministic way, it is possible to establish the order in which the events must occur.

2.4. Enterprise Application Integration: Guaraná Technology

Guaraná technology provides a domain-specific language that enables the design of integration solutions with a high level of abstraction using concrete graphical syntax and intuitive modelling concepts. This modelling language is based on the integration patterns documented by Hohpe and Woolf (2004).

The conceptual models designed using the Guaraná technology are platform-independent. Thus, software engineers do not need specific expertise in low-level integration technologies in order to develop integration solutions. This feature allows engineers to focus their efforts on designing models that address the problem.

Conceptual models are developed using the Guaraná technology in a graphic language. The transformation of the models into an executable code is obtained through model-driven engineering. In this way, models designed using the technology can be reused, to automatically generate integration solutions to be executed in different technologies. A schema for the conceptual models using the Guaraná technology is shown in Figure 4. In the Guaraná technology, tasks are classified according to their semantics.

- **Routing tasks:** this class of tasks does not change the status of messages; it only forwards them through a process.
- **Modifying tasks:** modifying tasks add or remove message data without changing the schema. An example of this functionality is the task that adds data to the correlated message.
- **Transforming tasks:** transforming tasks translate one or more messages into a new message with a different schema.
- **Stream dealer tasks:** these are tasks that work with a stream of bytes and help to compress/decompress, encrypt/decrypt or encode/decode messages.
- **Mapping tasks:** mapping tasks change the format of messages that are processed, for example, from a byte stream to an XML document.

- **Communicator tasks:** communicator tasks are used on ports to interact with communication components, often called adapters.

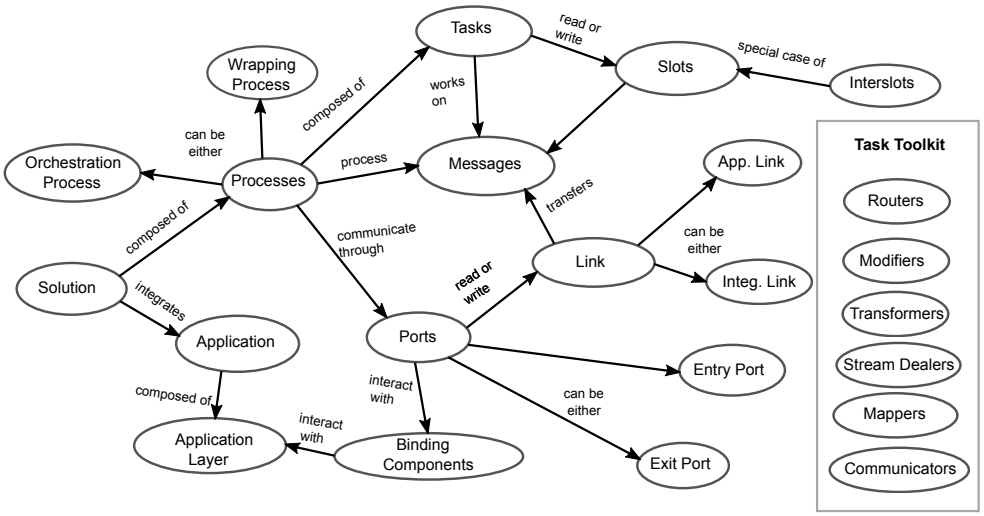

Figure 4: Conceptual model of Guaraná technology

Guaraná technology provides an implementation using a task-based runtime engine, which differentiates it from other integration technologies that use a process-based execution model. In the task-based model, threads are allocated at the task level, allowing parallel processing of different messages by the same process. In the process-based model, the threads are allocated to the process; thus, when message processing begins, the other threads are blocked until the process is finished.

3. Related Work

Many different types of problems that use Petri nets as a simulation model have been proposed in the literature. Some approaches use timed and coloured Petri nets to model automation processes and describe the concurrence and dynamic synchronization in distributed systems.

Taghinezhad-Niar et al. (2017) modelled an algorithm using a coloured Petri net to perform task scheduling in federated cloud systems, in an efficient manner, considering the service level agreement (SLA) requirements of the users. Wang et al. (2015) proposed an approach to capture the interactions that occur between collaborating systems or components using

object-process methodology and coloured Petri nets, with the objective of creating an executable architecture model for a system of systems (SoS). Du et al. (2018) used hierarchical coloured Petri nets for modelling industrial design collaborative system workflow, providing a valid workflow model for the process management of industrial design collaborative systems. Kaur et al. (2017) proposed a novel scheme for efficient frequency support in a smart-grids environment by modelling a fleet of electric vehicles using coloured Petri nets. Hu et al. (2017) proposed a similarity computing method based on the Levenshtein distance and a Petri net logic, aiming to improve the efficiency of the similarity computation between a service request and the existing service processes in renting or recommending cloud-service processes. Liu et al. (2017) presented a novel coloured generalized stochastic Petri net model for IT infrastructures, which reflects the dynamic behaviour and service request processing procedure under the active-active mechanism. The experimental results of the simulations were obtained using CPN Tools.

Sun et al. (2018) proposed a novel shared control method based on fused fuzzy Petri nets for combining robot automatic control and brain-actuated control. The authors consider that the use of both fuzzy controls and Petri nets is robust and effective. Zhang et al. (2018) presented a method for diagnosing power-grid faults using intuitionistic fuzzy logic. For this purpose, the authors obtained a diagnostic model for power systems that could calculate the certainty and the uncertainty of electrical device fault events using intuitionistic fuzzy Petri nets. Li et al. (2018) reported that no efficient methods have been proposed for the liveness analysis of general unbounded Petri nets, except for some of their subclasses. In this sense, the authors present an effective method for comprehensively analysing the properties of general unbounded Petri nets using a lean reachability tree. Similarly, an extension to reconfigurable Petri nets was proposed by Tigane et al. (2017), in order to provide a suitable tool for the formal modelling and verification of reconfigurable systems. Arciniegas et al. (2017) presented the modelling of an automation process for an electric car production line using Petri nets and GRAFCET with a shared resource. The proposal uses three concurrent processes: fabrication of the mechanical structure, assembly of the electric motor and assembly of the car batteries. An et al. (2017) developed a model for a signal control system with transit priority using coloured Petri nets. According to the authors, the proposed model ensures that transits can pass through intersections with less delay or no delay.

Strzęciwilk et al. (2018) used simulation with Petri nets to evaluate and verify waiting times and traffic intensity, focusing on analysis of a priority queuing system supporting quality of service (QoS). Allani et al. (2018) proposed a a coloured Petri net combined with timed Petri nets to model workflow management systems based on documents with time constraints. Wang et al. (2018) presented a stochastic timed Petri net-based healthcare workflow and resource modelling technique. The authors proposed a simulation framework that supports the analysis of optimal provisioning of resources, ensuring timely care services.

Entezari-Maleki et al. (2017) proposed a model based on timed coloured Petri nets to evaluate the service composition in multi-cloud environments, minimizing the number of clouds involved in a service request. Shmeleva (2017) specified online statistical analysis algorithms using coloured Petri nets. The authors reported that the same approach can be implemented for other statistical moments and other networking technologies. Zeineb et al. (2016) proposed a formal modelling and validation approach based on coloured Petri nets for the development of a generic model representing the overall behaviour of the different components of a smart grid.

The current literature reports that Petri net analysis allows the evaluation of the structure and dynamic behaviour of the system modelled. The result of this evaluation may lead to improvements or changes in the system. Our work, however, proposes the behaviour analysis of, and the identification of possible performance deficiencies in, the integration solution based on the identification of the length of stay of messages in the system and on the message accumulation in the slots, considering different processing priorities, while still in the project phase, by means of the development of a formal simulation model using coloured and timed Petri nets.

4. Translating an Integration Solution model into Timed and Coloured Petri nets

In this work, we propose the translation of the Guaraná conceptual model into a timed and coloured Petri net model. This translation offers several advantages. The main benefit is the fact that timed and coloured Petri nets are very well-established mathematical models. In addition, timed and coloured Petri nets use simple components, and there are several tools available for Petri net simulation and analysis.

4.1. Case Study and Software Ecosystem

The problem of integration is a reality in the marketing area. Its role is to disseminate advertisements from various customers, using advertisement stations called local stations. The software ecosystem is composed of eight heterogeneous applications: social networks, video monitor, SM media, remote access, advertising station, hardware monitor, printer and reporting system.

The data flow starts with the creation of the advertisement to be disclosed. In the sequence, the file is received by the system and stored in a database. In this way, the publishing process begins. The company filters the database to find advertisements that have not been published, to be shared on social networks. To send the advertisement to the local station, the company uses remote access software. In this software, the advertisement is included in a folder mapped by a tool that manages the reproduction of videos. The hardware monitor has the function of monitoring the behaviour of the following properties: status of the network, printer, temperature and the hardware of the local station. The company monitors each of these pieces of software individually.

Figure 5 shows the integration flow of the conceptual model. The integration process begins at port P_0, consulting the database for files. The files are transformed into a message, which is sent to the task filter (T_0) using the slot S_0. The task replicator (T_1) distributes the messages to a task translator (T_2) and a task replicator (T_3). A copy of the message is sent to the task correlator (T_5). The task context content enricher (T_6) receives the message through slots S_8 and S_9. The task translator (T_7) translates the message into the advertising station application. The merge task (T_8) receives all inputs from the video monitor, hardware monitor and printer applications and forwards them to the task assembler (T_9). The task translator ($T_{10)}$ translates the message and sends it to port P_7 and to the reporting system application.

In order to solve the problem of integration of the different applications, a temporal analysis was performed using tokens with different processing priorities, to identify the behaviour of the following variables during the whole process, aiming to find performance bottlenecks: (i) the average time of tokens in the places and (ii) the maximum number of tokens stored in the places.

Using coloured and timed Petri nets it was possible to represent different tokens and also assign time to each of them. The simulation tool used was CPN Tools (for coloured Petri nets). This tool has its own programming

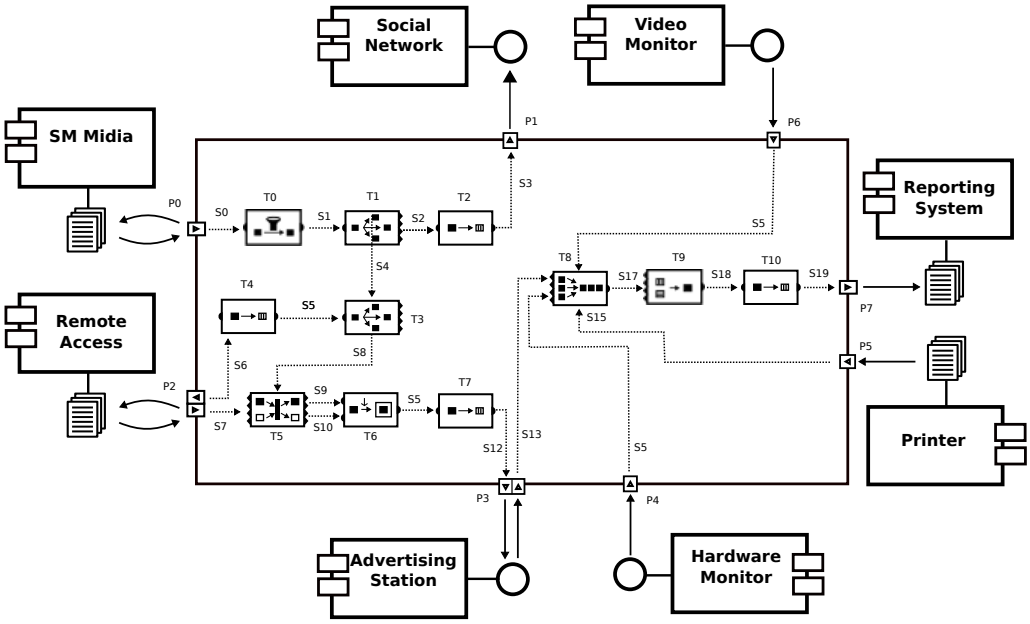

Figure 5: Conceptual model of the integration solution using the Guaraná technology

language for describing attributes of network elements. This language provides declarations of colour sets (types), variables, constants, functions and procedures.

In order to perform the temporal analysis of the developed computational model, three types of priorities were defined: token with high priority, token with medium priority and token with low priority. For each priority, a time was assigned: high priority (time 1), medium priority (time 2) and low priority (time 3). These priorities, in a real system, represent messages of different types and sizes.

We use the correspondence analysis of the models to map the Guaraná notation onto coloured and timed Petri nets. Three main steps were used. a) Determining the visions that will be the subject of analysis. This step focuses on particular aspects of the system. b) Determining the feasibility of a vision to be mapped to RdP. It is possible to determine the feasibility of the mapping process after the detailed analyses. c) Elaborating an equivalence table between the Guaraná notation and RdP. The mapping process can be seen as a mechanism of identifying correspondences between the components (diagrams) of the models (visions) and the features present in RdP. Figure 6 shows the equivalence between the main elements of the Guaraná and

the timed and coloured Petri nets technologies. The main elements of the Guaraná technology are tasks, slots and messages, which are analogous to the transitions, places and tokens respectively in the Petri net. In Petri nets, places and arcs have a similar function to slots. The arcs link a place to a transition and indicate the direction of flow of the tokens.

4.2. Mathematical Formulation of the Integration Problem

The formal representation of the simulation model is expressed by Equation 6, where P represents the set of places (Equation 7), T represents the set of transitions (Equation 8), I represents the mapping of the input arcs and O that of the output arcs (Equation 9 and Equation 10 respectively), A_r is the set of arcs (Equation 11, K is the capacity associated with each place (Equation 12), W is the weighting function (Equation 13) and M_o is the initial marking function (Equation 14). The tasks are represented by the following variables: A_1 (assembler), Z_1 (correlator), M_1 (merge), R_1, R_2 (replicator), T_1, T_2, T_3, T_4 (translation), F_1 (filter) and E_1 (content enricher). The heterogeneous applications are represented by the variables Sm, Vm, Sn, Rs, Hm and Pr.

If RdP is a Petri net, then:

$$RdP = \{P, T, W, I, A_r, O, M_o, K\} \tag{6}$$

$$\begin{aligned} P = \{&S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, \\ &S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16}, S_{17}, S_{18}, S_{19}\} \end{aligned} \tag{7}$$

$$\begin{aligned} T = \{&P_0, F_1, R_1, T_1, P_1, R_2, T_2, P_2, Z_1, E_1, T_3, \\ &P_3, P_4, P_5, P_6, M_1, A_1, T_4, P_7\} \end{aligned} \tag{8}$$

$$\begin{aligned} I = \{&(Sm, P_0), (S_0, F_1), (S_1, R_1), (S_2, T_1), (S_3, P_1), \\ &(S_4, R_2), (S_5, T_2), (S_6, P_2), (S_7, Z_1), (S_8, Z_1), \\ &(S_9, E_1), (S_{10}, E_1), (S_{11}, T_3), (S_{12}, P_3), (S_3, M_1), \\ &(S_{14}, M_1), (S_{15}, M_1), (S_{16}, M_1), (Vm, P_6), \\ &(S_{17}, A_1), (S_{18}, T_4), (S_{19}, P_7)\} \end{aligned} \tag{9}$$

$$
\begin{aligned}
O = \{&(P_0, S_0), (F_1, S_1), (R_1, S_2), (R_1, S_4), (T_1, S_3), \\
&(P_1, Sm), (R_2, S_5), (R_2, S_8), (T_2, S_6), (P_2, S_7), \\
&(Z_1, S_9), (Z_1, S_{10}), (E_1, S_{11}), (T_3, S_{12}), (P_3, S_{13}), \\
&(P_4, S_{14}), (P_5, S_{15}), (P_6, S_{16}), (M_1, S_{17}), (P_5, Pr) \\
&(A_1, S_{18}), (P_1, Sn), (P_4, Hm)(T_4, S_9), (P_7, Rs)\}
\end{aligned} \tag{10}
$$

$$
A_r \subseteq (P \times T) \cup (T \times P) \tag{11}
$$

$$
K : P \to \mathbb{N} \cup \{\infty\} \tag{12}
$$

$$
W : A_r \to \mathbb{N}^+ \tag{13}
$$

$$
\begin{aligned}
M_o &: P \to \mathbb{N} \\
&\text{such as:} \\
\forall p \in &: M_o(p) \leq K(p)
\end{aligned} \tag{14}
$$

The values of the transition firing rates are described according to Equation 15, considering the order of the set T. The term "firing rates" refers to firing percentages, but these are represented and implemented in simulators in decimal form. The filter task must eliminate repeated messages, so its firing rate is 95% ("0.05" in Equation 15). This means that 5% of messages will not go forward in the model (this rate is estimated). The rest of the tasks have no restrictions, i.e., all tokens have the same chance of being executed ("1" in Equation 15). The filter task is needed to eliminate the errors in messages.

$$
Td = \{1, 0.05, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\} \tag{15}
$$

4.3. Simulation Model Using Timed and Coloured Petri Nets

The transition T_0 was modelled as the port P_0, which is the beginning of the workflow through the SM media application (database of media). The transitions P_1 and P_7 represent the output ports for the social networks and reporting system applications respectively. The transitions P_4, P_5 and P_6 represent the input ports for the hardware monitor, printer and video monitor applications, and the transitions P_2 and P_3 are request ports for

the remote access and advertising station applications. The modifier, router and transformer task equivalences are represented by Figures 7, 8 and 9 respectively.

We translated the tasks in the Guaraná technology into timed and coloured Petri nets, while maintaining all their functionality. Figure 10 represents the simulation model developed in timed and coloured Petri nets. The input port $P0$ (Figure 11) has the function of inserting messages into the solution to be processed. Its representation in coloured and timed Petri nets is the transition P_0, which forwards the token to the place S_0. The semantic expression $t@ + tp(t)$ verifies how long the message (token) requires for processing. The output port (Figure 12 sends messages to the integrated applications. Its equivalent graph is a transition connected to a place in which the token is stored, and it indicates the end of the process.

Guaraná DLS	**Petri nets**
Task 	Transition 
Slot	Place 
Message 	Token 

Figure 6: Equivalence between Guaraná technology and Petri net components

Petri net	**Guaraná**	**Task name**
		Context Content Enricher

Figure 7: Modifier tasks

Petri net	Guaraná	Task name
		Correlator
		Merger
		Filter
		Replicator

Figure 8: Router tasks

Petri net	Guaraná	Task name
		Translator
		Assembler

Figure 9: Transformer tasks

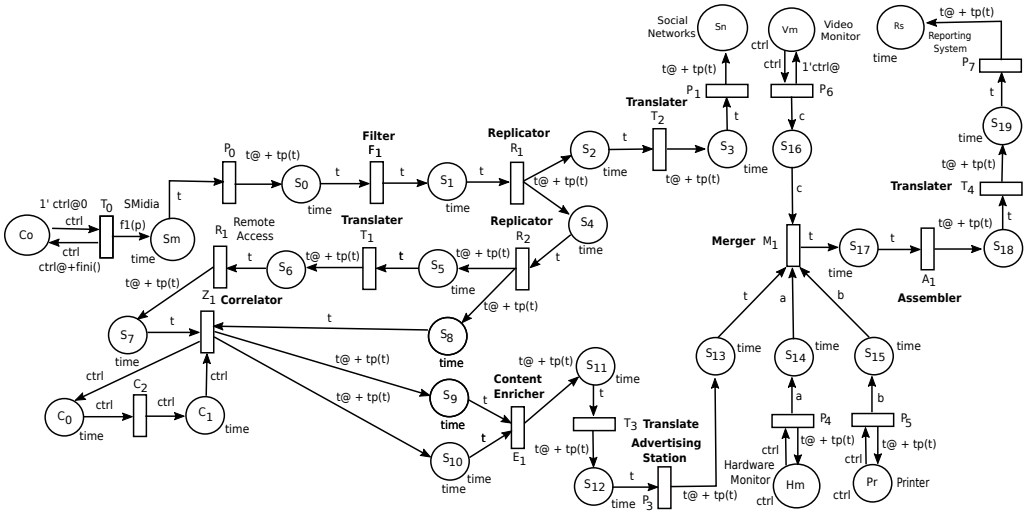

Figure 10: Proposed simulation model using timed and coloured Petri nets

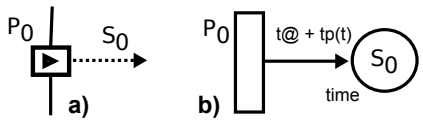

Figure 11: Entry port using **a:** Guaraná technology and **b:** timed and coloured Petri nets

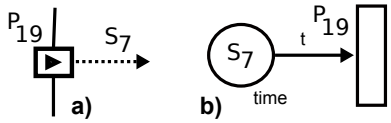

Figure 12: Exit port using **a:** Guaraná technology and **b:** timed and coloured Petri nets

The solicitor port requests information from an application. Its equivalent in timed and coloured Petri nets is shown in Figure 13, in which the transition P_2 is analogous to the port P_2, just as the places S_6 and S_7 are analogous to the slots S_6 and S_7. Remote access is the application in which the transition will execute the request through the t variable.

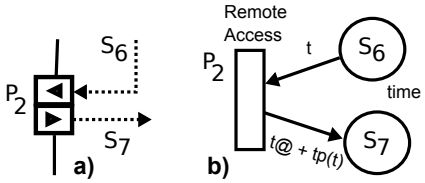

Figure 13: Solicitor and responder ports using **a)** Guaraná technology and **b:** timed and coloured Petri nets

The semantics of the filter task does not allow incomplete messages to enter the system. In timed and coloured Petri nets, this task is represented by the filter transition that contains the code *(if filter (f) then 1't else empty)) @tp (t)*, which allows representation of the function of discarding tokens, as described in Figure 14.

The translator task adapts the message to the application format. As the message is analogous to the token, this task is represented by an input place, a translator transition and an output place, expressing the input, processing and output of the message using the variable t, as shown in Figure 15.

The correlator task is a very important task within the message flow, because it can locate correlated messages that can be processed together. In Figure 16, this task is represented in the correlator transition, by places C_1

and C_2 and by transition C_2. The places C_1 and C_2 and the transition C_2 represent a delay in the token, which arrives in the correlator transition and waits until it finds its corresponding token. This correspondence is made through the time of each token: when this time is equal, the token is sent to the places S_9 and S_{10}, in order to maintain the functionality of this task.

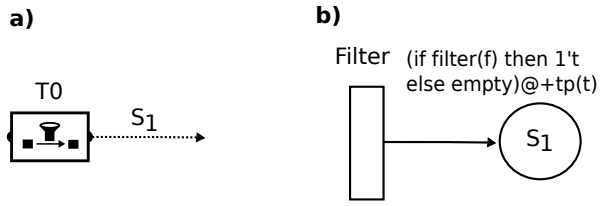

Figure 14: **a:** Filter task in Guaraná technology and **b:** filter task in timed and coloured Petri nets

Figure 17 shows the model of the replicator task. Its function is to make copies of the original message without changing its content and to forward them to the next step of the solution. This model is represented by an input place, a replicator transition and two output places. The merger task receives several messages from multiple slots and directs them to a single slot, as shown in Figure 18. This task takes messages from the S_{13}, S_{14}, S_{15} and S_{16} slots and sends them to the merger transition, which groups them all in the slot S_{17}. The different messages are represented by the variables a, b and c.

The context content enricher task receives correlated messages and combines them into a single message. For timed and coloured Petri nets, the graph that expresses this task is shown in Figure 19. This is the transition named context content enrich, which only fires when there is at least one token in each of the places S_9 and S_{10}. Figure 20 shows the assembler task, whose function is to construct a new message from two or more messages.

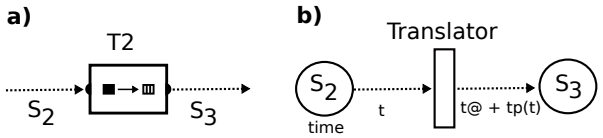

Figure 15: **a:** Translator task in Guaraná technology and **b:** translator task in timed and coloured Petri nets

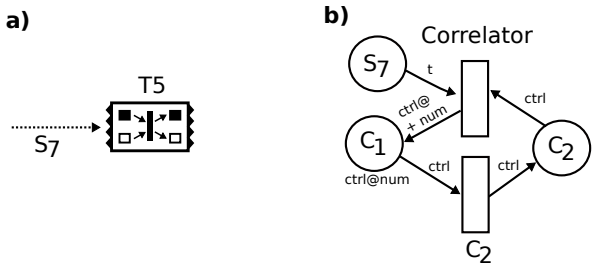

Figure 16: **a:** Correlator task in Guaraná technology and **b:** correlator task in timed and coloured Petri nets

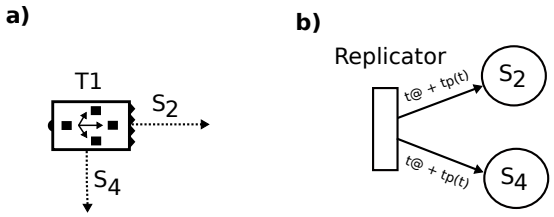

Figure 17: **a:** Replicator task in Guaraná technology and **b:** replicator task in timed and coloured Petri nets

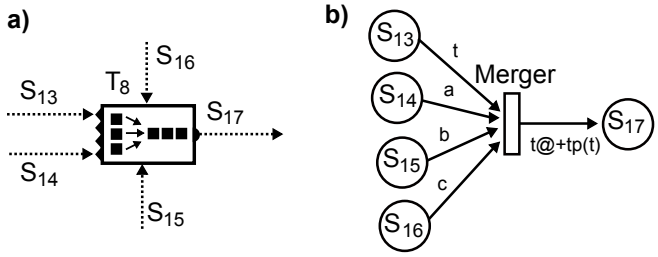

Figure 18: **a:** Merger task in Guaraná technology and **b:** merger task in timed and coloured Petri nets

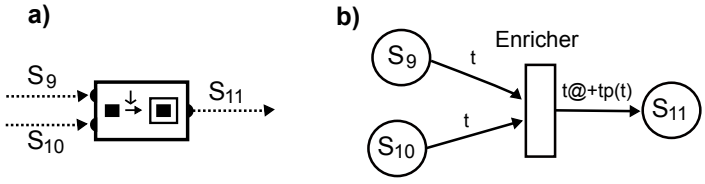

Figure 19: **a:** Context content enricher task in Guaraná technology and **b:** context content enricher task in timed and coloured Petri nets

Figure 20: **a:** Assembler task in Guaraná technology and **b:** assembler task in timed and coloured Petri nets

5. Experiments

In order to understand the behaviour of the integration solution, we observed some variables during the experiment in different operating scenarios. We analysed the time and the maximum number of messages accumulated in the slots. We assumed that the task processing time was fixed and the slots were not FIFO (first in, first out).

For this purpose, three experiments were defined, with 1,000, 5,000 and 10,000 messages, which are equivalent to tokens in the simulation model using Petri nets. For each experiment, three scenarios were created, as shown in Table 1. For each scenario configuration we randomly distributed the number of tokens according to the three priority types (1, 2 and 3). The priority type is represented in the table as the variable time. Each scenario was simulated 25 times, to exclude any discrepancies in the data (Grinstead and Snell, 2012). All experiments used priorities 1, 2 and 3, which are equivalent to colours in Petri nets.

Configuration 1			Configuration 2			Configuration 3		
Total	Token	Time	Total	Token	Time	Total	Token	Time
1,000	300	1	1,000	500	1	1,000	200	1
	300	2		250	2		600	2
	400	3		250	3		200	3
5,000	1,000	1	5,000	3,000	1	5,000	1,500	1
	2,000	2		1,000	2		2,000	2
	2,000	3		1,000	3		1,500	3
10,000	3,000	1	10,000	3,000	1	10,000	2,000	1
	3,000	2		2,500	2		6,000	2
	4,000	3		2,500	3		2,000	3

Table 1: Configurations of simulation scenarios

The simulation model was designed using CPN Tools for Petri nets. This simulation tool presents the average time of permanence of the messages in each place and the maximum size of messages accumulated in the places. The places represent the slots in the conceptual model of the integration solution.

For this model, a set time was created for high, medium and low colours (priorities). These colours indicate the priority of each message, regardless of its size. With the creation of the set "time", the variable t was inserted, which is an element of this set. Associating a time for a colour means adding a time value to the colour.

Each colour is processed with a different time. A high-priority colour is processed in a one-by-one time unit, the medium priority is processed two by two, and the low priority is processed by every three time units. For this purpose, the functions $tp_2(x)$, tp_x, $f_2(r)$ and $f_1(r)$ were created.

The tokens were randomly shot into the model using a firing control system. This means that, every one, two or three units of time, the control inserts a token into the entry port of the model randomly based on the value of its priority. The time increment represents the interval of token entries in the system.

For the development of the filter task we created a set F, where $F = \{int\ with\ 1..20\}$, consisting of integers between 1 and 20. After that, the Boolean filter function *fun filter (x) = (x <= 5%)* was created. This means that 95% of the messages will follow the flow, and 5% will be withdrawn.

Figure 16 shows the correlator created in Petri nets. The correlator transition is composed of places C_1 and C_2 and transition C_2. This task waits for the arrival of all correlated tokens, to continue the flow.

In order to generate the inputs of the video monitor, hardware monitor and printer applications, a specific control for the input port has also been developed. We created a set *increment* and the variables a, b and c, with these tokens, which were randomly inserted between high, medium, and low priority colours.

5.1. Discussion

After the configuration of the simulation model in CPN Tools, the experiments were performed. Initially, we performed simulations with 1,000 messages, presenting the results of the variable "average time of permanence of the messages in the slots", considering the three scenarios. Different processing times were inserted in the messages, according to their priorities, as shown in Figure 21.

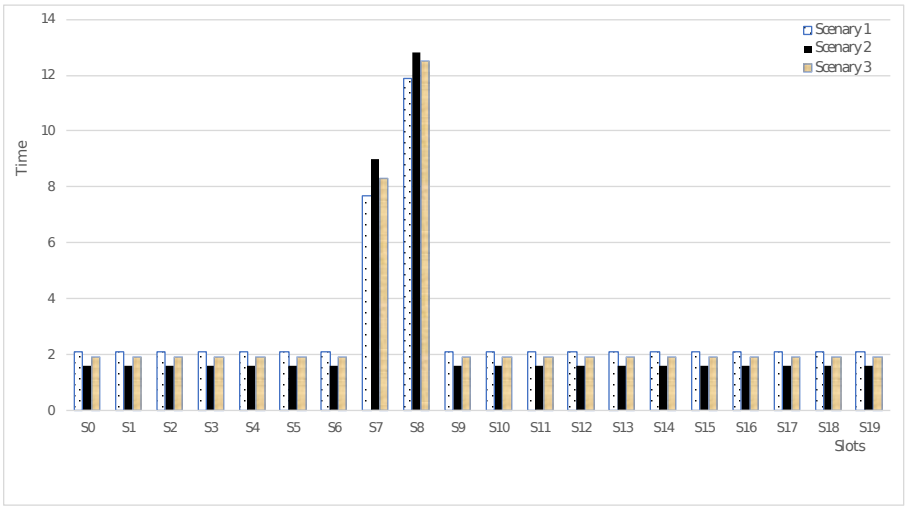

Figure 21: Average time of stay of messages in slots (1,000 tokens)

In both scenarios, we realized that the message takes longer to process in the S_7 and S_8 slots, because the request port P_2 is asynchronous and it needs to wait to be correlated. Thus, the correlator task causes a delay whenever it needs to wait for messages that have the same identity to correlate.

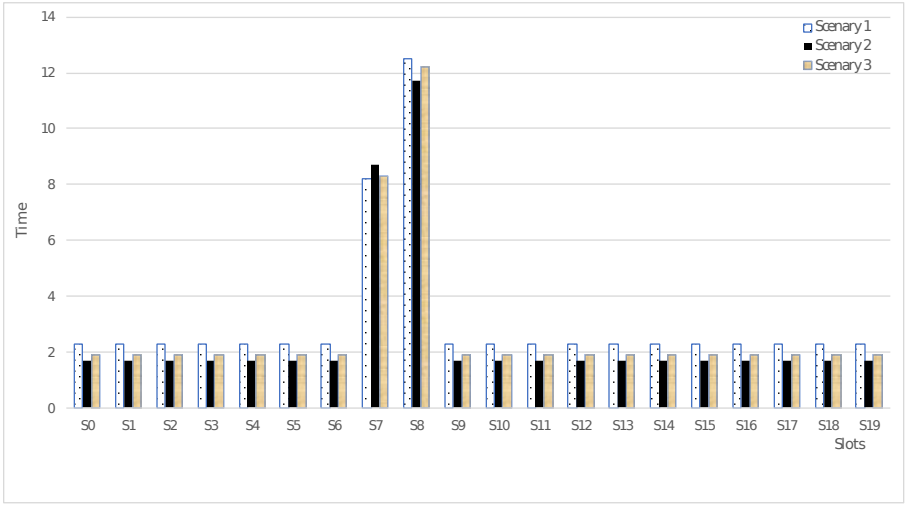

Figure 22: Average time of stay of messages in slots (5,000 tokens)

We realized that this situation is repeated in the experiments with 5,000 and 10,000 messages, as shown in Figures 22 and 23 for the three scenarios.

It is important to note that slots S_{14}, S_{15} and S_{16} are not included in the analysis, since the messages from these slots do not require any processing; they are released with only the status information of each of the applications to which they are connected and they are included in a single slot (S_{17}) with the S_{13} slot message.

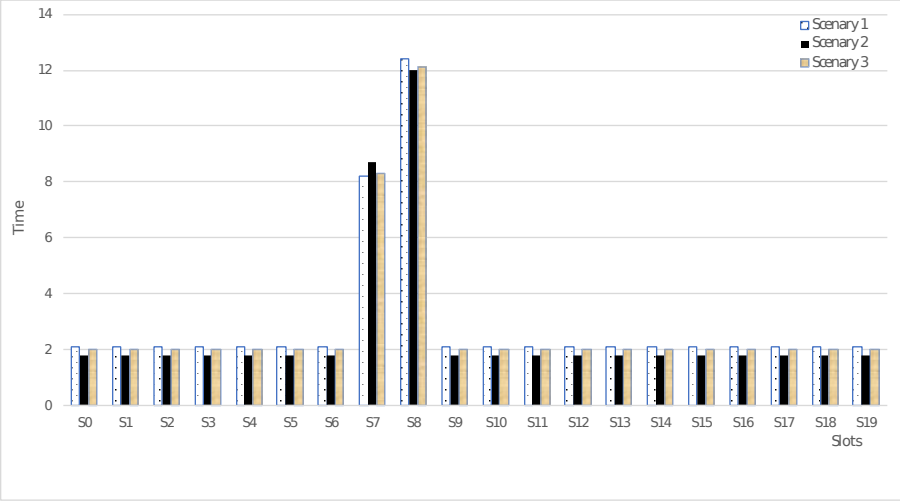

Figure 23: Average time of stay of messages in slots (10,000 tokens)

The accumulation of messages was also concentrated in slots S_7 and S_8, as shown in Figures 24, 25 and 26. The experimental results of the simulations show that the correlator strongly influences the behaviour of the system. We realized that the accumulation increases considerably in both of the slots (S_7 and S_8) that are connected to the correlator, while in the other slots analysed, the number of accumulated messages was small.

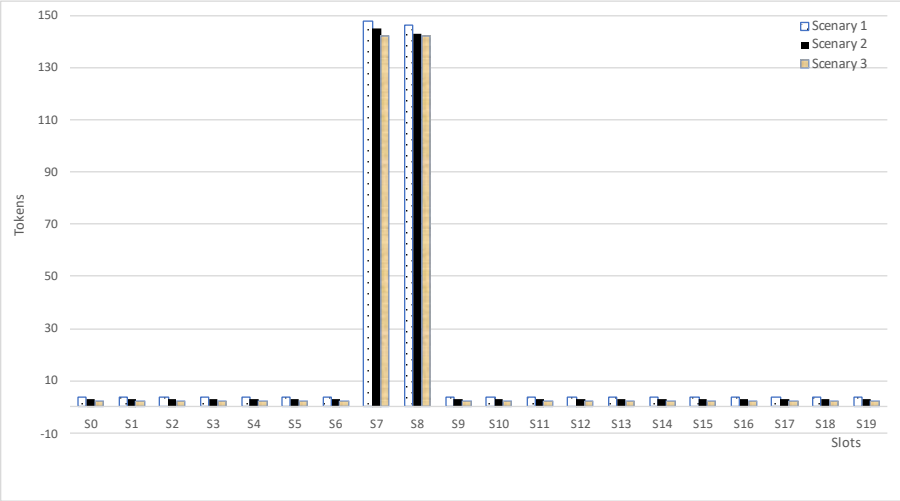

Figure 24: Maximum message accumulation in slots (1,000 tokens)

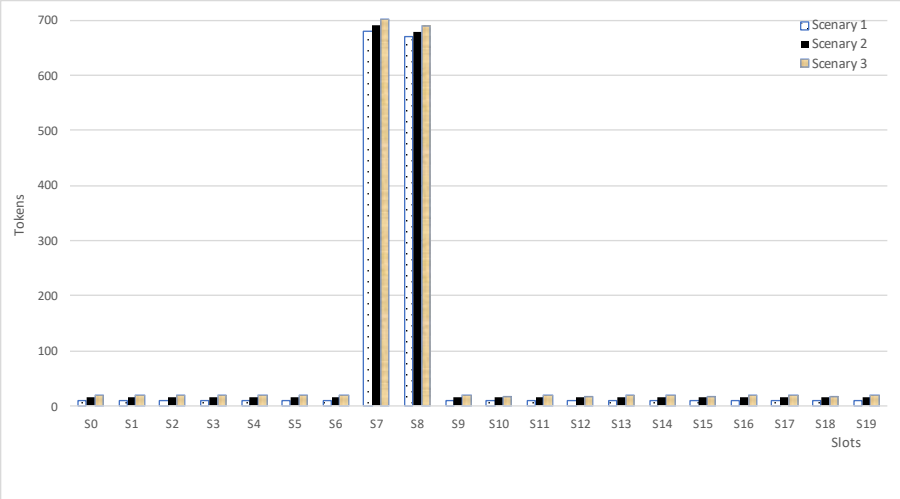

Figure 25: Maximum message accumulation in slots (5,000 tokens)

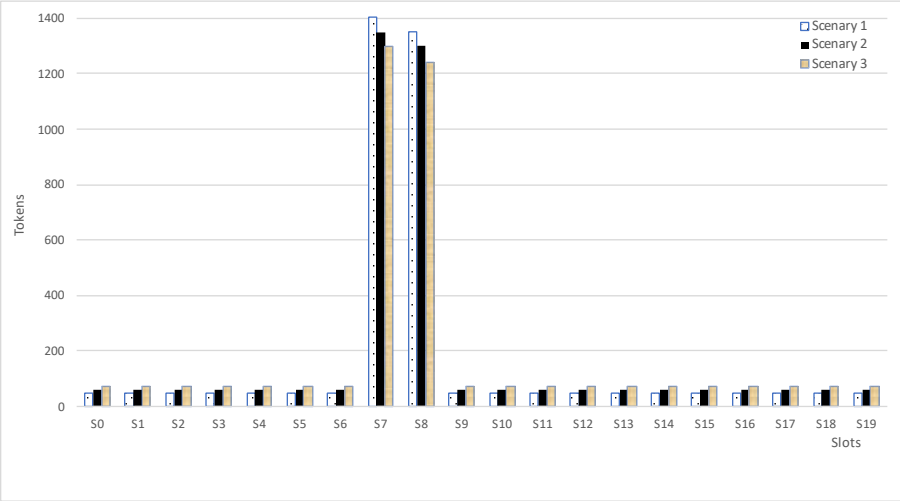

Figure 26: Maximum message accumulation in slots (10,000 tokens)

5.2. Formal Verification

The formal verification of the model was performed using the technique described by Kleijnen (1999), in which an expert in the area determines the behaviour of the system. An integration solution is a messaging system composed of temporary storage units called slots, in which message accumulation is expected to exist.

The correlator task (T_5) is connected by input slots S_7 and S_8. Slot S_8 receives the message from the replicator task (T_3), and S_7 depends on the processing of the P_2 port. In this way, it is expected that there will be a greater accumulation of messages in S_7 and S_8, because these slots store messages to be correlated by the T_5 task. The aim of task T_5 is to correlate the messages; thus, it causes S_7 and S_8 to present performance bottlenecks. The experiment performed with the simulation model showed that, at the end of the simulation, a token accumulation was generated. The places S_7 and S_8 accumulated a greater number of messages than the others.

Following the proposal of Kleijnen (1999), the comparison of the expected characteristics with those found in the experiment demonstrates that the expected message accumulation in the slots was repeated in the places. This demonstrates that the simulation model operates as a message-oriented system.

The accumulation of messages in the queue was expected in the slots that are connected to the input of the correlator task (T_5): in this case, the S_7 and

S_8 slots. In fact, the accumulation in places S_7 and S_8 was higher than in the others. The comparison between the behaviour expected by the experts in the area and the experiment, shows that there is similarity between the behaviour of the integration solution and the simulation model.

The proposal of Sargent (2005) for verifying the model is applied to the simulation model itself. Experiments were performed with 1,000, 5,000 and 10,000 tokens. Each experiment was executed once, and the intention was to describe the message flow, in order to discover whether the model was implemented correctly in the tool.

The tokens were only eliminated after the filter task. Therefore, it is necessary to count them and multiply the result by the inverse of 0.95, then add the results to the values that have accumulated in the slots during the process. The number of tokens processed and accumulated must be equal to the number of tokens triggered by the input port of the simulation model. The values used to perform the inverse multiplication of 0.95 were the numbers of tokens that arrived at the end of the simulation model process of each experiment. The simulation model was validated by means of the sum of three variables. The first variable is the number of messages that left the integration process, the second is the number of messages accumulated in the slots and the third is the number of messages accumulated in the filter. The sum of all these variables must be equal to the number of messages that entered the system, i.e., 1,000, 5,000 or 10,000.

6. Conclusion

This work developed a simulation model based on coloured and timed Petri nets from a conceptual model of an integration solution. We used the simulation to analyse the behaviour of an integration solution before its implementation, in order to assist software engineers to identify the possible errors and performance bottlenecks, in the design phase. The modelling steps used in this work are generic and could be used in any case study. This procedure saves financial and time costs by improving the quality of the proposed integration solution. The main contribution of this work is the identification of possible performance bottlenecks in integration solutions, before their implementation, using the mathematical formalism of Petri nets.

Timed and coloured Petri nets can be used as mathematical formalism options to find performance bottlenecks in an integration solution for a discrete-event system.

References

Alla, H., Ghomri, L., 2012. Modeling and simulation by hybrid petri nets, in: Proceedings of the Winter Simulation Conference, IEEE. pp. 3407–3414.

Allani, I.C., Zouari, B., Ghedira, C., 2018. Colored petri net model for secure document management in business process systems, in: 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT), pp. 829–834.

An, Y., Zhu, C., Chen, P., Li, Y., 2017. Modeling and analysis of transit signal priority control systems based on colored petri nets, in: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2701–2706. doi:10.1109/SMC.2017.8123034.

Arciniegas, D., Herrera, M.A., Táutiva, K., Bermudez, L.M., Castellanos, J.S., Angulo, J., 2017. Automation process modeling of a electric cars production line through petri nets and grafcet, in: 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC), pp. 1–6. doi:10.1109/CCAC.2017.8276475.

Du, S., Wu, P., Wu, G., Yao, C., Zhang, L., 2018. The collaborative system workflow management of industrial design based on hierarchical colored petri-net. IEEE Access PP, 1–1. doi:10.1109/ACCESS.2018.2809439.

Entezari-Maleki, R., Etesami, S.E., Ghorbani, N., Niaki, A.A., Sousa, L., Movaghar, A., 2017. Modeling and evaluation of service composition in commercial multiclouds using timed colored petri nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems PP, 1–15. doi:10.1109/TSMC.2017.2768586.

Grinstead, C.M., Snell, J.L., 2012. Introduction to probability. American Mathematical Soc.

Hohpe, G., Woolf, B., 2004. Enterprise integration patterns: Designing, building, and deploying messaging solutions. Addison-Wesley Professional.

Hu, Q., Zhao, Z., Wang, D., Du, J., 2017. A similarity computing method based levenshtein distance and logic petri net for renting cloud service processes, in: 2017 3rd IEEE International Conference on Computer and Communications (ICCC), pp. 2445–2449. doi:10.1109/CompComm.2017.8322974.

Kaur, K., Rana, R., Kumar, N., Singh, M., Mishra, S., 2017. A colored petri net based frequency support scheme using fleet of electric vehicles in smart grid environment, in: 2017 IEEE Power Energy Society General Meeting, pp. 1–1. doi:10.1109/PESGM.2017.8274627.

Kleijnen, J.P.C., 1999. Validation of models: Statistical techniques and data availability, in: Proceedings of the 31st Conference on Winter Simulation: Simulation—a Bridge to the Future - Volume 1, ACM, New York, NY, USA. pp. 647–654. URL: http://doi.acm.org/10.1145/324138.324450, doi:10.1145/324138.324450.

Li, J., Yu, X., Zhou, M., 2018. Analysis of unbounded petri net with lean reachability trees. IEEE Transactions on Systems, Man, and Cybernetics: Systems PP, 1–10. doi:10.1109/TSMC.2018.2791527.

Liu, Y., Li, X., Lin, Y., Kang, R., Xiao, L., 2017. A colored generalized stochastic petri net simulation model for service reliability evaluation of active-active cloud data center based on it infrastructure, in: 2017 2nd International Conference on System Reliability and Safety (ICSRS), pp. 51–56. doi:10.1109/ICSRS.2017.8272796.

Messerschmitt, D.G., Szyperski, C., 2003. Software Ecosystem: Understanding an Indispensable Technology and Industry. MIT Press, Cambridge, MA, USA.

Rezai, M., Ito, M., Lawrence, P., 1995. Modeling and simulation of hybrid control systems by global petri nets, in: Proceedings of the IEEE International Symposium on Circuits and Systems - Volume 2: ISCAS,, IEEE. IEEE. pp. 908–911.

Sargent, R.G., 2005. Verification and validation of simulation models, in: Proceedings of the 37th Conference on Winter Simulation, Winter Simulation Conference. pp. 130–143. URL: http://dl.acm.org/citation.cfm?id=1162708.1162736.

Shmeleva, T.R., 2017. Measuring subnets of colored petri net models: Online technique for networks perfomance evaluation, in: 2017 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), pp. 1–5. doi:10.1109/UkrMiCo.2017.8095417.

Strzęciwilk, D., Pękala, R., Kwater, T., 2018. Performance analysis of priority queueing systems using timed petri nets, in: 19th International Conference Computational Problems of Electrical Engineering, pp. 1–4.

Sun, F., Zhang, W., Chen, J., Wu, H., Tan, C., Su, W., 2018. Fused fuzzy petri nets: a shared control method for brain computer interface systems. IEEE Transactions on Cognitive and Developmental Systems PP, 1–1. doi:10.1109/TCDS.2018.2818173.

Taghinezhad-Niar, A., Javadzadeh, T., Farzinvash, L., 2017. Modeling of resource monitoring in federated cloud using colored petri net, in: 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), pp. 0577–0582. doi:10.1109/KBEI.2017.8324866.

Tigane, S., Kahloul, L., Bourekkache, S., 2017. Reconfigurable stochastic petri nets: A new formalism for reconfigurable discrete event systems, in: 2017 International Conference on Mathematics and Information Technology (ICMIT), pp. 301–308. doi:10.1109/MATHIT.2017.8259733.

Wang, J., Tian, J., Sun, R., 2018. Emergency healthcare resource requirement analysis: A stochastic timed petri net approach, in: 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), pp. 1–6.

Wang, R., Agarwal, S., Dagli, C.H., 2015. Opm amp; color petri nets based executable system of systems architecting: A building block in fila-sos, in: 2015 Annual IEEE Systems Conference (SysCon) Proceedings, pp. 554–561. doi:10.1109/SYSCON.2015.7116810.

Zeineb, M., Sajeh, Z., Belhassen, Z., 2016. Generic colored petri nets modeling approach for performance analysis of smart grid system, in: 2016 7th International Renewable Energy Congress (IREC), pp. 1–6. doi:10.1109/IREC.2016.7478905.

Zhang, X., Yue, S., Zha, X., 2018. Method of power grid fault diagnosis using intuitionistic fuzzy petri nets. IET Generation, Transmission Distribution 12, 295–302. doi:10.1049/iet-gtd.2017.0471.