Chapter

MATHEMATICAL MODEL FOR THE SIMULATION OF AN INTEGRATION SOLUTION FOR APPLICATIONS IN THE ACADEMIC CONTEXT OF UNIJUÍ

Adriana R. Kraisig*, Franciéli C. Welter, Igor G. Haugg, Rafael Z. Frantz, Fabricia Roos-Frantz and Sandro Sawicki
Unijuí University, Department of Exact Sciences and Engineering,
Ijuí, RS, Brazil

Abstract

Companies often acquire or develop applications to support decision-making and improve their business processes. These applications compose the software ecosystem, which is usually heterogeneous and its applications are frequently developed without taking into account integration, thus handicapping their reuse. The Enterprise Application Integration (EAI) area provides methodologies, techniques, and tools for companies to develop integration solutions. The problem addressed in this chapter is to identify the possible performance bottlenecks in the integration solution which deals with the UNIJUÍ University process of re-enrollment, in order they can be minimized before the solution implementation. The occurrence of these possible bottlenecks is a problem, because

*E-mail address: maryshelei@yahoo.com.br (Corresponding author).

if a conceptual model is implemented with bottlenecks, it can generate failures, which increase the costs, time and risks of the solution. In this respect, it is proposed to identify possible performance bottlenecks, using the conceptual model, whereby a formal simulation model is developed using the mathematical formalism from Timed and Colored Petri Nets. It is through the simulation that it seeks to know the behavior of the system, aiming to identify tasks that may represent performance bottlenecks. It was possible to analyze the variable from the simulation: messages average time of stay in the slots. The simulation results of the variable were interpreted and analyzed, identifying the occurrence of performance bottlenecks.

Keywords: Application Integration, Domain-Specific Language, Conceptual Model, Simulation Model, Petri Nets

PACS: 05.45-a, 52.35.Mw, 96.50.Fm

1. Introduction

The use of information systems has become increasingly frequent in business practices in companies. In this sense, an information system can be defined as any set of data and information that are organized in an integrated way, in order to meet the demand and anticipate the needs of the users. In addition, decision support information systems are systems that collect, organize, distribute and make available the information used in this process, in order to provide support to the company's business processes. In this chapter, we consider that an information system is a software designed to support companies in their business processes.

A software ecosystem consists of a set of software solutions that support and automate user activities and transactions that are associated with a social or business ecosystem [2]. According to Pressman [18], application software, or simply application, is a program that solves a specific need of business processes, facilitating decision- making. Therefore, companies acquire or develop applications for efficiency in their decision-making, and this set of applications, which can be acquired from different suppliers or developed in-house, make up their software ecosystem.

Generally, the applications that make up the companies software ecosystem are heterogenous and are developed without taking into account the possibility of reuse. One way to promote communication between these applications is to

use tools and technologies designed to integrate applications. An advantage of the integration of applications is the use of the technological base of the systems already in operation, avoiding a high cost investment in new platforms [9].

The Enterprise Application Integration (EAI) area is related to the development of solutions that solve specific problems of integration of business processes. The EAI seeks to develop methodologies, techniques and tools to create integration solutions that allow the reuse of the applications of the software ecosystem through its integration [14]. The objective of an integration solution is to keep the data and the functionalities of the applications in synchronization or to develop new functionalities from those already existing in such a way that the applications are not altered by the solution [10]. A typical integration solution orchestrates a suite of enterprise ecosystem applications allowing their data and functionality to be shared. Among the various technologies available which enable to design conceptual models of integration solutions are: Camel [12], Mule [6], Spring Integration [7] and Guaraná [10].

Integrating applications is not a trivial task, and developing the solution can involve costs, time, and risk. An integration solution is a new application, which must go through several stages of development, like any other application. According to Sommerville [22], the development of an application basically passes through five stages, namely: specification, design, implementation, testing and evolution. The first step defines what is expected as a result. In the second one, we create the conceptual models of the application, which specify the structure of the application, the data and the interfaces between the components. In the third step, based on the created models, the application code is generated. The fourth step detects errors and verifies the functionality of the application. The last step is to meet the customer's changing needs. Thus, the analysis of integration solutions with the objective of knowing their behavior, still in the design phase, can represent an improvement in the quality of the solutions developed.

Guaraná is one of the technologies that permits to design, implement and execute integration solutions. However, this technology does not offer any tool for the analysis of the conceptual models developed. In this chapter, we present a proposal based on the simulation of discrete events to analyze the behavior of an integration solution, based on its conceptual model developed in Guaraná, focusing on the identification of possible performance bottlenecks.

Simulation is a method that uses a computational mathematical model to enable the study and analysis of the behavior of the system without the need to make changes in the actual system and, therefore, making it possible to pre-

dict a future behavior [21]. Discrete event systems change their state at distinct moments in time, from the occurrence of events. Integration solutions can be characterized as discrete event systems, because when an event occurs all the components involved in the solution consume a certain time of execution. Thus, the occurrence of an event changes the state of the solution. So, for the simulation of the integration solution, a model based on the Colored and Timed Petri Nets was developed, one of several mathematical representations for discrete systems.

The remainder of this chapter has its structure organized as follows: Section 2 presents a brief theoretical framework; Section 3 presents related works; Section 4 details the software ecosystem and presents the conceptual and simulation models for the case study; Section 5 describes the experimentation process adopted, as well as the scenarios and variables, and it also presents the main results and discussion; finally, Section 6 discusses the main conclusions.

2. Background

This section describes some simulation concepts, presents the types of Petri Nets and their main elements, and, finally, introduces the Guaraná DSL technology.

2.1. Simulation

Simulation is a field of research that deals with the experimentation of models, what allows to predict the behavior and performance of real systems. According to dos Santos [5], simulation is the imitation, during a certain period of time, of the operation of a system or a process of the real world. Simulation involves the generation of an abstract situation of the system based on the simulation model, from which one can infer how the real system works.

The behavior of the system is analyzed from a simulation model. This model usually takes the form of a set of constraints related to the operation of the system. These constraints can be expressed through mathematical, logical and symbolic relations between the entities or objects of interest of the system. Once built and validated, a model can be used to investigate a number of questions about the actual system.

For the construction of a simulation model, it is first necessary to define the objectives of the simulation. It is important to consider the performance

measures, that is, the exit variables of interest of the simulation model. When analyzing a simulation model, we should also distinguish three basic elements: the entity, the attribute, and the activity. Any object involved with the model is called an entity. A property of this entity is called an attribute. Any process that generates a change in the model is called an activity. For example, in a banking system modeled as a discrete event system, the clients arriving at the system would be entities; the balance and personal credit available to each customer would be attributes, and the service provided by a specific department of this bank would be an activity.

A simulation model of discrete events is characterized for reproducing the activities involved in the system, in order to predict its behavior and performance, taking into account that each event occurs at a particular moment in time, causing a change of state in the system. It is possible to formulate models of simulation of discrete events in three ways: (a) taking as input the descriptive memory and the description of the process used by the entities of the system; (b) taking as input the complete description of the activities of the entities involved in the system, and (c) defining the changes that may occur in the states at each moment of the event. The simulation model would be implemented using simulation strategies [21].

Figure 1 presents the simulation process, understood as a scientific method, in which, first, one must formulate hypotheses, followed by the development of the model; in the sequence, it is possible to test these hypotheses through the model and analyze if they were validated taking into consideration the results obtained. It is emphasized that the simulation requires a formal language to specify the system to be analyzed.

Figure 1. Scientific Method Applied to Simulation.

The elaboration of a simulation model, according to Paul andBalmer [16], presents three major stages, according to Figure 2. In the first step, it is important to clearly understand the system to be simulated, its objectives and the possibilities of study of a system, such as the decision about the comprehen-

siveness of the model, the level of details and all the hypotheses assumed. After these decisions, the formulation of the conceptual model is carried out. In this step, the input data must also be collected. In the second step, the simulation model is implemented from the conceptual model, through formal language, and then verified, that is, tested to verify if this model meets the objectives of the simulation. In the third and final step, after the verification of the simulation model, the experiments are performed, giving rise to the experimental model. Several tests of the model are performed and the results of the simulation are analyzed. The results might be satisfactory or not.

Figure 2. Stages of Elaboration of a Model.

2.2. Petri Nets

Proposed by Carl Adam Petri in 1962, Petri Nets (MoP) are a mathematical formalism that allows us to model the behavior of systems, describing the relationships between conditions and events, and they allow us to study properties such as parallelism, synchronization, and resource sharing [1]. Petri Net is a modeling tool with a strong mathematical base that allows the representation of parallel, concurrent, asynchronous and non-deterministic systems [4].

Petri Nets are two-part graphs, because they present two types of nodes, called places and transitions. Places are equivalent to system state variables and are represented graphically by circles or ellipses. The transitions are equivalent to the actions performed and are represented by bars or rectangles. Graphs are always directed from places to transitions and/or transitions to places. These places and transitions are connected by arcs. The arcs are represented graphically by arrows that indicate the flow direction of the modeled system.

Figure 3 presents the basic elements of a Petri Net: places, transitions and arcs. When an arc connects a place to a transition, it is called the input arc, but when it connects a transition to a place, it is called the output arc. In the figure, two places are represented: P0 and P1; a T0 transaction; and two arcs: an input arc, connecting P0 to the transition T0, and an output arc, interconnecting the transition T0 to the place P1. The value of each state variable is determined

by the number of tokens in each place. Tokens are represented graphically by black dots, as shown in Figure 3. When transitions are triggered they consume tokens from their input places and can or not generate other tokens in their places of exit. In this way, transitions remove or add tokens in network locations whenever a shot occurs. However, a transition will only be able to be triggered if its input places have the minimum number of tokens, determined by the weight of the input arc.

Figure 3. Basic Elements of a Petri net (Francês [8]).

A Petri Net can be formally defined as a quintuple R = (P, T, A, V, K), composed of the set of places P, the set of transitions T, the set of the arcs connecting the places to the transitions or the transitions to the places, the valuation or weight of the arcs represented by V and the set of capacities of the places K. There are several extensions for this type of Petri Net, named Place/Transition, among which we highlight: Colored, which allow their tokens to be individualized, and Timed, which deal with temporal aspects.

Colored Petri Nets are characterized for reducing the size of the model, allowing the tokens to be individualized. Thus, different processes or resources can be represented in the same network. Colors do not just mean colors or patterns, they can represent complex data types [13]. Like the Place/Transition networks, Colored ones are also directed and split graphs. However, instead of integer weights, they are associated with the arcs inscriptions that determine how many and which tokens are to be removed or added to the places of entry and exit, in the triggering of a transition. Inscriptions, called guards, can also be associated with transitions. Guards restrict the enabling of the transition under certain conditions. The initial state of a Colored network is also determined by inscriptions associated with places. Each inscription is, in general, an expression constructed from previously defined constants, variables and operators. In addition, the Colored networks have a set of declarations to indicate the nature of the elements mentioned in the various inscriptions, similarly to the declaration of variables in any programming language.

Timed Petri Nets are characterized for allowing the association of time to their components [17]. This association may be realized in several ways, like:

- The time associated with places. The tokens (after triggering) will be available to trigger a new transition after a certain time which is associated with the place;
- The time associated with the marks. The time indicates when the mark will be available for triggering;
- The time associated with transitions. It uses times associated with the transitions.

This type of network can be deterministic or stochastic. The deterministic ones indicate absolute times relative to the execution of the corresponding events. The stochastic ones allow us to consider uncertainties at the instants of the execution of system events by associating them with probability functions.

2.3. Guaraná DSL

Guaraná technology provides a domain-specific language allowing to design integration solutions at a high level of abstraction using concrete graphical syntax and intuitive modeling concepts. This modeling language is based on the patterns of integration documented by Hohpe andWoolf [11]. The components of this language allow to represent all integration processes and their communication ports.

Guaraná DSL technology consists of five main elements: application, integration process, ports, slots and tasks. In Table 1, it is possible to observe how each component is represented graphically. Processes are blocks that group tasks together. The processes have ports, which are responsible for communicating with the applications. These ports might be: (a) input, (b) output, (c) request or response.

The slots are equivalent to temporary storage units. They enable asynchronous processing of messages that participate in a process, allowing for independence between tasks. In this context, the slots interconnect the tasks, so each slot receives the message already processed from the previous task and leaves it available to be processed by the next task. Once a message is processed and dispatched to the next slots, the task is ready to process another message [9].

The ports allow the sending and receiving of messages. Thus, the gateway sends a message to a slot and the slot makes it available for a task. The exit port always reads a message from one slot and leaves it available to the next element

Table 1. Graphic Technology Notation Guaraná DSL (Frantz et al. [10]).

in the stream. Request and response ports allow communication between the process and the integrated applications.

A message is made up of header and body. The header contains predefined properties, such as message identifier, correlation identifier, and message priority. The structure of the messages depends on the integration solutions in which they are involved. In this sense, the message is considered an abstraction of a part of the information that is exchanged and transformed.

Guaraná DSL allows the modeling of different tasks, represented graphically by an icon, which is associated with the function that this task performs. The tasks that form the processes are their main element, and they are responsible for the processing and modification of the messages. Thus, a task is responsible for reading a message from the input slot, processing it, and writing it in the next slot.

Integration solutions can be characterized as a discrete events system. This way, it is possible to translate the conceptual model of an integration solution into a Petri Net. Table 2 shows the equivalence of the elements of Guaraná DSL (tasks, slots and messages) with the elements of the Petri Net (transitions, places and tokens).

Guaraná has as constructors: messages, tasks, slots, ports and integration process and Petri Nets have: tokens, transitions and places. From this, it is possible to represent an integration solution through the use of the Petri Nets mathematical formalism, allowing the development of a simulation model, which can be simulated and analyzed.

Table 2. Equivalence of the Elements (Roos-Frantz et al. [20]).

In Petri Nets, the triggering of a transition changes its state [20]. When the transition is triggered, the tokens connected to the input arcs are removed and added to the places connected to the output arcs. A transition is enabled if the number of tokens determined by the input arcs exist in their respective places. The amount of tokens generated by the output arcs is not necessarily the same of that removed by the input arcs.

In an integration solution, incoming messages wait in the slot while the task is executing. The messages follow a policy of discipline to be performed by the tasks. After being processed, the messages follow the flow of the integration solution. The slot discipline, in an integration solution, defines the order of message processing. Figure 4 shows the equivalence of state change in a solution modeled with Guaraná and a state change in a solution of Petri Nets.

Figure 4. Equivalence of the Exchange of States (Roos-Frantz et al. [20]).

This chapter characterizes the integration solution that deals with UNIJU University re-enrollment process as a discrete event system, using the mathe-

matical formalism of Colored and Timed Petri Nets. In Petri Nets, places along with arcs perform similar functions to tasks and slots. The arcs are responsible for making the connection between places and transitions or transitions to places, indicating the flow of messages.

3. Related Work

In this section we present some related works that use Petri Nets for the modeling and simulation of discrete events systems with the objective of analyzing the performance and performance bottlenecks of a system.

The work presented by Yamada et al. [23] simulates the stages of operation of a sugarcane industry using Petri Nets. The industry system includes actions of receiving, unloading and moving raw material to the extraction process. With the simulation, we aim to analyze proposals for optimization and identification of performance bottlenecks without interfering with the actual system operation. The processes of receiving the sugarcane and feeding of the grinding center have discrete characteristics, and the time of each execution interferes in the performance of the system. The construction of the model is made from the study of the activities of the process. Parallelism of activities was verified with the simulation of the model developed, what causes an overload in the feeding of the grinding center in situations of successive arrivals of raw material. Thus, the need for the synchronization of activities was felt, since the unloading resources are shared among the sectors. The model does not consider its deterministic and stochastic characteristics, although it has denoted the variables of the system. The author suggests the use of Timed Petri Nets to differentiate each step of the process and consolidate the validation of the model.

The work presented by Roos-Frantz et al. [20] originates from a case study, based on a real integration problem that deals with a project to automate the registration of new users in a single repository of the County Council of Huelva, Spain. This proposal presents a simulation model performed with Stochastic Petri Nets developed from the conceptual model of the integration solution. The authors consider that the simulation can improve the quality of integration solutions developed with Guaraná technology and also that from the simulation one can identify information related to the performance of the solution.

The work presented by Ramos andde Oliveira [19] uses Colored Petri Nets for the specification and formal verification of an intelligent tutor system (STI) model, which uses problem-based learning (PBL) as a pedagogical strategy. The

specification and the formal verification permit to verify if the planned functionalities of the pedagogical model are realized, before the stage of implementation of the system. Experiments indicate overall consistency and benefits of the proposal. The work was carried out using CPN Tools. According to the authors, the proposal presents as benefits the possibility of specifying, verifying and simulating the main functionalities of the system. In addition, the simulation allows to identify new alternatives that the system can offer, such as: help, orientation, verification of the model operation and analysis of system performance.

The work carried out by Miyagi et al. [15] presents the modeling of health systems with an interpreted Petri Nets approach, applied to the outpatient services of Hospital das Clnicas de So Paulo (HC). The modeling aims to study the flow of patients seeking outpatient care, non-urgent patients and, mainly, patients seeking care for the first time, with the purpose of helping the decision making by the system's management and analyzing the evolution of the system and the behavior of the sectors. The model was developed through the PFS (Production Flow Schema) and MFG (Mark Flow Graph) methodologies, Petri Nets interpretations, which show the resources involved and the movement of the system components (people, equipment, information) in which the details of the system are gradually being inserted into the model for a better understanding, considering the discrete evolution of the system. The simulation of the model considered suggestions for changing the flow of patients, identified the possibility of eliminating some processes that overloaded sectors of the system and suggested new processes.

Finally, the work by Carvalho et al. [3] proposes a multifunctional work performance analysis model in U-shaped production lines using the mathematical formalism of the Timed Petri Nets. The problem approached concerns the relation of time of execution of the tasks in the workstations, evaluating the time of each operation that composes the task of a certain station, the formation of intermediate stocks between the workstations, and their interference in the productive flow of the system . The model simulation analyzes the time of execution of each station under a scenario that determines a construction goal of 67 pieces in a time interval of 60 minutes. The investigation of the difference in the execution time of each process can result in the formation of intermediate stocks due to the accumulation of semi-finished products among the workstations and lead to idleness, and ultimately compromise the production goal. As an alternative solution are analytically determined the maximum load of each intermediate stock and the redistribution of the operators in nine sectors. With

these modifications, improvements in production goals were observed at the determined times.

4. Case of Study

The proposal presented in this chapter was validated from a case study that presents a real integration problem modeled with Guaraná DSL. The integration solution was modeled based on the need to improve the service to the students of UNIJU University. The purpose of this integration solution is to automatically provide information regarding the re-enrollment possibilities of each student, by generating a list containing all the subjects which the student has not yet attended and which will be offered in the next semester.

4.1. Software Ecosystem

The proposed integration solution involves five applications: Portal, Student Registration, Subjects Registration, Billing System and E-mail Server. In this context, each application runs on different platforms. The Portal, Student Registration, Subjects Registration and Billing System are developed in-house at the university. These applications were not developed with the objective of being integrated, so they need to be integrated through an integration solution in order to be able to collaborate and support the process of re-enrollment. The E-mail Server offers POP3 and SMTP interfaces. Portal provides information for students, such as the subjects already taken and the percentage of the course taken. In addition, it carries out processes that enable the student to use the Portal for checking the subjects they want to study in the next semester. Student Registration has a database with information such as name, student ID and course. Subject Registration has a database that contains information such as course name, linked course, semesters, available classes and date. When a student completes the enrollment or reenrollment, through the Portal, it is possible, with the information obtained in the Student Registration, to consult the Subjects Registration, which subjects the student has not yet completed and which subjects will be offered in the next semester. With this information, it will be possible to generate a list of possible subjects to be studied in the next semester. From this subjects list, it is possible to consult the price of the subjects in the Billing System. This list can be sent to the student by e-mail using the E-mail Server.

4.2. Conceptual Model

The conceptual model developed with Guaraná DSL, presented in Figure 5, represents the integration solution proposed for the integration problem described. The solution takes information from the Portal through port P1, which reads messages containing student requests. The T1 task filters the incoming messages, accepting only re-enrollment requests that were made in the re-enrollment period. After that, the accepted messages will be directed to Student Registration through port P2, to obtain information regarding the student who made the request. P3 task replicates the messages and sends a copy to port P3, by which is received from application Subject Registration the list of the subjects in the course that the student is enrolled. Task T6 enriches the second copy performed by task T3, correlating with the information returned by Subject Registration. Task T7 has the function of discarding the subjects that the student has already attended and the disciplines that will not be offered in the semester. Then, task T8 replicates the message again. A copy is sent to port P4, from which the price of the subjects returned by the Billing System application will be obtained. Task T11 enriches the second copy performed by task T8, correlating with the information received from the Billing System. First, the messages are directed to task T12, which performs a filtering in order to avoid that the messages are sent to the students without being complete. Then the messages will be directed to port P5 which, through the E-mail Server application, forwards to the student the list of subjects with their prices.

4.3. Simulation Model

The development of the simulation model followed the equivalences between the two modeling languages presented in Table 3. Each component of the integration solution was replaced by the corresponding Petri Net component. Figure 6 shows the simulation model in Colored and Timed Petri Nets developed in the CPN Tools software. The simulation model considers messages with different sizes, which imply different times of processing. In order to differentiate the messages, a function for each type of message was defined, safeguarding messages with different processing times.

Since the messages are equivalent to the tokens, the task is represented by an input slot, a transition and an output slot, which represent, respectively, input, processing and output. T_CTRL transition inserts tokens into the system. P1 (asynchronous) transition models the input port. The transitions P2

Figure 5. Conceptual Model.

(synchronous), P3 (asynchronous), and P4 (asynchronous) represent the request ports for Student Registration, Subject Registration and Billing System, respectively. P5 (asynchronous) transition models the output port for the E-mail Server application. The simulation model was built following the organization and connection of the tasks, the ports arrangement and the direction of the messages flow.

The filter tasks, represented by the TA, TAA and TAAA transitions, have the function of removing messages from the stream according to the filtering criterion defined. These tasks were defined by a Boolean function, which returns a true value if the value of variable a is 19 or lower and false if it is 20. It means that 95 percent will be true and 5 percent will be false, working as a filter. We highlight that the TA transition is equivalent to the removal of tokens representing messages sent out of time. TAA transition removes from the flow the tokens corresponding to the messages which contain the subjects the student has already taken and the disciplines that will not be offered in the semester. TAAA transition filters incomplete messages by removing the corresponding tokens. The task translator, represented by T2, T4, T9 and T13, has the function of converting the message into a format that the application understands.

The task replicator, represented by T3 and T8, is used to replicate tokens for all their outputs. The task correlator, represented by T5 and T10, is used to locate in the flow the correlated messages that must be processed together. The

Table 3. Ratings Comparison Table.

transitions are only triggered when there are tokens in the places of entry, referring to the transitions, to project tokens together in their places of exit. The task content enrichment represented by T6 and T11 receives correlated messages, combining them into a single one. Transition T6 is triggered when there is at least one token at places S9 and S10 and only one token at place S11. Transition T11 is triggered when there is at least one token in places S17 and S18, and only one token at place S19, representing the task functionality.

5. Experiments

One of the main steps in the study of a system through simulation is the experimentation, in which we obtain the measures of the variables to be observed under different operating scenarios of the system. In this research, we took into account the variables average time of permanence of the messages in the slots

Figure 6. Formal Model with Colored and Timed Petri Nets.

and maximum and average size of the slots. The experiments were carried out using six different scenarios, in order to identify where the possible performance bottlenecks of the integration solution are formed.

5.1. Description

To represent messages with different sizes, the TAM set was defined with the values in 1, 5, 10, 15 and 20 kB. The TAM set is defined as timed because it represents the processing time of each message. The variable t is an element of this set.

For the processing time of each message were created 5 functions f(t) to differentiate the time relative to the message type. In the case:

- The 1 kB message randomly takes a time from 8 to 12 units of time;;
- The 5 kB message randomly takes a time from 40 to 60 units of time;;
- The 10 kB message randomly takes a time of 80 to 120 units of time;
- The 15 kB message randomly takes a time from 120 to 180 units of time;
- The 20 kB message randomly takes a time of 160 to 240 units of time.

The input port of the messages represented by the T_CTRL transition has the function of inserting the tokens in the system. The output port task acts contrarily to the input port, sending messages to the integrated application, its Petri Net equivalent graph indicates flow of tokens outwards the process. The task request-and-response port (P2, P3, and P4) requests information from external applications which provide the responses.

For the representation of a synchronous port P2, it was necessary to create an Anti-S3. This Anti-S3 permits that only1 token exist at a time in place S3.

Regarding the filter task, it was necessary to create the set F, which consists of integers between 1 and 20, including 1 and 20. With the creation of the set (F), the variable f, which is an element of this set. Later, the filter function was created, which is a Boolean function that returns a true value if the value of a is 19 or lower, and false if it is equal to 20. This means that 95 percent will be true and 5 percent false, functioning as a filter. The filter task (TA, TAA and TAAA) has the function of removing messages from the flow of the integration solution according to the defined filtering.

The task correlator is represented by a transition, T5. Place C1 sends a token to place C2. When there is a simultaneous existence of three tokens, a token in place S6, a token in place S8, and a token in place C2, T5 transition is enabled. Likewise, it occurs with T10 transition. Place C3 sends a token to place C4. When there are three concurrent tokens, a token in place S14, a token in place S16, and a token in place C4, the transition is enabled.

The task translator (T2, T4, T9 and T13) has the function of transforming the message into a format that the next application understands, as the messages are equivalent to the tokens, the task is represented by an input slot, a transition and an output slot, which represent the input, the processing and the output of a task, respectively.

The task replicator is used to replicate messages to all its outputs. This task is represented by T3 and T8 transitions.

The task content enrichment receives correlated messages and combines them into a single one, as represented in the equivalent graph in Petri Nets. T6 transition triggers when there is at least one token at places S9 and S10, and only one token is placed in place S11. The same fact occurs in the T11 transition, which triggers when there is at least one token at places S17 and S18, and only one token is placed in place S19 representing the functionality of the task.

At each transition, the time associated to the token is increased, and this increment represents the sum of time that the message remains in the previous place with the time that the message takes to be processed by the transition. The first token arrives at time 0 and at each transition it adds 200 units of time.

Monitors of the type Mark Size were created to obtain the data. Each monitor is related to a slot, which generates a log that records the number of tokens in each slot and also provides the length of time that each message stays in each slot. Therefore, it allows to answer the variable average time of permanence of messages in the slots.

The slots are considered FIFO. There are synchronous and asynchronous ports. After that, we analyze what happens to the average time of permanence of messages in the slots.

The experiments were carried out in 6 different scenarios. In all scenarios we assume that the tasks processing times are equal and there are 5 message sizes (1, 5, 10, 15 and 20 Kb) represented by different tokens. Here are the different scenarios:

- Scenario 1: every 25 units of time, the T_CTRL transition launches a

token in the network entry;

- Scenario 2: every 50 units of time 1 token is launched at the network entry;
- Scenario 3: every 100 units of time 1 token is launched at the network entry;
- Scenario 4: every 200 units of time 1 token is launched at the network entry;
- Scenario 5: every 400 units of time 1 token is launched at the network entry;
- Scenario 6: every 800 units of time 1 token is launched at the network entry.

The 6 scenarios used to perform each experiment consider an input load of 10.000 tokens (randomly chosen among possible values). The time increment represents the interval in which token entries occur in the system. The stop criterion in all scenarios occurs when you finish injecting the 10.000 tokens. In each scenario were collected data for the variable average time of permanence of messages in the slots, which represents the time that the different types of messages remained in each slot. Thus, if a message stays in a particular slot for too long, it indicates delay, creating accumulations that cause bottlenecks.

5.2. Analysis of Results

In this section, we present the results of the variable average time of permanence of the messages in the slots in the six scenarios:

In scenario 1, according to the graph of Figure 7, it can be noted that it takes longer to process the messages in slot S2 because there is a smaller input interval, in this case, 25 units of time. This longer time is due to the fact that the request port P2 is synchronous. While the database is processing a request from external application, the synchronous port does not accept the input of new messages. A synchronous port makes one request at a time, that is, it makes a request and waits for the response for this request before moving on to the following. It differs from an asynchronous port, which makes several requests to the application without having to wait the response from a request to make

the next one. The longer time found in the slots S6 and S14 is due to the need of using the correlator, which causes a delay because it must wait for messages that have the same identity to correlate. Another relevant fact is that larger messages take longer to process.

Figure 7. Average length of time for messages in slots, scenario 1.

In scenario 2, according to the graph of Figure 8, it can be noted that it takes longer to process the messages in slot S2 because there is an input interval of 50 units of time, that is, messages are injected every 50 units of time. This longer time is also due to the fact that the transition which represents request port P2 is synchronous. The longer time found in slots S6 and S14 is due to the need of using the correlator, which causes a delay because it must wait for the messages that have the same identity to correlate. Larger messages take longer to process.

In scenario 3, according to the graph of Figure 9, it can be noted that it takes longer to process messages in slot S2 because there is an input interval of 100 units of time, that is, messages are injected every 100 units of time. This longer time also occurs because the transition which represents request port P2 is synchronous. The longer time found in slots S6 and S14 is due to the need of using the correlator, which causes a delay because it must wait for the messages that have the same identity to correlate. Larger messages take longer to process.

In scenario 4, according to the graph of Figure 10, it can be noted that it

Figure 8. Average length of time for messages in slots, scenario 2.

Figure 9. Average time for messages to stay in slots, scenario 3.

takes more time to process the messages in slot S2, because there is an input interval of 200 units of time, that is, messages are injected every 200 units of

time. It is denoted that the processing time is decreasing in comparison to the previous scenarios, because the message input range is larger. This longer time is also justified because the transition that represents the request port P2 is synchronous. The longer time found in slots S6, S8, S14 and S16 is due to the need of using the correlator, which causes a delay because it must wait for messages that have the same identity to correlate. Larger messages take longer to be processed.

Figure 10. Average time for messages to stay in slots, scenario 4.

In scenario 5, according to the graph of Figure 11, it can be noted that the processing time of messages in slot S2 is normal compared to the other scenarios. This is because the input interval is 400 units of time; that is, messages are injected every 400 units of time. The longer permanence time found in slots S6, S8, S14 and S16 occurs because of the need to use the correlator, which ends up causing a delay because it must wait for the messages that have the same identity to correlate. Larger messages take longer to process.

In scenario 6, according to the graph of Figure 12, it can be noted that the processing time of the messages in slot S2 is normal compared to the other scenarios. This is because the input interval is 800 units of time; that is, messages are injected every 800 units of time. The longer permanence time found in slots S6, S8, S14 and S16 is due to the need of using the correlator, which leads to a delay, because it must wait for messages that have the same identity to correlate.

Figure 11. Average length of time for messages in slots, scenario 5.

Larger messages take longer to process.

Figure 12. Average time for messages to stay in slots, scenario 6.

A relevant fact observed in all scenarios is that the larger messages remain

longer in the slots because 5 functions f(t) have been created to differentiate the relative time to the message type. The 1 kB message takes randomly a time of 8 to 12 units of time. The 5 kB message takes randomly a time of 40 to 60 units of time. The 10 kB message takes randomly a time of 80 to 120 units of time. The 15 kB message takes randomly a time of 120 to 180 units of time. The 20 kB message takes randomly a time of 160 to 240 time units.

6. Conclusion

Companies often acquire or develop new applications to support the decisions that will be taken and to improve their business processes. Companies have their own software ecosystem, which is usually heterogeneous and consists of different applications. Most of these applications were designed without taking into account the possibility of being reused. The goal of an integration solution is to keep the data and functionality of existing applications in sync in a way that these applications are not altered by the solution. In order for the applications to collaborate with each other, companies can use integration solutions. Its correct functioning results in success, agility and dynamism in the business processes of companies. The behavior analysis and the identification of possible performance bottlenecks in application integration solutions often involve their implementation for further execution and testing. As the solutions are built, it involves costs (time, resources) and risks of failures that tend to be high, hampering the proper functioning of integration solutions in situations where large demands for information are processed.

The search for reliable and quality integration solutions presupposes an indispensable analysis of their behavior. According to the literature review, an integration solution in business application can be characterized as a system whose model is classified as stochastic, dynamic and discrete. The classification of the integration solution used in this research obeys the characteristics of a discrete event system. Therefore, it was possible to use a mathematical model along with discrete event simulation techniques to analyze the behavior of an integration solution that deals with the re-enrollment process of Uniju University, confirming the initial hypothesis.

Afterwards, a simulation was performed with CPN Tools, which allows the modeling, analysis and simulation of discrete event systems. To carry out the simulation, it was necessary to start from a conceptual model of the integration solution, that is, the case study. After that, the transcription of the elements

for the Colored and Timed Petri Nets was carried out and a formal simulation model was proposed, which was submitted to experiments to obtain data. From the simulation it was possible to analyze the variable duration of the messages in the slots. The results of the simulation for this variable have been interpreted and analyzed, identifying the occurrence of possible performance bottlenecks in different scenarios.

As for the results, it was observed that when the time of entrance of the messages in the system increases, consequently, there is a longer interval for the processing of the messages in the transitions. In addition, larger messages take longer to process in relation to smaller messages. From this, it can be concluded that the formation of bottlenecks is more evident when the input gap is smaller. The bottlenecks are located in the first slots because P2 port is synchronous and in the correlator slots. This way, the waiting time becomes a bottleneck when it presents a disproportionate accumulation of messages waiting to be processed in comparison to the general state of the system.

This study permitted to analyze the behavior of the solution in different scenarios, and it suggests the occurrence of performance bottlenecks in the integration solution. For this research, the use of simulation in the design phase showed to be important because it made possible to identify in which scenarios and conditions performance bottlenecks tend to occur.

Acknowledgments

The research work on which we report in this chapter was supported by the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES) and the internal Research Programme 2015/17 at UNIJUI University.

References

[1] N. M. C. Barroso, J. M. Soares, G. C. Barroso, J. C. M. Mota,andH. B. Neto. *Modelagem de conceitos e processos matemáticos por redes de petri coloridas: o caso da integrabilidade de funções reais. Bolema*, 27(45):75, 2013. Modeling of concepts and mathematical processes by petri nets: the case of integrability of real functions.

[2] J. Bosch. *From software product lines to software ecosystems.* In*Proceedings of the 13th International Software Product Line Conference*, pages111–119, 2009.

[3] H. J. R. Carvalho, A. R. Yoshizawa, H. L. J. Pontes,andA. J. V. Porto. *Análise de desempenho do trabalho multifuncional em linhas de produção, em forma de U pela modelagem e simulação usando Redes de Petri Temporizadas.* In*XXXVII Simpósio Brasileiro de Pesquisa Operacional*, pages109–121, 2005. Performance analysis of multifunctional work on U-shaped production lines by modeling and simulation using timed Petri nets, in The XXXVII Brazilian Symposium of Operational Research (SBPO).

[4] J. Desel andW. Reisig. *The concepts of petri nets. Softw. Syst. Model.*, 14 (2):669–683, May2015.

[5] M. P. dos Santos. *Introdução à simulação discreta. Rio de Janeiro: UERJ*, 1999. Introduction to Discrete Simulation.

[6] D. Dossot, J. D'Emic,andV. Romero. *Mule in action.* Manning, 2014.

[7] M. Fisher, J. Partner, M. Bogoevici,andI. Fuld. *Spring Integration in action.* Manning Publications Co., 2012.

[8] C. R. L. Francês. *Introdução às redes de petri.* Technical report, Laboratório de Computação Aplicada, Universidade Federal do Pará, 2003. Introduction to Petri Nets.

[9] R. Z. Frantz, R. Corchuelo,andF. Roos-Frantz. *On the design of a maintainable software development kit to implement integration solutions. Journal of Systems and Software*, 111:89–104, 2016.

[10] R. Z. Frantz, A. M. R. Quintero,andR. Corchuelo. *A domain-specific language to design enterprise application integration solutions. International Journal of Cooperative Information Systems*, 20(02):143–176, 2011.

[11] G. Hohpe andB. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley, 2003.

[12] C. Ibsen andJ. Anstey. *Camel in action.* Manning Publications Co., 2010.

[13] K. Jensen andL. M. Kristensen. *Coloured petri nets: Modelling and validation of concurrent systems.* Springer Publishing Company, Incorporated, edition1st, 2009.

[14] D. S. Linthicum. *Enterprise application integration.* Addison-Wesley Professional, 2000.

[15] M. M. Miyagi, P. E. Miyagi,andM. Kisil. *Modelagem e análise de serviços de saúde baseados em Redes de Petri interpretadas. Production*, 11(2):23–39, 2001. Modeling and analysis of health services based on interpreted Petri nets.

[16] R. J. Paul andD. W. Balmer. *Simulation modelling*. Chartwell-Bratt, 1993.

[17] L. Popova-Zeugmann. *Time and petri nets.* Springer, 2013.

[18] R. S. Pressman. *Software engineering: A practitioner's approach.* McGraw-Hill, Inc., 2009.

[19] E. S. Ramos andJ. M. P. de Oliveira. *Especificação e verificação formal de um modelo de STI-PBL por Redes de Petri Coloridas. Revista Brasileira de Informática na Educação*, 17(03):53–66, 2010. Specification and formal verification of a STI-PBL model by Colored Petri Nets, in The Brazilian Journal of Computers in Education.

[20] F. Roos-Frantz, M. Binelo, R. Z. Frantz, S. Sawicki,andV. B. Fernandes. *Using Petri Nets to enable the simulation of application integration solutions conceptual models.* In*International Conference on Enterprise Information Systems*, pages87–96, 2015.

[21] S. Sawicki, R. Z. Frantz, V. M. B. Fernandes, F. Roos-Frantz, I. Yevseyeva,andR. Corchuelo. *Characterising enterprise application integration solutions as discrete-event systems.* In*Handbook of Research on Computational Simulation and Modeling in Engineering*, pages261–288. IGI Global, 2016.

[22] I. Sommerville. *Software engineering*. Addison-Wesley Publishing Company, edition9th, 2009.

[23] M. C. Yamada, A. J. V. Porto,andR. Y. Inamasu. *Aplicação dos conceitos de modelagem e de Redes de Petri na análise do processo produtivo da indústria sucroalcooleira. Revista Pesquisa Agropecuária Brasileira. Brasília*, 37(6):809–820, 2002. Application of modeling and Petri net concepts in the productive process of the sugarcane industry, in Brazilian Journal of Agricultural Research.