Petri nets in engineering

It is also commonly acknowledged that Petri nets can be used in engineering applications. Petri net models are an effective way to address the design, development, and validation phases of systems. Applications that are covered include dataflow-computer systems design workflows and process-driven systems, fault-tolerant systems Reader systems, industrial control systems, distributed systems, distributed databases, concurrent and parallel programming and systems, and just to name a few. Within the Petri net framework, it is possible to study properties of practical interest, like fairness in the execution of tasks, deadlock avoidance, state reachability, and process interlocking, among others. depicts a very basic and conceptual communication mechanism, for instance. The act of communication is examined from the viewpoint of the transmitting process. A sender process uses the output buffer to transmit a message, then stops working while it waits for the input buffer to acknowledge it. While a receiver process waits for an input message, it becomes blocked. When a message is received, the receiving process reads it and uses the input buffer to deliver an acknowledgment. The process resets its logic when the communication act is accomplished so that it is prepared for the subsequent communication. This model might be expanded to incorporate broken communication channels, which might lose the messages, have acknowledgement timeouts, or have other useful properties. Petri net models have been used extensively in the research and design of communications protocols. An illustration of a communication protocol using Petri nets. Process 1 uses the output buffer to deliver a message and uses the input buffer to wait for a response. Input buffer sends an acknowledgment after Process 2 reads the message from the output buffer. Following the conclusion of the communication protocol, the both processes restart their logics to get ready for the next communication act . There are some extensions to the Petri nets to handle specific aspects of different engineering problems . Some of the extensions add structure and information to the tokens, transitions, and places of a net . These extensions allow the construction of models that are quite compact compared to the models obtained with the traditional approach. These models are called Colored Petri Net (CPN) . As an illustration, depicts a CPN for a task scheduling problem . The structure of the net represents the different stages of the working processes in a distributed multitasking environment. The left-hand side of the net structure represents the stages that the processes perform to acquire a job . The right-hand side represents the stages that the working processes perform to release the resources and update the state of the overall scheduling problem . The network's structure serves as a metaphor for the many stages of a distributed multitasking environment's operating operations. The stages that the processes go through to obtain a job are represented on the left side of the net structure. The working procedures that release resources and update the state of the overall scheduling challenge are shown on the right side. The model in design the can be simulated to examine how various scheduling policies function under various workload scenarios. For a fixed number of tasks, it is feasible to approximation the optimal number of processes needed by the scheduling issue. Furthermore, given a rate of task increment over discrete time intervals, it is possible to examine the best rate of working process increment . A model of a task scheduling issue on a colored Petri net. The network's structure serves as a representation of the many stages of a multitasking environment's processes. To obtain the tasks that must be completed, the working processes compete with one another. Each token is a composite unit that contains data about how the working process is progressing. This model's simulation enables us to examine how various *scheduling policies perform. With the rise of cloud computing and the vast amounts of data in social networks recently , the machine learning techniques and the methods related to the data analytics are essential tools in the study and investigation of the big data. There are several proposals to allow the Petri net models learn some kind of fuzzy reasoning and decision making. Similar approaches as that of the supervised and unsupervised learning have been addressed . shows a Petri net representing a workflow pattern for a customer reclaim system . The customer may initiate a request at any time. Two activities are launched in parallel once a request is in the system . First , a ticket check process is executed. Second, an examination of the request is performed . At this point, based on the machine learning, data analytics and/or l statistical learning mechanisms, guards for the transitions b and c are constructed. The guards allow deciding when it is more convenient to execute an in-deep examination process or a casual examination process . On the one hand, it saves time by executing a casual examination when the guard determines that it is more likely that the characteristics of the request are that of a genuine customer request . The customer may initiate a new request at any time. Two activities are launched in parallel for every request. One activity is to check the ticket. The other is to examine the request . At this point, a guard for transitions B and C is built using machine learning and data analytics. The next step is a decision-making procedure, after which the request may be granted, rejected, or resubmitted. The automatic code generation is another significant field of engineering where Petri nets have been employed successfully. The development of the system software supporting them has grown more difficult due to the exponential rise of cloud computing and the proliferation of solutions based on the Internet of Things . This type of system must meet a number of requirements, including the usual support for media-reach services as well as the detection of signals in both soft and hard real-time. This confluence of criteria makes it difficult to create an accurate and effective system of this kind. Model-based design methodologies offer potential solutions to these problems. Petri net patterns are a good way to illustrate the complicated behavior and set of requirements that this kind of software must handle. With simple Petri net blocks, it would be simple to depict synchronization methods, message passing, rise conditions, critical sections, concurrent and parallel processes, task activation criteria, and user interactions, to name a few. Then a simulation process might make it possible to analyze how well the solution works and tweak some of the process's variables. a Petri net model for three concurrent processes. The parallel process execution is started by the transition tl. Every process is unrestricted till its operations are completed. The transition t5 synchronizes the processes' end in this design. In other words, if one process completes its tasks, it must wait for the others to finish. The loop continues forever when each process is complete. The activity burden of each process is represented by the transitions t2, t3, and t4. Different strategies can be used to lengthen these transitions. This enables the investigation of the system's performance under various work load situations, which is necessary for the creation of real-world solutions, within the context of an appropriate simulation procedure.*

Written By Bsaam Aloush