

*Chapter 1***MODELING, ANALYSIS AND SIMULATION OF  
ENTERPRISE INTEGRATION SOLUTIONS USING  
MARKOV CHAINS****Márcia Horn<sup>1</sup>, Sandro Sawicki<sup>\*1</sup>, Fabricia Roos-Frantz<sup>1</sup>, Rafael Z. Frantz<sup>1</sup>, and  
Igor G. Haugg<sup>1</sup>**<sup>1</sup>Unijuí University, Department of Exact Sciences and Engineering, Ijuí, RS, Brazil**PACS** 05.45-a, 52.35.Mw, 96.50.Fm. **Keywords:** Enterprise Application Integration, Simulation, Markov Chains.**Abstract**

It is common sense that in order to assist their business processes, companies use heterogeneous software applications, mostly composed of legacy systems, third-party purchased software packages or systems developed by its own programmers teams as a solution for a specific problem. In this scenario, the main challenge facing companies is that most of their applications are not designed considering integration with other applications. The area of Enterprise Application Integration (EAI) has become essential to the information management, because it provides methodologies and tools to design and implement integration solutions without affecting the data structure and

---

<sup>\*</sup>Corresponding Author Email: sawicki@unijui.edu.br

current applications. The Guaran technology is a tool that provides the development of integration solutions allowing designs at a high level of abstraction, using a very intuitive and concrete graphical syntax. Integration solutions are softwares whose main function is to synchronize data among different applications or reuse functionalities. The creation of an integration solution follows the software development process; typically it includes the specification phase, the design, the implementation, testing and evolution. In this chapter we aim at analysing the behaviour and identifying bottlenecks performance in application integration solutions through the development of formal models of simulation based on Markov chains. The identification of bottlenecks in the early stages of development can contribute to improve the quality of integration solutions. As a case study, one personal phone management system implemented at Unijui University and developed by Guaraná technology is used. The experimental results show that it is possible to evaluate the quality of an integration solution still in the design stage without implement it. The proposed simulation model was validated using formal verification techniques.

## 1. Introduction

To follow a highly competitive business market and succeed in its initiatives, it is necessary that companies be able to use the huge amount of information generated at all times and manage them intelligently, producing valuable knowledge for decision making. Daily, companies face the challenge to receive large volumes of data from different systems.

In this context, information technology supports business organizations with the objective of meeting its management needs, offering agility and quality in its business processes. This support is provided by applications that compose the software ecosystem [26]. The software ecosystem is composed of a set of applications, usually heterogeneous, that is, they may have been developed using different technologies, programming languages, data models and operating systems.

Usually applications are not designed to run their functions in an integrated way. The application integration is necessary when an isolated application is no longer able to manage or support certain business processes. According to Hohpe and Woolf [17], application integration provides methodologies and tools to develop and implement integration solutions. Integration makes it possible to reuse applications already existing, in order to orchestrate these applications to keep them synchronized or provide new features that can be built from existing ones [38].

The Enterprise Application Integration (EAI) area uses computational techniques and tools so that companies can integrate data and functionalities offered by different applications. The purpose of an integration solution is to keep a series of data and application functionalities in sync or to develop new functionalities in relation to the existing ones, in such a way that the applications are not modified in the integration solution [17].

In order to integrate applications, software engineers generally follow a style of integration. The different existing integration of styles takes into consideration some criteria, such as application coupling, simplicity of integration, technology used by applications and data format [17]. Some technologies use integration patterns, such as Camel [18], Spring Integration [12], Mule [11] and Guaraná [13]. These technologies contains support for projects and development of integration solutions, as well as a domain-specific language based on

messages.

With the evolution of business processes, there is currently a large amount of business organizations that need to perform integration between their applications. Therefore, an integration solution includes a set of applications, or to orchestrate them in order to keep them synchronized or to provide new features already offered by those applications. The success of the correct and efficient implementation of the integration solution becomes essential. Thus, to improve the quality of the solutions developed, an analysis of the behavior of the integration solution is necessary to find the possible performance bottlenecks, still in the design phase.

The simulation is a field of research that deals with the experimentation of models that allow to make predictions about the behavior and the performance of actual systems. It is based on the use of mathematical techniques in order to understand the behavior of a actual system through formal models.

An integration solution can be characterized as a stochastic, dynamic and discrete model [36]. Stochastic model systems occur when one or more of their input variables is random. The integration solutions have identical characteristics to the stochastic model. Dynamic models are characterized by representing systems that change their state over time. In this case, one has the number of messages to be processed in a certain execution time, and the operations performed in the messages. The discrete models change their state at specific points of time, from the occurrence of events. By consuming specific processing time, the integration solution resemble discrete models.

Characterizing a solution of conceptual integration as a model of discrete events, it allows predicting the future behavior of the system, through simulation techniques [36]. There are several tools already supported to support discrete event simulation, such as SimEvents [6], ProModel [33], PRISM Model Checker [24], Pipe [10] and Arena [7].

This chapter presents the modeling, analysis and simulation of the behavior of an integration solution. The main objective is to understand the behavior and identify performance bottlenecks in integration solutions of enterprise applications modeled on the Guaraná technology, before its implementation and execution, through the development of a mathematical model in Markov Chains. As a case study, a system of personal phone management implemented at Unijui University was used. Analyzes and simulations were performed using the tool called PRISM Model Checker [24]. This chapter is organized in the following way. Section 2 presents the related work about discrete event simulation and Markov Chains. Section 3 describes the theoretical framework on Guaraná technology, its Domain-Specific Language and construction blocks. This section also covers the basic concepts of Markov chains. Section 4 presents the formulation of the integration problem. Section 5 describes the Markov Chain simulation model developed, its equivalence to the conceptual model of the integration solution. Section 5 discusses the experimental results. Finally, section 6 presents the final considerations.

## 2. Background

In this section we present Guaraná technology, as well as its Domain-Specific Language (DSL), construction blocks and concrete syntax. Besides that, concepts of Markov Chains and Discrete Time Markov Chains (DTMC) are discussed .

## 2.1. Guaraná Technology

The Guaraná technology is the tool that supports the development of integration solutions. It allows to design solutions in a high level of abstraction, using a concrete graphical syntax that allows the software engineers to have the vision of the whole set of processes that compose an integration solution. The implemented models are platform independent, so software engineers do not rely on the knowledge and skills to design their solutions. With Guaraná technology, it is possible both to design integration solutions and to use it as a tool for implementing, executing and monitoring its solutions. In addition, Guaraná technology makes use of tasks based on design patterns.

### 2.1.1. Domain-Specific Language

The Domain-Specific Language (DSL) is a language used to solve specific problems in a particular domain, through a proper language applied software development. The use of a DSL is recommended when a specific problem has a simplified grouping of correlated abstractions and notations [27].

Domain-Specific Languages can be used by users with little knowledge in programming languages, since it contains a high level of abstraction and ease of use. With the simplicity of DSLs, effective communication is possible with customers, ease of maintenance, readable syntax and easy understanding, making the lowest development time.

Guaraná technology has a Domain-Specific Language with the objective of designing solutions for integration of business applications, with a high level of abstraction and reasonable cost, providing software engineers, the use of tools to create and implement solutions to integration. It allows the maintenance of the focus in the problem with graphical support and very intuitive modeling concepts. This language is called Guaraná DSL. Guaraná DSL is a modeling language based on the patterns of conceptual integration [13]. The transformation of Guaraná models into executable code can be achieved using model-driven engineering.

### 2.1.2. Construction Blocks

The implementation of a Domain-Specific Language allows software engineers to maintain their focus on the problem domain through graphical support and very intuitive modeling concepts. According to Frantz et al. [13] a solution presents a set of processes that integrate a set of applications. Guaraná DSL is a modeling language based on the conceptual integration patterns[13], which are supported by constructors such as:

- **Message:** An abstraction of information that is modified and shared through an integration solution. It consists of a header, message body and one or more attachments. The header includes custom properties and often the following predefined properties: message identifier, correlation identifier, sequence size, sequence number, address sender, expiration date, and message priority. The body contains the payload data, whose type is defined by the template parameter in the message class. Attachments allow messages to carry extra pieces of data associated with the payload, for example an image or an e-mail message.

- **Task:** An atomic operation that can be executed in a message, such as a division, aggregation, translation, filter, combination, replication, among others. A task may have one or more entries as well as one or more outputs where messages arrive, are processed, and sent.
- **Slot:** A channel connecting the input of a task with the output of another task where the messages are processed asynchronously by the tasks. A slot can follow policies to distribute messages between tasks, such as priority-based outputs, or a FIFO.
- **Port:** The ports can be considered an abstraction of the details necessary for the communication of components within a software ecosystem. In general, a port allows the writing, reading, requesting and responding of communication operations with the integration components.
- **Process:** The integration logic that performs the transformation, routing, modification, and operations related to message time. The process consists of ports that allow communication with the integration components, slots and set of tasks to specify the logic of integration.

Conceptually, an integration solution adds one or more integration processes through which message flows which are processed asynchronously. The integration stream is implemented as a pipeline and filter architecture, where the channels are implemented by slots and the filters are implemented by the tasks. Each task realizes an integration pattern [17] and its execution depends on the availability of messages in the slots connected to its inputs. Slots are key builders to trigger asynchronously an integration solution, so the messages are stored in the slots until they can be processed by the next task in the integration stream.

## 2.2. Concrete Syntax

Guaraná technology makes it possible to design integration solutions using graphical syntax. The functionality and structure of an integration solution are completely defined using the building block language. Through these symbols it is possible to express all proposed classes: *Application*, *Process*, *EntryPort*, *ExitPort*, *IntegrationLink*, *ApplicationLink*, *Slot* e *Task*. An integration solution modeled from this language is formed by a set of applications and tasks that cooperate to integrate the different Applications. Guaraná tasks are based on Enterprise Integration Patterns (EIP) by Hohpe and Woolf [17]. This language provides a set of tools, called *task*, that are used to model different business application integrations.

The symbol used to represent tasks has inputs and outputs that can be observed as saliencies on the side of the icon. These inputs and outputs are used to connect tasks to each other, through slots. In an integration solution are transmitted messages, which contain information that is sent by the applications. In Guaraná technology the tasks are classified according to their semantics in *Router*, *Modifier*, *Transformer*, *Stream Dealer*, *Mapper and Communicator*. These task groups are responsible for the communication between the processes.

Notation	Concept	Notation	Concept
	<i>Application</i>	$\rightarrow$	<i>IntegrationLink</i>
	<i>Process</i>	—	<i>ApplicationLink</i>
	<i>EntryPort</i>	$\cdots\rightarrow$	<i>Slot</i>
	<i>ExitPort</i>		<i>Task</i>

Table 1. Symbology of the concrete syntax (from Frantz et al. [13])

### 2.3. Markov Chains

Markov Chains is a mathematical formalism for the modeling of systems as a Stochastic process. The modeled system is characterized by its states and the way in which they alternate. Markov Chains are stochastic processes with discrete states. The parameter, which is usually the time, can be discrete or continuous. Markov Chains are characterized by their future state depending only on their current state, being that past states do not influence the future state. Markov Chains are stochastic processes with the so-called Markovian property.

A Markovian process is a Chain Markov when the random variables  $X_t$  have a particular dependence relationship with time, these variables are defined in a space of discrete states. Transitions between states are modeled by a defined continuous or discrete stochastic process. According to Pinheiro [31] a stochastic process is defined as a collection of random variables ( $X(t)$ ) indexed by a parameter  $t$  belonging to a set  $T$ . Often  $T$  is the set of nonnegative integers (but other sets are perfectly possible) and  $X(t)$  represents a measurable characteristic of interest at time  $t$ .

In the context of Markov Chains, it is possible to represent a system using Continuous Time Markov Chains (CTMC) or Discrete Time Markov Chains (DTMC). A discrete-time Markov chain is a collection of random variables from a Markov process that assume values within an enumerable state space (finitely or infinitely), where we have the probabilities of transitions occurring from one state to another. Continuous-time Markov chains provide a different paradigm for system modeling. In CTMCs transitions between states can occur at any moment, that is, at any instant of time, unlike discrete-time Markov chains, where transitions always occur at discrete and known time instants [4].

This chapter discusses the use of Discrete Time Markov Chains, since they represent a model of a stochastic process that describes activities that end in events, these events generate the transitions of states. A Chain Markov is represented by a state machine, that is, a sequence of random variables  $X_t$  that represents the state at a given time. The variable  $X_2$  represents the state of the system at time 2. The domain of  $X_t$  is the state space itself.

#### 2.3.1. Discrete Time Markov Chains Properties

An important property of Markov Chains is lack of memory, where previous states are irrelevant, it is also known as a Markovian process. In terms of probabilities, a Discrete Time Markov Chains with state space  $S$  is a stochastic process  $\{X_n\} n \in T$ , where  $T =$

$\{0, 1, 2, \dots\}$ , such that the following properties are true:

- For any  $i \in S$  we have:

$$P(X_0 = i) = P_i \quad (1)$$

- For any  $i, j \in S$ , and  $n \in T$ :

$$P(X_{n+1} = j | X_n = i) = P_{ij} \quad (2)$$

- For any  $n \in T$  and  $i_0, i_1, \dots, i_{n-1}, i, j \in S$ , is used:

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = P(X_{n+1} = j | X_n = i) \quad (3)$$

Case  $X_n = i$  then the process at the instant  $n$  is in the state  $i$ . In particular, the third property shows that the present ( $X_n$ ), the future ( $X_{n+1}$ ) and the past ( $X_0, X_1, \dots, X_{n-1}$ ) are independent.

Equation 3 represents the probability of states by establishing a set of discrete states and that the next state  $s$  depends on the current state and not on the past. That is, the previous states are irrelevant to the prediction of the following states as long as the current state is known.

Markov Chains involve a matrix, called a transition matrix, whose elements are the transition probabilities from one state to another. Transition from one state to another is called a step. At each step the probability of reaching a next state is independent of the probabilities of the previous states. The transition matrix can be represented by a matrix  $n \times n$  where  $n$  is the number of chain states and each position  $ij$  of the matrix represents the probability of moving from state  $i$  to state  $j$ .

For Markov Chains homogeneous,  $P(X_{n+1} = j | X_n = i) = P_{ij}$ . Let  $\{X_n\}_{n \in T}$  be a homogeneous Markov chain with discrete state space  $S = \{1, 2, 3, \dots\}$ . In this case we have  $P_{ij} = P(X_{n+1} = j | X_n = i), i, j \geq 0$  independent of  $n$ . An intuitive way of presenting and operationalizing the transition probabilities between states is given by the stochastic matrix. A stochastic matrix of the transition Markov chain can be represented according to the matrix  $P$ , where each entry in the matrix  $P$  must be positive and the entries of each row in the matrix must add 1.  $P$  can be written as:

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1S} \\ P_{21} & P_{22} & \dots & P_{2S} \\ \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \dots & \vdots \\ P_{S1} & P_{S2} & \dots & P_{SS} \end{bmatrix} \quad (4)$$

That is, the transition matrix of a Markov chain with  $k$  states, must be:

$$P_{1j} + P_{2j} + \dots + P_{sj} = 1, j = 1, 2, \dots, s \quad (5)$$

Graphically a Discrete Time Markov Chains is represented by a diagram of states and transitions, which is a finite directed graph, where states are represented by nodes and transitions between states are represented by valued arrows, as shown in Figure 1:

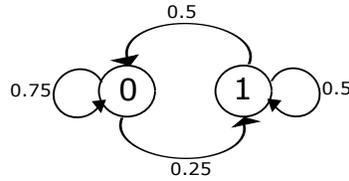


Figure 1. State diagram of two states

It is observed in Figure 1 that the sum of the transitivity probabilities of a state  $i$  for all other states of the Chain Markov is always equal to 1. Step 0 tends to follow step 1 with the probability of 0.25 and stay in step 0 with probability 0.75, step 1 tends to follow step 0 with the probability of 0.5 and with the probability 0.5 to remain in that step.

Generally it does not have to define how many states have a system of a Markov Chain, what can you do is specify the probabilities for each of the possible states. Thus, each transition matrix has a probability column vector representing the  $s$  states. A vector of state probabilities of a discrete time markov chain is called stationary if the transitions described by the matrix  $P$  do not modify these probabilities of state, that is,  $v_j = \sum v_i p_{ij}$  [4].

$$P = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_S \end{bmatrix} \quad (6)$$

The state  $v_1$  represents the probability of the system being in the first state,  $v_2$  represents the probability of being in the second state and  $v_s$  represents the probability of the system being in the  $s$ -th state. The choice of the use of Markov Chains of Discrete Time for this study is justified because it is a particular case of stochastic processes and, therefore, used for discrete event systems, which makes it possible to identify and analyze the performance bottlenecks also in integration solution.

### 3. Related Work

This section presents papers that use Discrete Time Markov Chains (DTMC) in discrete events simulation tools. It should be noted that there are different and numerous works in the literature that show the use of mathematical techniques along with the simulation process. However, regarding the authors, there are no works in the literature that relates the use of simulations and mathematical techniques to analyze the behavior of solutions for integrating business applications. The sections are subdivided into two parts: the first describes works that use the mathematical technique with Discrete Time Markov Chains with different simulation tools and the second presents works that use Discrete Time Markov Chains with the PRISM simulation tool.

## 4. Verification model using simulation tools

Ahluwalia and Singhal [2] proposes to model the performance of the communication architecture of a parallel machine. To ensure focus on the analyzes, the authors are restricted to a Single Instruction, Multiple Data (SIMD) machine. The authors developed a Discrete Time Markov Chains model in the network architecture to compute the delay time introduced by routing algorithms and network architecture.

In order to model the performance of the machine, they developed a Markov Chain model of a single node of the router, since modeling an entire system of routing would be much more complex. As the state of a router's buffer changes every cycle, the authors developed a Discrete Time Markov Chains model, and calculated the matrix of transitivity probabilities and the probabilities of stable Markov states Chain.

Gomes and Wanke [15] propose to apply the concept of Markov Chains in the management of spare parts. The authors pointed out that in a Chain Markov applied to inventory management, possible states denote the different stock positions that may occur over time. In this study, the authors modeled the consumption and stockpile policy of replacement parts through Markov Chains. Through the property of convergence of the Markov Chains it is possible to infer the distribution of probabilities of the stock position and, from these values, to determine indicators such as average stock, probability of lack and probability of stock supply. The authors found that the Markov Chains method is a more efficient computational alternative than the Excel spreadsheet simulation. They tested the maximum level of nine units in stock and a request point of six units, with average demand of two units per time period, obeying the Poisson distribution. The results suggest that both distributions are equivalent. However, gains in relation to less computational effort and execution time are substantial in Markov Chains modeling.

James et al. [19] presented two parameter estimation algorithms for fast samples of homogeneous Markov Chains. The algorithms proposed by the authors are obtained through the discretization of the stochastic differential equations involved in the estimation of hidden Markov models using the Expectation-Maximization (EM) algorithm. The first algorithm is based on the discretization of the continuous-time filters, discovered by Elliott, to estimate quantities used in the EM algorithm. The second is based on the discretization of continuous time facilitators, producing the reassessment equations of Baum-Welch.

Sandmann [34] investigates the importance of an ideal sampling to estimate probabilities of state in DTMC, both along infinity and steady state. The importance of sampling is a technique of reducing variance for efficient simulation through a change of measure. In particular, it can be applied to the simulation of rare events (events occurring with an extremely small probability) of Markon Chains. Such rare events are important in determining substantially the performance and reliability of the system. A common technique in applying the importance of sampling to the Markovian process is to change the arrival of service rates, which corresponds to the evolution of the Continuous Time Markov Chains transition rates, or the transition to the underlying probabilities of the Discrete Time Markov Chains.

Abrahám et al. [1] presented a model verification algorithm, using Discrete Time Markov Chains. The algorithm is based on the detection and capture of strongly connected components, besides, it offers counterexamples, which can be interactively refined by the user. During the verification of the algorithm, all information for the phases of recursion

are stored. This allows a user who needs to obtain a counterexample for a given property to have accessibility to formulate an abstract against example. In the end, it briefly produces counterexamples that are constructed according to the structure of Discrete Time Markov Chains.

Clément et al. [5] propose a study from the observation of uniformity of the population protocols models, which allows to apply an abstraction counter. The contribution is to highlight two recipes for verifying the correct stabilization property in models with finite population sizes. For a population of  $k$  agents,  $\pi = \{\pi_1, \dots, \pi_k\}$  denotes the set of agents. Each agent in the system is modeled as a finite state machine, which represents the agent program. These programs are uniform: each agent executes the same finite state machine, so that it does not depend of the number of agents in the system. The strongly anonymous model: agents cannot store a unique identifier. When two agents meet, they can interact and make a change in their states. The underlying communication network is a complete graph; each pair of agents can meet. The initial state of each agent is defined by the protocol. The goal of a population protocol is to calculate for each agent the same value as the protocol output. The value to be calculated is the value of a function of the initial state of the protocol.

Basagiannis et al. [3] introduced the verification of a probabilistic model as an approach of an assisted tool, viable to systematically quantify security threats. These threats are expressed as probabilistic accessibility properties, which are automatically verified through a DTMC, which represents the protocol participants and the invader model. The verification analysis of the probabilistic model is performed in the PRISM software. The probabilistic model verification approach is based on transitions labeled between model states, with information about the probability of occurring.

Kwon and Agha [25] described a new way of modeling network sensors with DTMC and expressed the aggregate properties of the network, both in their transient states and in their steady state. The authors modeled the transition dynamics between states as a DTMC and used the experimental results to illustrate and validate the method with the simulation in PRISM Model Checker.

Kumar and Vasudevan [21] introduced a systematic and rigorous methodology to verify the design correction in Register Transfer Level (RTL). The source code RTL uses statistical analysis techniques to calculate the probabilities. They modeled the RTL probabilistic modules in a DTMC, which are then formally verified for probabilistic invariants using the PRISM tool. Kumar and Vasudevan [22] performed the timing check in RTL in the context of process variables. They constructed macro models for the variable delay of RTL operators to represent them as Gaussian random variables. Gauss delay variables are used as the DTMC state variables. Finally, the PRISM tool is used to simulate the performance.

Norman et al. [28] presented an implementation of a control for the probability of  $\pi$ -calculation, being an algebraic process that supports the modeling of the simultaneity, mobility and probabilistic behavior of a discrete model. Formal verification techniques for this calculation have applications in several domains, including protocols ad-hoc mobile network and random security protocols. The authors show an automatic procedure for the construction of the Markov decision process that represents a process of probabilistic  $\pi$ -calculation. This can be verified using verifiers of existing probabilistic models, such as PRISM Model Checker.

Paulevé et al. [30] propose a technique for adjusting temporal characteristics within stochastic calculations, presenting the construction of the stochasticity absorption factor in the classic stochastic calculation, with exponential rates. The probabilistic model of stochastic calculations with the Erlang distributions was obtained using the PRISM tool.

## 5. Integration Problem

The integration solution proposed by Frantz [14] is illustrated in Figure 5. It is noticed that the workflow starts at the input of the port  $P_0$  that, periodically, searches the Central with the intention of finding new telephone calls. Each call results in a message that added the  $P_0$  on the  $S_0$  queue, enabling the  $T_0$  task can be performed. Whenever a job is running, incoming messages are waiting in line, and will be selected for processing by FIFO discipline.

Task  $T_0$  filters and discards all calls that result in toll-free messages and routes only those that cost to the queue  $S_1$  to be executed by task  $T_1$ . Task  $T_1$  makes a copy of the message, which is forwarded to queue  $S_2$  to be processed by task  $T_2$ , which translates the contents of the message into the format of the Human Resources application, forwarding to queue  $S_3$  to query the application, via the gateway request  $P_{1(A)}$ . Messages return through the request port  $P_{1(B)}$  and sends the  $S_5$  queue. Another copy is forwarded to queue  $S_4$ , which waits for a message correlated by queue  $S_5$  to be processed by task  $T_3$ .

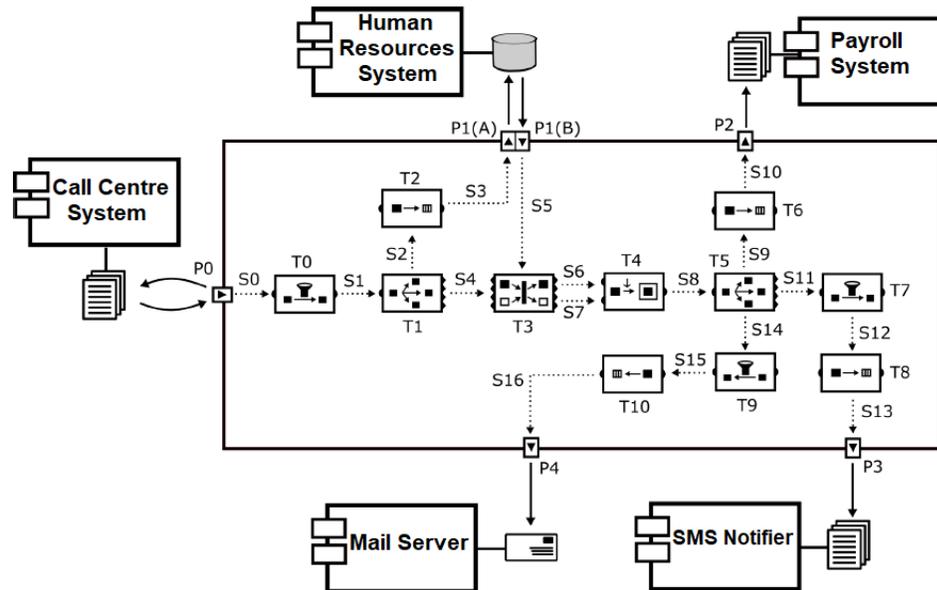


Figure 2. Conceptual Model Developed using Guaráná technology to represent an integration solution for the Telephone Call Management Problem. Source: [14].

The task  $T_3$  analyzes the incoming messages and produces as output a set of correlated data, containing in the queue  $S_6$  information found in the Human Resources and in the  $S_7$

queue the Central Telephone information. When there are correlated messages, they are conducted for the  $T_4$  task that adds to the body of the message the information that returned from the Human Resources application with the information coming from the Telephone Company.

After the message transits in task  $T_4$ , it is forwarded to queue  $S_8$  so that task  $T_5$  can be performed. The task  $T_5$  makes a copy of the enriched message and has how to target the applications of port  $P_3$  referring to Payroll; Port  $P_4$  referring to the Messaging Service and port  $P_5$  referring to E-mail Server. A copy forwarded to the queue  $S_9$  that is destined to task  $T_6$ , which, when executed, translates the contents of the message to the format of the Payroll application.

After the message is processed by task  $T_6$ , the message is routed to queue  $S_{10}$  that destines port  $P_2$  informing the debit orders in the payroll. Another copy is forwarded to the  $S_{11}$  queue that is destined to task  $T_7$ , which, when processing the message, analyzes and discards the message that does not have the employee's telephone number information. After the message is filtered, it is forwarded to queue  $S_{12}$  that is destined to task  $T_8$ , which, when executing, translates the content of the message to the format of the application and sends the message to the  $S_{13}$  queue that is destined to port  $P_3$ , which sends a text message to the employee's cell phone informing the charge. Another copy is forwarded to queue  $S_{14}$  which is destined to task  $T_9$ , which, when processing the message, analyzes and discards the message that does not have the employee's email address information. After the message is filtered, forwarded to  $S_{15}$  queue which has as destination the task  $T_{10}$  that translates the content of the message into the format of the E-mail Server application and forwards the message to the queue  $S_{15}$  that is destined to port  $P_4$ , which sends an expense statement to the employee's email.

## 6. Formal Model Proposed

This section presents the transition diagram between matrix states of a Markov chain and the mathematical model. Therefore, it is necessary that the variables that will be observed to identify the performance bottlenecks of the integration solution are presented.

### 6.1. Observed Variables

Another common feature between the integration solution presented by Guaraná and a system of discrete events is the relationship between its elements and its operating structure. In the modeling of discrete event systems, as an integration solution, the identification and relation with the variables to the performance and the way in which they interact with each other and with the other elements, aids the understanding of the system.

In the Guaraná technology, the entities are messages, that is, objects that allow interaction with the system. A process is a sequence of activities. An activity occurs between two events, and during an event, the state of the system can change. The occurrence of an event in the Guaraná technology can be characterized as the arrival of a message to be processed by a task. Each time a task processes a message, the system changes its state. A task occurs between two events, so you can be characterized as an activity and, consequently, a process can be characterized as a sequence of tasks. In general, it is also possible to define the size

Table 2. Main variables used as measures of system performance [32]

System	TS = mean time that the message stays in the system NS = mean number of messages in the system
Arrivel process	$\lambda$ = mean arrival rate IC = mean interval between arrivals
Queue	TF = mean time spent by the message in the slot NF = mean of messages in the slot
Service process	TA = mean time of service $\mu$ = mean number of messages executed

of queues, however, in an integration solution designed with Guaraná technology, the queue size, characterized as a slot can contain an infinite number of messages.

The  $M/M/1$  queuing model presented by [20] is a model based on the queuing theory that is widely used for discrete and simulation events. This model also allows the construction and understanding of basic ideas and methods of queuing theory. According to [32] in the model  $M/M/1$  both arrivals and service are Marcovian (which is to say that they follow the Poisson distribution or the negative exponential) and that we have a single attendant, being the finite population.

Prado [32] states that the service process is also quantified by an important random variable. The Greek letter  $\mu$  means average attendance rate, and  $TA$  time or mean time of service. The representation of a queuing system requires knowledge about some random variables used, the main ones being the following:

- $\lambda$  = mean arrival rate
- IC = mean interval between arrivals =  $1/\lambda$
- $\mu$  = mean number of messages executed
- TA = mean time of service =  $1/\mu$

Table 2 describes the main variables used as measures of performance regarding the system, process of arrival, queue and service process. Table 4 describes the formulas that treat the main random variables of the  $M/M/1$  model. When we know the probability distribution that describes the number of customers in the system, we can calculate the queue parameters: average number of clients in the queue, average number of clients in the system, average time during which the client queues, average time during queue which client is in the system. We can also calculate the probability of number customers in the system, the probability distribution of the number of customers in the system.

## 6.2. Equivalence of the elements of Guaraná technology with Markov chains

This section discusses the equivalence of the elements of an integration solution projected in the Guaraná technology with the elements that make up the Markov Chains. The

Table 3. Random variables to evaluate the behaviour of M/M/1 model [32]

Name	Description	Formula
NF	mean number of messages in the queue	$NF = \frac{\lambda^2}{\mu(\mu-\lambda)}$
NS	mean number of messages in the system	$NS = \frac{\lambda}{\mu-\lambda}$
TF	mean time that the message stays in the queue	$TF = \frac{\lambda}{\lambda(\mu-\lambda)}$
TS	mean time that the message stays in the system	$TS = \frac{1}{\mu(\mu-\lambda)}$

characterization of an integration solution as a system presenting a stochastic behavior makes it possible to demonstrate the equivalence of the solution with Markov Chains. A stochastic process represents, in this context, the evolution of the system of random variables in time. In other words, a process with degree of uncertainty, which evolves through a set of possible indications, from one of the possible states.

Discrete Time Markov chains can be described as a state transition model, increased with probabilities. Formally, it is described as a finite set of states ( $S$ ), an initial state  $s_0$  belonging to  $S$ , a transition probability matrix ( $P$ ) of  $S \times S \rightarrow [0-1]$  where the sum of the transitions of a state must be 100% , a labeling function attributing to the states a set of atomic propositions ( $L : S \rightarrow 2^{AP}$ ). Time is commonly seen in Discrete Time Markov Chain models as a discrete, homogeneous time interval, and time-independent transpose probabilities. An execution of a Markov Chain model of Discrete Time is represented by a path, with equal lengths of the same transitions. The path probability calculations allow to analyze the behavior of the system and the reason in properties, such as probabilistic accessibility. Quantitative and qualitative properties, including repeated accessibility and persistence, may be of interest in the context of a system behavior analysis, and conducted with the verification of the probabilistic model of the Markov Chain of Discrete Time. Modeling of retributive structures can also be used to represent benefits or cost characteristics. Retributive (instantaneous) and retributive (cumulative) transitions are modeled by the retributive functions  $rs: S \rightarrow R \geq 0$  e  $rt: SS \rightarrow R \geq 0$  respectively. A retributive structure can be used to measure the number of time slots spent in a state or the chance that the system is in a specific state after a certain number of time slots [23] [29].

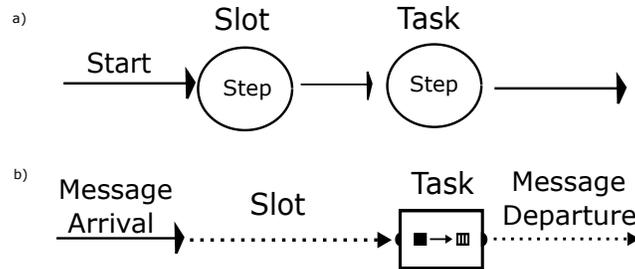


Figure 3. Generic model of a representation in Markov Chains (a) with the equivalence in Guaraná.

The Markov decision processes, such as the Markov Chain of Discrete Time , model a

system as a discrete set of states and transitions between states that occur at discrete time intervals. In addition, the Markov decision extends the models of the Discrete Time Markov Chain process by allowing for non-determinant choices. This type of modeling is especially suitable for competition, unknown environments and application of non-specific scenarios. Since Discrete Time Markov Chain models are totally probabilistic, they are unable to solve some aspects of the system, such as non-determinant choice [29].

Formally, a Markov decision process is a tuple  $(S, s_{init}, Steps, L)$  where  $S$  is a finite set of states,  $s_{init} \in S$ , and  $S$  is the initial state. The steps:  $S \rightarrow 2^{ActDist(S)}$  is the transition probability function, acting with a set of actions,  $Dist(S)$  othe set of discrete probability distributions on  $S$ , and  $(L : S \rightarrow 2^{AP})$  a labeling with atomic proportions. A path in a Markov decision process is a sequence of states and pairs of actions/distributions, that is,  $s_0(a_0, \mu_0)s_1(a_1, \mu_1)s_2$ , representing the execution of a system. Solving non-determinant paths (transformation of the model into a DTMC) and probabilistic choices, and then calculating a measure of probability on paths [23] [29].

Figure 3 (a) illustrates a generic model of Markov Chain elements. This process is based on Markov models, which have a similar structure of the conceptual model of an integration solution designed with Guaraná technology. In Figure 3 (a), from the moment a message is in the queue, the process starts, messages arrive at the beginning of the process, and wait in the system. Similarly, Figure 3 (b) illustrates a piece of an integration solution developed with Guaraná technology, in which message arrival represents the beginning of the process. In Markov Chains, the steps represent the slots and tasks of the Guaraná technology.

### 6.3. Simulation Model

Figure 4 shows the working schema of the Markov decision process, and aims to help the understanding of the method used in the model that represents an integration solution. As soon as there is a message in the queue, the process starts with the  $P_0$  state. This state indicates that a message has been found and taken from the queue. Then, in the  $T_0$  state, only messages containing the cost of the telephone call are chosen. Port  $P_1$  sends a message to the Human Resources System in order to obtain information from the employee.

The message goes to the Payroll system, state  $P_3$ , which discounts the value of the call made by the employee. Upon completion of the discount, the employee is notified by states  $P_4$  e  $P_5$ , which respectively represent an SMS messaging system and an e-mail server. However, it must undergo a process of filtering to prevent these systems from receiving an empty message or incomplete information from the receiver.

The Markov decision process represented as a tuple  $M = (S, s_{init}, P, L)$  the exampre in Figure 4 can be described as:

$S$  finite set of states. Ports =  $P_0, P_1, P_2, P_3, P_4$ , Slot =  $S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16}$  and Tasks =  $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}$  which represent a finite set of states, or state space.

$P = S \times S \rightarrow [0,1]$  is the transitivity probability matrix.

$L =$  The labeling with atomic propositions  $(L : S \rightarrow 2^{AP})$ , which are simply associated with the states in the current example and  $AP = [\text{start, filter, copy, translate, consult, compare, mix, debit and notify, forward, delete}]$ , the set of atomic propositions.

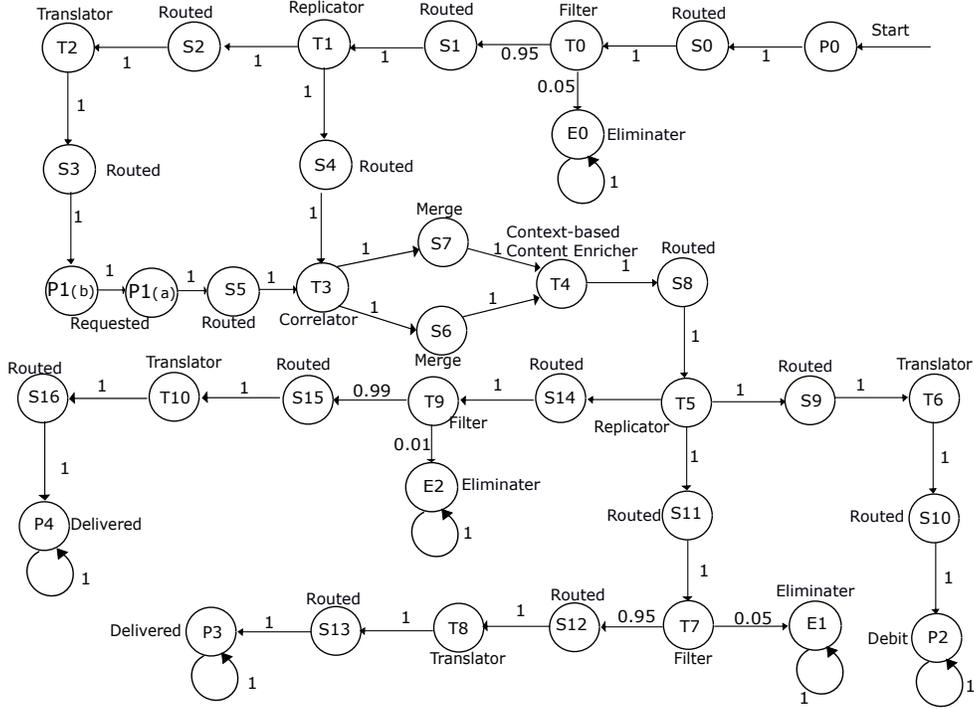


Figure 4. Diagram of Transformation between Matrix States of a Markov Chain corresponding to the Conceptual Model

### 6.3.1. Mathematical Model

Markov chains of discrete time, the random variables  $X_0, X_1, \dots, X_n, \dots$  are discrete, the set  $T$  is discrete ( $T = 0, 1, \dots, n, \dots$ ), then:

$$P(X_t = x_j | X_{t_1} = x_{i_1}, X_{t_2} = x_{i_2}, \dots, X_{t_n} = x_{i_n}) \quad (7)$$

The stochastic process  $X_t$  is a Markov Chain if the distribution of  $X_t$  is independent of all previous states in which the chain was found, with the exception of state immediately before, that is,

$$P(X_t = x_j | X_{t-1} = x_{i_{t-1}}, \dots, X_2 = x_{i_2}, X_1 = x_{i_1}) = P(X_t = x_j | X_{t-1} = x_{i_{t-1}}) \quad (8)$$

A Markov model where the state space  $I$  is discrete and is described by its state transition matrix. This matrix is dynamic because it allows the transition probabilities to change as a function of time  $t$ , where  $t$  is discrete.

Consider a Markov chain with  $N$  states  $x_n \in I$  and let  $x_i, x_j \in I$ . We denote  $x_i(t)$  to mean that the process is in state  $x_i$  at time  $t$ .

**Definition 1:** If  $p_{ij}$  is the probability of transitioning from state  $x_i(t)$  to state  $x_j(t+1)$ , then the matrix  $N \times N$ , given by

$$P_{ij} = Pr(X_{n+1} = j | X_n = i) \quad (9)$$

that is,

$$P = [p_{ij}] \quad (10)$$

It is called the state transition matrix of the Markov chain. Note that, in Definition 1, the sum of the lines of the matrix  $P$  must always be equal to 1. The transition matrix can also be given by a state transitions diagram. Figure 5 shows the state transitions diagram for a Markov chain with only 2 states.

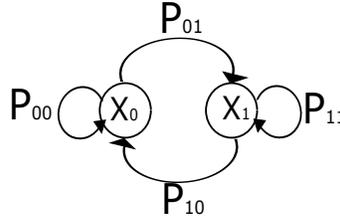


Figure 5. Transition diagram between two states of the matrix of a Markov chain. Source [9]

**Proposition 1:** For arbitrary  $t$ , we have that:

1. The probability of transitioning from state  $x_i(t)$  to state  $x_j(t+n)$  (in  $n$  steps) is given by  $P_{i,j}^n$ ;
2. The  $n$  step transition matrix, denoted by  $P_n$ , is calculated as the power  $n$  of the transitional matrix  $P$ , that is

$$P^n = P^n. \quad (11)$$

According to Figure 5, the transitions diagram represents the transition between two states, so the transition matrix is a square matrix, and is represented by the matrix  $P$ :

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{2 \times 2} \quad (12)$$

To simulate a Markov process, considering an initial state  $x_0$ , one can choose a successor state according to the probabilities  $p_{0j}$ , para  $j = 1, \dots, N$ , determining a new state  $x_1$ . The process is repeated to generate the next state, and so successively. Due to the probabilistic nature of the model, each time this simulation is repeated, it is likely that a different sequence of states will be obtained as a result. So the only way to analyze the process is to keep track of the probabilities of being in a state.

**Definition 2:** Let  $S_i(t)$  be the probability that a Markov process is in a state  $x_i$  at time  $t$ . Then the vector

$$s(t) = \begin{pmatrix} S_1(t) \\ S_2(t) \\ \vdots \\ S_n(t) \end{pmatrix} \quad (13)$$

is called the vector of distribution of state probabilities of the Markov chain at time  $t$ .

Let  $s^T(0)$  be the initial distribution of the process ( $s^T$  is the transposed vector of  $s$ ). The evolution of the distribution vector is governed by the transition matrix in steps  $t$ .

**Proposition 2:** For any time  $t$ , we have that

$$s^T(t) = s^T(0)P^t, \quad (14)$$

where  $P^t$  is calculate and  $s^T$  is the transposed vector of  $s$ .

## 7. Formal Verification

The verification of the proposed model follows the simulation techniques suggested by Sargent [35] and de Freitas Filho [8]. In each scenario, 25 repetitions were performed. According to de Freitas Filho [8] the differences between the different rounds of repetitions are expected. Such differences, considering the averages of these repetitions should be small. Sargent [35] points out that a large amount of variability between repeats makes the results of the model questionable. The repetitions are organized into tables and analyzed, and later the average of the messages accumulated in the queues in each slot is performed. The analysis found that there are no discrepancies in results.

By the definition presented by Kwiatkowska et al. [23], a Markov Chain of Discrete Time  $D$  is a tuple  $(S; \bar{s}; P; L)$ , where:

- $S$  is a finite set of states, represented by ports, slots, and tasks;
- $\bar{s} \in S$ , is the initial state;
- $P : S \times S \rightarrow [0, 1]$  is the transition probability  $P$  matrix where  $\sum_{s'} P(s, s') = 1$  for all  $s \in S$ ;
- $L : S \rightarrow 2^{AP}$  Is a marking function that assigns to each state  $s \in S$  the set  $L(s)$  of atomic propositions that are valid in the state.

Each  $P(S; s_0)$  element of the transition matrix gives the probability of doing a transition from state  $s$  to state  $s_0$ . Note that the probabilities of transitions that come from a single state must add one. Finalizing states, that is, those from which the system can not move to another state, can be modeled by adding a single transition back to the same state, with probability 1.

According to the state diagram between states of the matrix of a Chain Markov corresponding to the conceptual model, we obtain the partial transition diagram described in

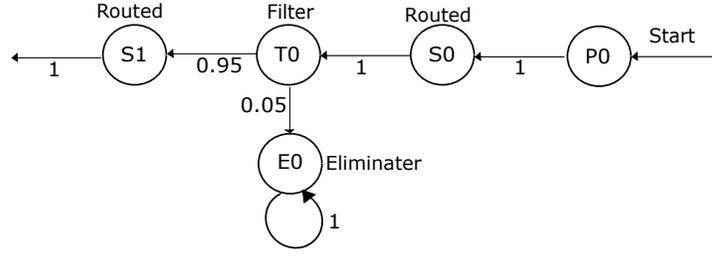


Figure 6. Partial state diagram of conceptual model

Figure 7., which describes the behavior of the complete transition matrix and presents a simple example of a Discrete Time Markov Chain  $D_1 = (S_1; \bar{s}_1; P_1; L_1)$ . In the graphical notation, states are drawn as circles and transitions as arrows, labeled with their associated probabilities. The initial state is indicated by an input additional arrow. The Discrete Time Markov Chain  $D_1$  has four states:  $\{P_0, S_0, T_0, S_1\}$ , with the initial state  $\bar{s} = S_0$ . The partial probability transition matrix representing Figure 7.  $P_1$ , is given by:

$$s(t) = \begin{bmatrix} & P_0 & S_0 & T_0 & E_0 & S_1 \\ P_0 & 1 & 0 & 0 & 0 & 0 \\ S_0 & 0 & 1 & 0 & 0 & 0 \\ T_0 & 0 & 0 & 0,95 & 0,05 & 0 \\ E_0 & 0 & 0 & 0 & 1 & 0 \\ S_1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

The atomic propositions used to label the states are taken from the set  $AP = \text{port } P_0$ , slot  $S_0$ , filter  $T_0$ , eliminated  $E_0$  and slot  $S_1$ . In this model the Discrete Time Markov Chain is a simple process that sends messages into an integration solution. After a time step, it enters the state  $P_0$  from which, with probability 1 the message is sent to the other step  $S_0$ , which goes to another time step, with probability 1 the message is sent to the next step  $T_0$ , and with probability 0.05 the filter task  $T_0$  eliminates free messages, and with a probability of 0.95 the messages follow in the flow.

Another analysis is done according to de Freitas Filho [8] and Sargent [35]. In each scenario, a simulation is analyzed in detail. According to the authors, subtracting the number of messages that entered the integration solution, the messages that are executing in the system and the messages that left by each of the ports, the result would have to be equal to zero.

To prove that the model is valid, it is considered that the simulation model studied presents: (1) three output ports; (2) task that allows to replicate the messages; and (3) each task is running each time you have queues in the slots. In these conditions for each exit port is structured a generic way to verify each scenario.

## 8. Experimental Results

The experiments and scenarios in which the simulations are submitted are described in this section. In order to carry out the simulations, a model of states and transitions is used where each transition occurs a certain probability. The PRISM Model Checker tool is used to make modeling and formal verification of probabilistic models.

### 8.1. Scenarios submitted

For the realization of the simulations, ten scenarios are defined, with entry into the system of 10,000, 20,000, 30,000, 40,000, 50,000, 60,000, 70,000, 80,000, 90,000 and 100,000 messages. In order to increase the efficiency of the obtained results, each scenario was submitted to 25 repetitions. After all, in statistics, an experiment is repeated a large number of times with the same data, following the Big Numbers Law [16]. Empirically, for the simulation experiments that are performed, the population mean is usually obtained with approximately 25 repetitions. The number of repeats is due to the simulation model being a probabilistic model. After the 25 simulations the mean arithmetic, variance and standard deviation were analyzed by the following methods [37]:

- **Arithmetic Mean:** is the most used position measurement. Is observed with the same number of observations. It has its ease as its calculation and as a disadvantage of being greatly affected by extreme values (outliers values). It has the following form:

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (16)$$

where:

$X_i$  = values of variable  $X$

$n$  = number of data

- **Analysis of Variance:** fundamentally, it aims to verify if there is a significant difference between the means and if the factors exert influence in some dependent variable. The analysis of variance is used when deciding whether the observed sample differences are real (caused by significant differences in the observed populations) or casual (due to the mere sample variability). Therefore, this analysis starts from the assumption that chance only produces small deviations, the great differences being generated by real causes. The analysis of variance is the mean of the quadratic deviations of each value in mean relation. The sample variance is given per:

$$S^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} \quad (17)$$

ou

$$S^2 = \frac{\sum_{i=1}^n x_i^2 - \frac{(\sum_{i=1}^n x_i)^2}{n}}{n - 1} \quad (18)$$

- **Standard deviation:** shows how much variation or "dispersion" exists in relation to average (or expected value). A low standard deviation indicates that the data tends to

be Close to the average; A high standard deviation indicates that the data is scattered over a range of values. The standard deviation is the square root of the variance.

$$S = \sqrt{S^2} \quad (19)$$

## 8.2. Analysis of experimental results

In this section the simulation results are presented. The simulation model is equivalent to the integration solution model designed in Guaraná technology. For the simulation, the conceptual model was implemented in the PRISM Model Checker tool. The purpose of the simulation is to evaluate the behavior and predict possible performance problems that the solution may present. The results are obtained through the simulations in different scenarios. The simulation experiments that are performed are obtained with 25 repetitions, [16]. The number of repetitions is due to the simulation model being a probabilistic model.

After the simulations of all the scenarios are performed, Table 4 is presented with a set of results of the averages obtained with the simulation in the PRISM tool for this case study.

Table 4. Message average stored in each slot corresponding to each scenario

Slots	10,000	20,000	30,000	40,000	50,000	60,000	70,000	80,000	90,000	100,000
$s_0$	65,00	73,76	104,52	147,16	132,28	135,46	221,92	215,56	252,12	374,72
$s_1$	18,72	13,28	18,96	16,88	20,88	21,66	21,24	15,32	11,08	14,28
$s_2$	16,88	29,76	15,44	14,16	11,00	18,54	18,76	16,56	14,00	17,76
$s_3$	12,04	20,84	14,28	19,76	13,40	20,87	17,36	21,48	13,40	14,92
$s_4$	80,28	101,96	98,24	84,36	83,24	89,04	96,48	91,08	76,56	115,37
$s_5$	16,48	15,68	22,20	16,56	22,08	27,45	18,48	15,8	15,48	16,96
$s_6$	12,56	19,72	20,36	14,32	12,48	20,25	14,36	10,16	19,64	10,96
$s_7$	12,56	19,72	20,36	14,32	12,48	20,25	14,36	10,16	19,64	10,96
$s_8$	20,08	17,24	21,08	11,28	16,80	17,75	23,64	17,12	19,40	18,12
$s_9$	12,84	15,20	15,80	18,28	15,32	20,91	18,8	11,84	12,96	25,84
$s_{10}$	20,64	13,4	19,68	19,96	14,08	17,25	16,44	15,96	33,08	11,16
$s_{11}$	22,32	14,12	26,20	22,00	21,04	26,70	21,04	19,36	23,28	13,76
$s_{12}$	10,00	8,60	6,64	7,88	5,48	7,541	8,36	8,60	8,720	9,96
$s_{13}$	6,56	10,64	6,60	6,72	9,40	4,458	10,12	4,80	3,920	12,04
$s_{14}$	9,88	6,72	18,16	13,48	12,80	16,66	12,16	8,72	11,32	7,96
$s_{15}$	21,68	17,20	12,48	13,8	12,20	16,75	12,60	10,64	20,92	17,36
$s_{16}$	11,60	14,44	11,80	17,6	20,56	16,12	23,80	19,40	24,80	14,08

Analyzing Table 4, it can be verified that the performance bottlenecks of the simulation system are in slots  $s_0$  and  $s_4$ , in the other slots there are few accumulated messages. According to the results it can be stated that the accumulation of messages in  $s_0$  is a result of the excessive demand of messages that enter the simulation system. That is, the more messages you enter into the system, the greater will be the accumulation in the slot  $s_0$ . This

also occurs because the simulation tool does not work in relation to time, but rather in steps according to the Markov Chain of Discrete Time. Besides, the more messages enter into the system, there will also be an accumulation in slot  $s_4$ . However, this accumulation is not in the same proportion as the slot  $s_0$ . This is due to the messages that are accumulated in the  $s_0$ , waiting to be processed causing a smaller amount of messages circulating through the system.

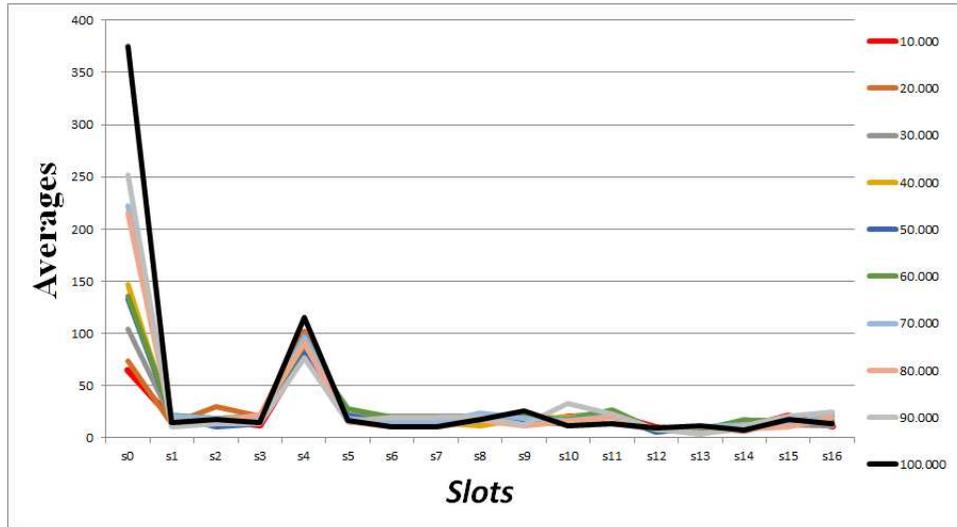


Figure 7. Message average stored in each slot

Analyzing the graphs of Figure 7, it is possible to verify that the performance bottlenecks of the simulation system are in slot  $s_0$  and  $s_4$ , in the other slots there are few accumulated messages. According to the results it can be stated that the accumulation of messages in the  $s_0$  is a result of the excessive demand for messages that enter the simulation system. That is, the more messages you enter the system, the greater the accumulation in the slot  $s_0$ . This also occurs because the simulation tool does not work in relation to time, but rather in steps according to the Markov Chain of Discrete Time. And, the more messages enter into the system, there will also be an accumulation in slot  $s_4$ . However, this accumulation is not in the same proportion as the slot  $s_0$ . This is due to the messages that are accumulated in the  $s_0$ , waiting to be processed causing a smaller amount of messages circulating through the system.

All discrete event simulations are subject to unusual values known as outliers. To identify if there is a discrepancy of values it is necessary that the outliers are calculated for the analysis, as well as highlighting in each graph the values that are outside the upper and lower limits of the standard deviation. The outliers are calculated and analyzed through the Box-Plot graph. It is observed in Figure 8 that Box-Plot graphs are based on the minimum value, first quartile (Q1), median (second quartile Q2), third quartile (Q3) and maximum value. The results of the slots of each scenario are necessary to organize a Box-Plot graph.

Considering a simulation in a queuing system of an  $M/M/1$  model, the occurrence of an outliers, over the variable average number of clients in the queue is due to the number of clients and the time of service. It is observed that if the service of a client takes more than

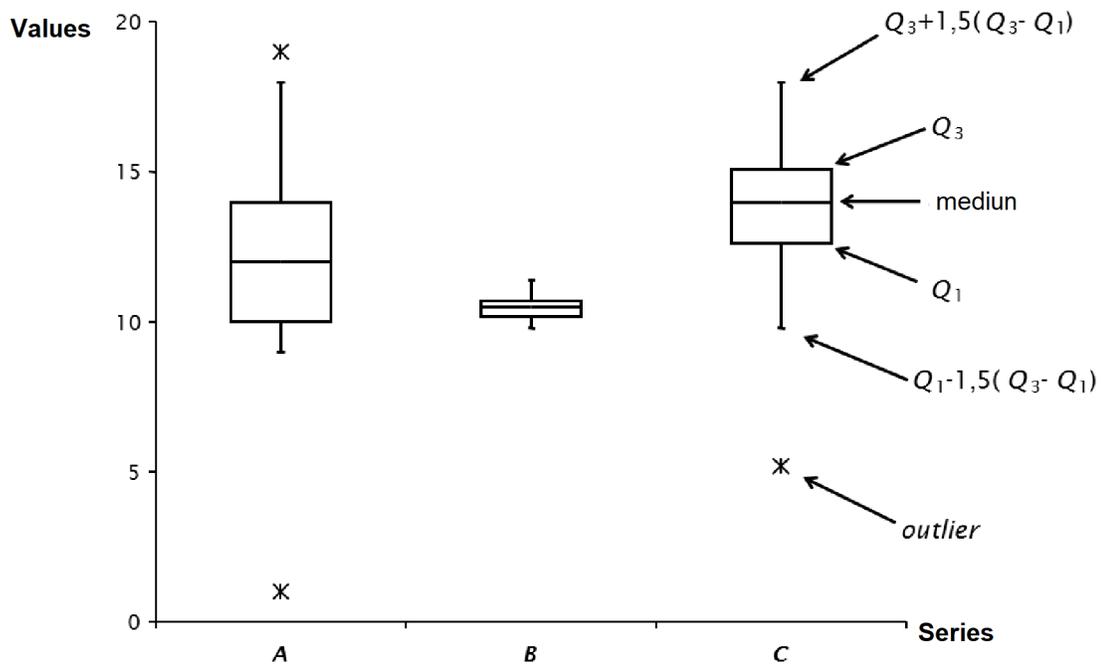


Figure 8. Box-Plot Diagram for Outliers identification.

the average, the amount of clients waiting in the queue increases. This client happens to be considered an outliers. Because it is a simulation of discrete events, it is considered that the occurrence of outliers can be influenced by the average rate of messages that arrive at the simulation system, or a certain task may be consuming specific processing time due to the size of a message in the system.

It can be seen from the results that there are no outliers in slot  $s_0$ , so all the values of the simulations are acceptable. In slot  $s_4$  there are three (12%) top outliers. In the slot  $s_0$  there are four (16%) top outliers, and in slot  $s_4$  there are three lower and five upper outliers. It is observed that in the scenario of 20,000 messages there are 32% of the values obtained with the simulation in slot  $s_4$  are outliers. There are two inner outliers in the slot  $s_0$  and in the slot  $s_4$  there are three lower outliers and two upper outliers, which correspond to 20% of the simulation values.

## 9. Conclusion

The companies are always looking to improve their business processes. Over the years, a company may have acquired or internally developed several applications that were not designed to share data and functionality, that is, they were built without integration concerns. Through the study carried out and presented, it has become possible to realize that there are works in the literature that have used the simulation of discrete events, techniques and tools to analyze systems to predict the behavior and to find the possible performance bottlenecks.

However, it can be seen that there is no evidence of discrete event simulations aiming at the analysis of conceptual models of integration solutions of business applications, using mathematical techniques and simulations. With the analysis of the conceptual model of the integration solution developed in Guaraná technology was developed state diagram between states of the matrix of a Chain Markov. It was noticed that there is equivalence of the elements of an integration solution projected in the Guaraná technology with the elements that make up the Markov Chains. Another common feature between the integration solution presented by Guaraná technology and a system of discrete events is the relationship between its elements and the structure of operation. The characterization of an integration solution with a discrete event system helped the understanding of the integration problem and the identification of performance bottlenecks. In this sense, it was observed that in the discrete event simulations, the variables are equated in a model, whose state changes occur in discrete points of time. The choice of the use of Markov Chains for this study is given, since it is a particular case of stochastic processes and, therefore, used for discrete event systems. This made it possible to study and analyze the formation of queues. Events generate the state transitions represented by a transition matrix. For the simulation the use of the PRISM tool is proposed, which is an application used for the formal modeling of systems. With this simulation approach, we sought to find performance bottlenecks still in the design phase reducing consequently the cost, risk and time to implementation. Thus, integration solutions are seen as discrete event systems and their conceptual models are translated into formal models, which are simulated using techniques and tools known for the simulation of discrete events.

With the results obtained, it was noticed that there is an accumulation of messages in slots  $s_0$  and  $s_4$ . In slot  $s_0$  the accumulation is the result of the messages entries in the system, and the larger the number of messages entering the modeled system, the greater is the accumulation. However, in slot  $s_4$ , the accumulation is the result of the correlator task that precedes this slot. In this case, in order for the correlator task to process the messages, it needs to have in its input slots, correlated messages, from replications of the replicator task. The availability of the integration solution described in a formal model, a stochastic verification model based on discrete event simulation, allowed a rigorous exploration, attributing the verification and analysis of reliable information. For future works, it is recommended to extend the application of the computational simulation technique in the PRISM tool to the mathematical technique Continuous Time Markov Chain (CTMC). The specification of the behavior of CTMC is done in a similar way to a Markov Chain of Discrete Time. The main difference is that command updates are labeled with (positive value) rates, rather than probabilities.

## References

- [1] E. Abrahám, N. Jansen, R. Wimmer, J.-P. Katoen, and B. Becker. *Dtmc model checking by scc reduction*. In *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the*, pages 37–46. IEEE, 2010
- [2] A. K. Ahluwalia and M. Singhal. *Performance analysis of the communication archi-*

- texture of the connection machine. *Parallel and Distributed Systems, IEEE Transactions on*, 3(6):728–738, 1992
- [3] S. Basagiannis, P. Katsaros, A. Pombortsis, and N. Alexiou. *A probabilistic attacker model for quantitative verification of dos security threats*. In *Computer Software and Applications, 2008. COMPSAC'08. 32nd Annual IEEE International*, pages 12–19. IEEE, 2008
- [4] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing networks and markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006
- [5] J. Clément, C. Delporte-Gallet, H. Fauconnier, and M. Sighireanu. *Guidelines for the verification of population protocols*. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 215–224. IEEE, 2011
- [6] M. I. Clune, P. J. Mosterman, and C. G. Cassandras. *Discrete event and hybrid system simulation with simevents*. In *Proceedings of the 8th international workshop on discrete event systems*, pages 386–387, 2006
- [7] P. J. de Freitas Filho. *Introdução à modelagem e simulação de sistemas: com aplicações em arena*. Visual Books, 2001
- [8] P. J. de Freitas Filho. *Introdução à modelagem e simulação de sistemas: com aplicações em arena*. Visual Books, 2001
- [9] G. P. Dimuro, R. H. Reiser, A. C. Costa, and P. Souza. *Modelos de markov e aplicações. VI Oficina de Inteligência Artificial, Pelotas: Educat*, pages 37–59, 2002
- [10] N. J. Dingle, W. J. Knottenbelt, and T. Suto. *Pipe2: a tool for the performance evaluation of generalised stochastic petri nets*. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):34–39, 2009
- [11] D. Dossot, J. D’Emic, and V. Romero. *Mule in action*. Manning, 2014
- [12] M. Fisher, J. Partner, M. Bogoevici, and I. Fuld. *Spring integration in action*. Manning Publications Co., 2012
- [13] R. Z. Frantz, A. M. Reina Quintero, and R. Corchuelo. *A domain-specific language to design enterprise application integration solutions*. *International Journal of Cooperative Information Systems*, 20(02):143–176, 2011
- [14] R. Z. Frantz. *Enterprise application integration: an easy-to-maintain model-driven engineering approach*. PhD thesis, Universidad de Sevilla, 2012
- [15] A. V. P. Gomes and P. Wanke. *Modelagem da gestão de estoques de peças de reposição através de cadeias de markov*. *Gestão & Produção*, 15(1):57–72, 2008
- [16] C. M. Grinstead and J. L. Snell. *Introduction to probability*. American Mathematical Soc., 1997

- 
- [17] G. Hohpe and B. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004
- [18] C. Ibsen and J. Anstey. *Camel in action*. Manning Publications Co., 2010
- [19] M. R. James, V. Krishnamurthy, and F. Le Gland. *Time discretization of continuous-time filters and smoothers for hmm parameter estimation*. *Information Theory, IEEE Transactions on*, 42(2):593–605, 1996
- [20] D. G. Kendall. *Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain*. *The Annals of Mathematical Statistics*, pages 338–354, 1953
- [21] J. A. Kumar and S. Vasudevan. *Variation-conscious formal timing verification in rtl*. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 58–63. IEEE, 2011
- [22] J. A. Kumar and S. Vasudevan. *Formal probabilistic timing verification in rtl*. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(5):788–801, 2013
- [23] M. Kwiatkowska, G. Norman, and D. Parker. *Stochastic model checking*. In *Formal methods for performance evaluation*, pages 220–270. Springer, 2007
- [24] M. Kwiatkowska, G. Norman, and D. Parker. *Prism 4.0: Verification of probabilistic real-time systems*. In *Computer aided verification*, pages 585–591. Springer, 2011
- [25] Y. Kwon and G. Agha. *Scalable modeling and performance evaluation of wireless sensor networks*. In *Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE*, pages 49–58. IEEE, 2006
- [26] D. G. Messerschmitt and C. Szyperski. *Software ecosystem: understanding an indispensable technology and industry*. MIT Press Books, 1, 2005
- [27] J. d. MORAES et al.. *Usando arquétipos e linguagem específica de domínio no desenvolvimento de aplicações ubíquas para o cuidado de saúde pervasivo*. In *CONGRESSO BRASILEIRO DE INFORMÁTICA EM SAÚDE*, volume 13, 2012
- [28] G. Norman, C. Palamidessi, D. Parker, and P. Wu. *Model checking the probabilistic pi-calculus*. In *Quantitative Evaluation of Systems, 2007. QEST 2007. Fourth International Conference on the*, pages 169–178. IEEE, 2007
- [29] D. Parker. *Lectures - probabilistic model checking*. University of Oxford, Department of Computer Science, 2011
- [30] L. Paulevé, M. Magnin, and O. Roux. *Tuning temporal features within the stochastic pi-calculus*. *Software Engineering, IEEE Transactions on*, 37(6):858–871, 2011
- [31] G. Pinheiro. *Teoria de filas e sistemas de comunicação*. *Apostila de Aula. Departamento de Engenharia Eletrônica e Telecomunicações, Universidade Estadual do Rio de Janeiro—UERJ*, 2013

- 
- [32] D. Prado. *Teoria das filas e da simulação*. Belo Horizonte, Nova Lima: Editora FALCONI, 5, 2014
- [33] H. G. Reyes and L. E. C. Barrón. *Simulación y análisis de sistemas con promodel*. Pearson Educación, 2006
- [34] W. Sandmann. *On optimal importance sampling for discrete-time markov chains*. In *Quantitative Evaluation of Systems, 2005. Second International Conference on the*, pages 230–239. IEEE, 2005
- [35] R. G. Sargent. *Verification and validation of simulation models*. *Journal of Simulation*, 7(1):12–24, 2013
- [36] S. Sawicki, R. Z. Frantz, V. Basto Fernandes, F. Roos-Frantz, I. Yevseyeva, and R. Corchuelo. *Characterising enterprise application integration solutions as discrete-event systems*. In *Handbook of Research on Computational Simulation and Modeling in Engineering*, pages 261–289. IGI Global, 2016
- [37] M. F. Triola et al.. *Introdução à estatística*, volume 9. Ltc Rio de Janeiro, 2005
- [38] R. Z. Frantz, S. Sawicki, F. Roos-Frantz, R. Corchuelo, V. Basto-Fernandes, and I. Hernández. *Desafios para a implantação de soluções de integração de aplicações empresariais em provedores de computação em nuvem*. 2014