TAXONOMY, TOOLS, AND A FRAMEWORK FOR COMBINING SIMULATION MODELS WITH AI/ML MODELS

Vijay Gehlot

Department of Computing Sciences and
Center of Excellence in Enterprise Technology
Villanova University
800 E. Lancaster Avenue
Villanova, PA 19085, USA
vijay.gehlot@villanova.edu

Peter Rokowski

Department of Electrical and
Computer Engineering
Villanova University
800 E. Lancaster Avenue
Villanova, PA 19085, USA
prokow01@villanova.edu

Elliot B. Sloane

Villanova University and
Foundation for Living,
Wellness, and Health
Sarasota, FL 34229, USA
ebsloane@gmail.com

Nilmini Wickramasinghe

Iverson Health Innovation Research Institute
Swinburne University of Technology and
Epworth HealthCare
Melbourne, AUSTRALIA
nilmini.work@gmail.com

ABSTRACT

There exist compelling use cases that demonstrate a combination of discrete event simulation and AI/ML approaches is not only possible but may yield improved results, despite their differences. This is illustrated well in the context of healthcare, particularly in an emergency department, where hard resource constraints and vague decision-making from large datasets, means neither approach sufficiently captures the entire picture of patient care. A hybrid approach may see simulation models providing details of paths that patients follow through processing, which covers associated resource utilization, and AI/ML models providing deeper insight into the treatment details of patients and their associated experiences. This paper provides a taxonomy and architecture for combining simulation models and AI/ML models. In addition, it details a concrete implementation with a Colored Petri Nets (CPN)-based simulation models coupled with Python-based AI/ML models. We illustrate the framework with several examples from the healthcare domain.

Keywords: machine learning, colored petri nets, model communication framework, synchronous coupling, asynchronous coupling.

1 INTRODUCTION

The availability of high-volume and high-quality data across a wide variety of domains has provided quite an impetus for the use of machine learning approaches and data analytics to gain useful insights into operations and management. Liberatore and Luo (2010) say “Executives are no longer content with making decisions based only on intuition and instinct; they require data and facts to support recommended courses of action.”

and define analytics as "... a process of transforming data into actions through analysis and insights in the context of organizational decision making and problem solving."

In the healthcare domain, the adoption of Electronic Health Records (EHR) has resulted in a large amount of healthcare data in electronic form that can be computationally processed. Several healthcare organizations are utilizing data mining, machine learning, and related approaches to analyze healthcare data and improve the quality of care (Umscheid et al. 2015). However, data analysis alone cannot give insights into the underlying process. For example, the efficacy of clinical interventions identified by data analysis cannot be evaluated unless the underlying cause and effects are modeled. A report by the US Institute of Medicine emphasizes that many serious errors result from systems and their interactions rather than individual failures (Reid et al. 2005). Early work on engineering medical processes and improving safety through the use of process modeling and analysis is reported in (Christov et al. 2008, Osterweil et al. 2007). Thus, to effect changes to improve healthcare and to design and deploy better systems for improving human health, we also need to adopt tools and techniques for process modeling, simulation, and analysis. Laker et al. (2018) explore the role that computer modeling and simulation can and should play in the research and development of emergency care delivery systems. The authors note that "One underutilized approach to addressing problems in health care quality and value, particularly in emergency care, is through the use of computer simulation modeling."

Simulation and machine learning, more often than not, are used in isolation from one another. On the face of it, simulation and machine learning may seem disconnected, there is an underlying common entity, namely, the data because every data that can be analyzed is produced and/or consumed by an underlying process or activity. As noted by (Greasley and Edwards 2021), "... whilst analytics is data-driven and has a focus on data and outputs and may have little knowledge of underlying processes, simulation is model-driven with a deep knowledge about processes." Working in the healthcare domain, we have come across instances where a coupled approach would be beneficial. For example, the combination of an aging and active population is creating additional demand for knee replacement (knee arthroplasty). Focusing on a cause-to-effect chain of actions is insufficient to provide statistical modeling that can forecast patient clinical outcomes, resource utilization, and complication risks. To better assess the episode of care for knee arthroplasty and continuum of care in general, it is essential to design and develop a sophisticated integrated knowledge network that can be used to model care pathways to ensure effective, efficient, and efficacious processes. A key contribution of this paper is providing a taxonomy and architecture for combining simulation models with AI/ML models. The paper also details a concrete implementation in the context of Colored Petri Nets (CPN)-based simulation models with Python-based AI/ML models. We illustrate the framework with several examples.

The remainder of this paper is organized as follows. In the next section, we provide a summary of relevant published work that deals with utilizing data mining, machine learning and related approaches with simulation approaches. Section 3 introduces a taxonomy and architecture for coupled simulation and AI/ML models. We provide a brief overview of CPN vocabulary and operations in Section 4 to help understand the details of Comms/CPN and PyCPN communication framework and associated examples that follow in Section 5. Two examples that combine CPN-based simulation with data analytics are presented in Section 6. We present our conclusions in Section 7. Finally, note that by simulation we mean discrete event simulation or equivalent frameworks and the term AI/ML includes both machine learning as well as data-mining tools and techniques.

2 RELATED WORK

One of the early works that combine data mining and simulation is reported in (Glowacka et al. 2009). The authors utilize a data mining model for predicting patient no-shows and then encode those as rules in the simulation model. They then establish that the coupling results in significant improvements over models that do not employ the rules. In terms of our taxonomy to be developed and discussed later in this paper, we would classify it as an asynchronous or loose coupling. Another work reported in (Feldkamp et al. 2017) couples in real-time simulation with an online stream mining component to get data mining results while simulation experiments are still running. We would classify this approach as a synchronous or tight coupling. Ceglowski et al. (2007) combine data mining and discrete event simulation for a value-added view of a hospital emergency department. Goodall et al. (2019) present a data-driven simulation approach to predict material flow behavior within remanufacturing operations. A work that combines discrete-event simulation and machine learning techniques in the context of the manufacturing processes and assembly lines is described in (Gyulai et al. 2014). Abohamad et al. (2017) present a hybrid approach that integrates process mining techniques in the conceptual modeling phase to support the development of simulation models. Aqlan et al. (2017) integrate data analytics and simulation modeling for defect management in manufacturing environments. A comprehensive survey of research works that combine discrete-event simulation with big data analytics is available in (Greasley and Edwards 2021).

3 A TAXONOMY AND ARCHITECTURE FOR COUPLED SIMULATION AND AI/ML MODELS

In the context of healthcare, the strength of simulation models is their ability to capture complex clinical workflows with resource constraints. However, as part of an episode of care and care continuum, such clinical workflows also involve many complex decision points where the use of an AI/ML model would be more appropriate. Similar situations arise in other application areas too as discussed in Section 2 above. Conceptually, one type of model (client) may utilize services of another type of model (server). In utilizing these services, the communication channels may be synchronous or asynchronous. In the context of simulation and AI/ML models, we get four different couplings as shown in Figure 1.

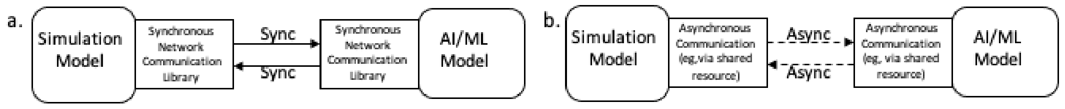

a.
Simulation Model
Synchronous Network Communication Library
Sync
Sync
Synchronous Network Communication Library
AI/ML Model
b.
Simulation Model
Asynchronous Communication (eg.via shared resource)
Async
Async
Asynchronous Communication (eg. via shared resource)
AI/ML Model

Figure 1: Synchronous (a) and asynchronous (b) couplings of Simulation and AI/ML models and their associated software architectures. The direction of arrow denotes client to server relationship. The solid arrow represents synchronous coupling whereas the dashed arrow represents asynchronous coupling.

The two diagrams (a) and (b) in Figure 1 illustrate the software architecture of these couplings. Note that each direction is independent, that is, if a simulation model as a client is invoking the services of an AI/ML model as a server, then the arrow is from the simulation model to the AI/ML model. Of course, at the communication level, the requests and responses will be bidirectional. The coupling (a) requires synchronous communication where one entity can invoke the other entity during active run-time to obtain results for a set of inputs sent over the network via blocking send/receive communication primitives. If the communication is asynchronous, then it may not require real-time communication between the entities as shown in (b). In fact, it could be implemented via a shared resource (file, database, cloud storage, etc.). Colloquially, we may refer to synchronous coupling as tight coupling and asynchronous coupling as loose coupling. Table 1 summarizes the four different ways of combining simulation models with AI/ML models.

As a concrete implementation, we have created a Python-based library (PyCPN) that can be used in conjunction with Colored Petri Nets (CPN) Tools simulation software to provide a realization of a tight (synchronous) coupling between a CPN simulation model and a Python-based AI/ML model (Figure 2). This same implementation can also be used for asynchronous coupling where access to cloud storage or database

Table 1: Four ways of combining simulation models with AI/ML models. SAM stands for Simulation And AI/ML. MAS stands for AI/ML And Simulation.

Option	**1**	**2**	**3**	**4**
Name	SAM_{sync}	$\text{SAM}_{\text{async}}$	MAS_{sync}	$\text{MAS}_{\text{async}}$
Client	Sim	Sim	AI/ML	AI/ML
Server	AI/ML	AI/ML	Sim	Sim
Coupling	Tight	Loose	Tight	Loose

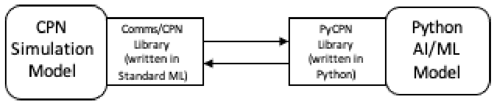

CPN Simulation Model
Comms/CPN Library (written in Standard ML)
PyCPN Library (written in Python)
Python AI/ML Model

Figure 2: Synchronous coupling and communication realization between a CPN simulation model and a Python-based AI/ML model.

server is required. Before going into the details of this implementation, we give a brief overview of CPN vocabulary and operations in the next section to help understand the details and examples that follow.

4 COLORED PETRI NETS (CPN)

At the very basic level, a Colored Petri Net consists of *places* (depicted as circles or ovals), *transitions* (depicted as rectangles or bars), and *arcs* (depicted as arrows) that connect a place to a transition or a transition to a place (Jensen and Kristensen 2009, Gehlot 2019). Places may contain *tokens* or data values, such as unit (or void), integers, strings, booleans, as well as values of a variety of compound types. The dynamics (semantics) of a CPN is defined by the *firing rule* where the *firing* of a transition removes tokens from its input place and adds tokens to its output place. The tokens removed and tokens added are governed by the expressions on the input arcs and output arcs of a transition, respectively. The firing of a transition is an abstraction of the occurrence of an event and the movement of tokens describes state changes. Figure 3 shows a net with 5 tokens in place P1 as determined by its initial marking `5`()`. When the transition T1 fires, it removes 2 tokens from place P1 and deposits 1 token in place P2, as determined by the associated outgoing and incoming arc inscriptions, respectively.

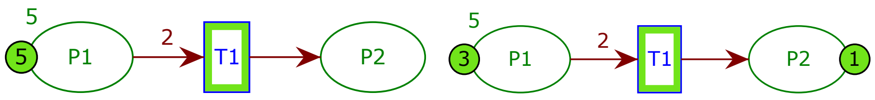


Figure 3: An example CPN before (left) and after (right) the firing of transition T1.

With the vocabulary of CPN, we can express the desired semantics of our couplings described in Section 3. Figure 4 depicts two nets. The one on the left defines the semantics of SAM_{sync} whereas the one on the right defines the semantics of $\text{SAM}_{\text{async}}$. In the net on the left, the two transitions *Sim Send and ML Receive Data* and *ML Send and Sim Receive Results* mark the synchronization point of the two processes since nether can fire until and unless one process is ready to send and the other process is ready to receive. This is in contrast with the net on the right, where neither process is blocked (assuming at least one token in *Shared Data Store* place). In fact, there is no requirement that both processes be active at the same time.

Two optional inscriptions associated with a transition in CPN are: a *guard* (list of conditions) and a *code segment*. The former can be used for finer control of transition firing. The latter is used to associate additional processing with the firing of a transition. Code segments are executed when the associated transition

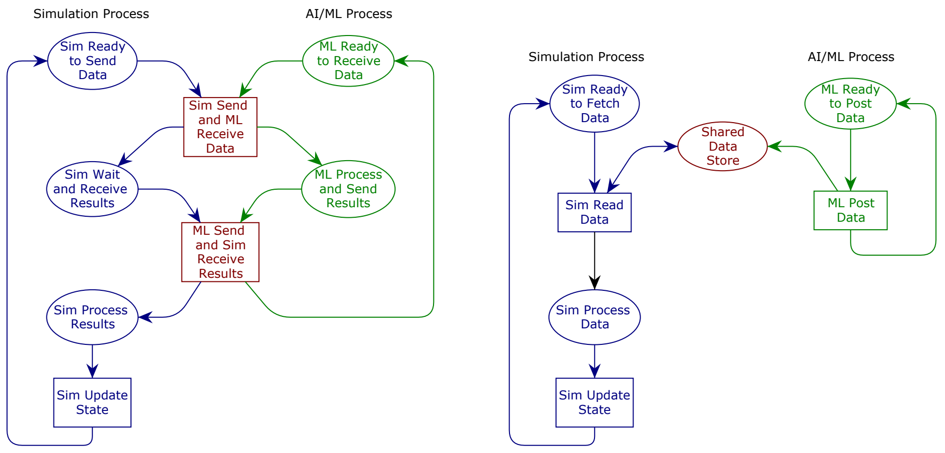
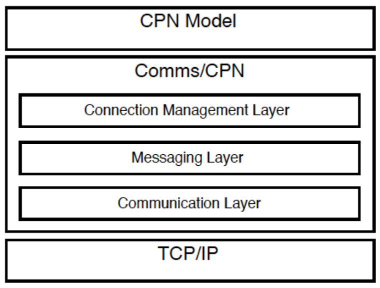


Figure 4: A CPN meta-model describing the semantics of SAM_{sync} (left) and $\text{SAM}_{\text{async}}$ (right).

fires. However, the firing is not considered complete, until after the execution of the code segment. Hence no movement of tokens or state changes occur while the code segment is executed. This allows the output of the execution to be used in determining new token values. In particular, if the code segment contains communication with an external process, then the transition waits until after the successful exchange of messages. CPN Tools, which is the software for creating, editing, simulating, and analyzing CPN models, has a built-in communication framework which we discuss next.

5 THE COMMS/CPN AND PYCPN COMMUNICATION FRAMEWORK

The communication framework was added to the CPN Tools software to create domain-specific visualization of the underlying simulation. However, the framework and the underlying protocol are general purpose and can be used in other instances too. In particular, it allows the reliable transfer of a string of arbitrary size (byte stream) (Gallasch and Kristensen 2001). Figure 5 gives the network architecture and associated connection management functions that are available as part of the Comms/CPN library.

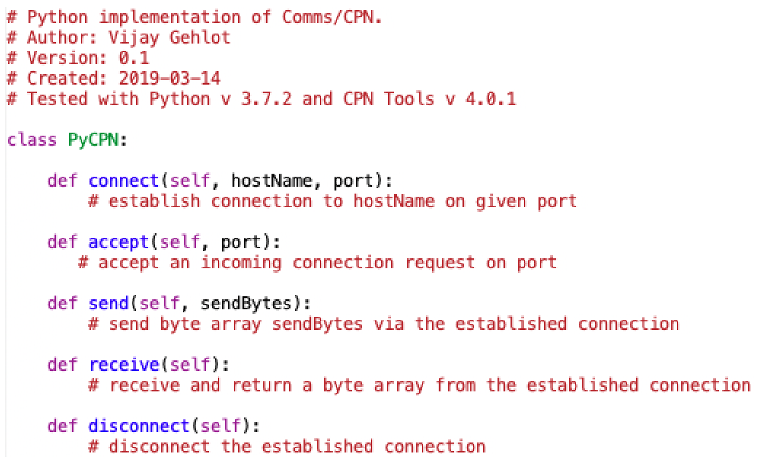


```
type Connection
exception ElementMissingExn of string
exception DupConnNameExn of string

val openConnection : Connection * string * int -> unit
val acceptConnection : Connection * int -> unit

val send : Connection * 'a * ('a -> Word8Vector.vector) -> unit
val receive : Connection * (Word8Vector.vector -> 'a) -> 'a

val closeConnection : Connection -> unit
```

Figure 5: The Comms/CPN architecture (left) and library functions (right) (Gallasch and Kristensen 2001).

Details of the underlying protocol and packetization are described in (Gallasch and Kristensen 2001). Using Comms/CPN as a basis, we have implemented a corresponding library called PyCPN in Python. We elected to go with Python because of its popularity and wide use in the AI/ML application domain. Figure 6 gives the interface of this library. An overview of the various available methods is as follows:

- **connect**: Used in a client mode to establish a connection to an external process. The parameter *hostName* specifies the IP address or domain name of the external server application as a string, and the *port* argument specifies the port number as an integer.
- **accept**: In a sever mode, listens and waits for an incoming connection on the specified *port* argument as an integer.
- **send**: Sends the sequence of bytes given by the parameter *sendBytes* using the established connection.
- **receive**: Receives and returns a sequence of bytes via the established connection.
- **disconnect**: Closes the established connection.



```
# Python implementation of Comms/CPN.
# Author: Vijay Gehlot
# Version: 0.1
# Created: 2019-03-14
# Tested with Python v 3.7.2 and CPN Tools v 4.0.1

class PyCPN:

    def connect(self, hostName, port):
        # establish connection to hostName on given port

    def accept(self, port):
       # accept an incoming connection request on port

    def send(self, sendBytes):
        # send byte array sendBytes via the established connection

    def receive(self):
        # receive and return a byte array from the established connection

    def disconnect(self):
        # disconnect the established connection
```

Figure 6: The PyCPN library interface.

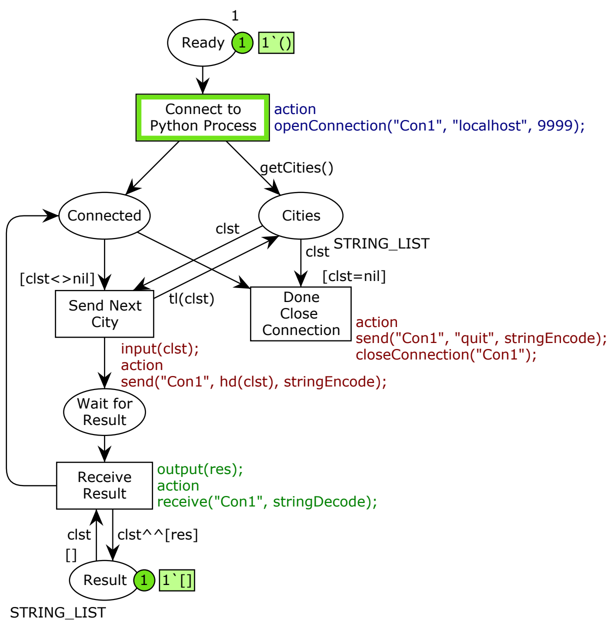
To send data using these functions, one must first serialize (convert to string) and then encode the string as a sequence of bytes. Similarly, the received bytes would need to be decoded as a string and then de-serialized by the process acting on it. The following Python functions may be used for encoding and decoding.

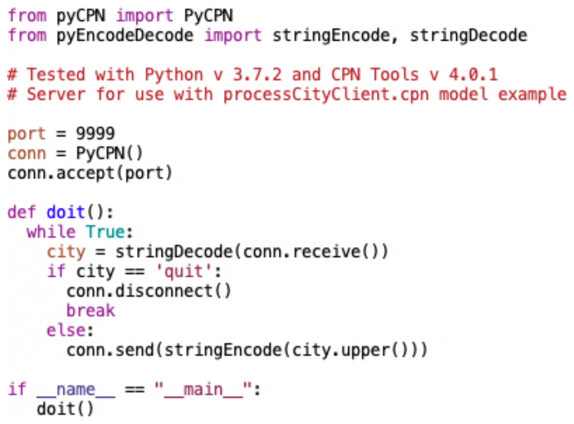
```
def stringEncode(str):          def stringDecode(bstr):
    return str.encode()             return bstr.decode()
```

Next, we provide two simple examples to show the workings of the communication frameworks. The first example, shown in Figure 7, creates a synchronous coupling SAM_{sync} between a CPN model running as a client invoking the services of a Python process running as a server.

In this figure, The code segment associated with the transition *Connect to Python Process* will establish the connection to the running Python process upon firing. At this stage, the place *Cities* will get populated by a list of city names in lowercase. The transition Send Next City and the associated code segment sends the name of the city to the Python process. The Python process converts it to all uppercase and sends it back to the CPN model, which receives it when the transition *Receive Result* fires and the associated code segment gets executed. The connection is terminated when the list is empty by sending a quit message to the Python process by the firing of transition *Done Close Connection*.

The second example, shown in Figure 8, creates a synchronous coupling MAS_{sync} between a Python process running as a client invoking the services of the CPN model running as a server. In this case the Python process repeatedly sends city names in all lowercase to the CPN model, which in return converts those to all uppercase and sends them back to the Python process.





```
from pyCPN import PyCPN
from pyEncodeDecode import stringEncode, stringDecode

# Tested with Python v 3.7.2 and CPN Tools v 4.0.1
# Server for use with processCityClient.cpn model example

port = 9999
conn = PyCPN()
conn.accept(port)

def doit():
  while True:
    city = stringDecode(conn.receive())
    if city == 'quit':
      conn.disconnect()
      break
    else:
      conn.send(stringEncode(city.upper()))

if __name__ == "__main__":
   doit()
```

Figure 7: SAM$_{\text{sync}}$ example with CPN client (left) connecting to Python server (right).

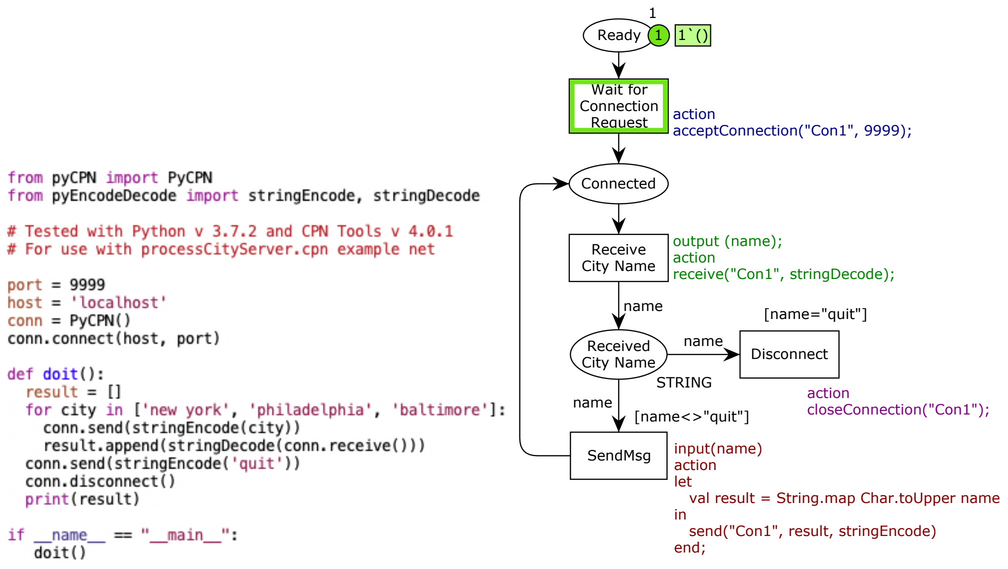
6 HEALTHCARE APPLICATION EXAMPLES

6.1 SAM$_{\text{sync}}$ Coupling Example—Musculoskeletal Virtual Care (MSKVC)

To illustrate our approach and use, we created a CPN simulation model of a Musculoskeletal Virtual Care (MSKVC) process (King et al. 2020). The overall patient flow and triaging described in the aforementioned paper is as follows:

"Patients seeking musculoskeletal care are introduced by contacting the centralized appointment desk originating most from a provider referral, emergency department (ED), or urgent care consults. Patients are diverted to a virtual short questionnaire that can be administered electronically or via telephone. The questionnaire would assess the acuity, severity or urgency and a secondary questionnaire that includes basic patient demographics, pertinent comorbidities, and key elements would provide insight into the nature of the patients' complaints, including type, location, character, acuity, recent surgical history, and newly obtained imaging. An acuity and urgency-based stratification of patients would be conducted based on outcomes of the virtual musculoskeletal triage questionnaire, where patients with chronic conditions and established follow-up patients are offered routine virtual visits according to availability and geographical location. Conversely, derangements that are deemed to be acute, especially among patients with recent surgical history, are directed to a virtual musculoskeletal triage channel where a live interview with a musculoskeletal provider would guide down-stream disposition. Based on the results of the triaging questionnaire, coupled with providers' assessment, patients would be referred to an urgency-appropriate disposition ranging from direct surgical admission, referral to the emergency department (ED) or orthopaedic acute care centre after direct surgical admission, referral to the emergency department (ED) or orthopaedic acute care centre after direct notification, and provision of a plan-of- care, or requesting appropriate imaging. Alternatively, patients could be scheduled for an on-demand (immediate), urgent (same-day), expedited (within 72 hours), or routine virtual visit with the appropriate member of an orthopaedic clinical care team."

Central to this MSKVC is a questionnaire that can be evaluated by an AI/ML model. In the simulation model of the underlying triaging process, at the decision-making state, the collected patient profile can be



```
from pyCPN import PyCPN
from pyEncodeDecode import stringEncode, stringDecode

# Tested with Python v 3.7.2 and CPN Tools v 4.0.1
# For use with processCityServer.cpn example net

port = 9999
host = 'localhost'
conn = PyCPN()
conn.connect(host, port)

def doit():
  result = []
  for city in ['new york', 'philadelphia', 'baltimore']:
    conn.send(stringEncode(city))
    result.append(stringDecode(conn.receive()))
  conn.send(stringEncode('quit'))
  conn.disconnect()
  print(result)

if __name__ == "__main__":
   doit()
```

```
action
acceptConnection("Con1", 9999);
```

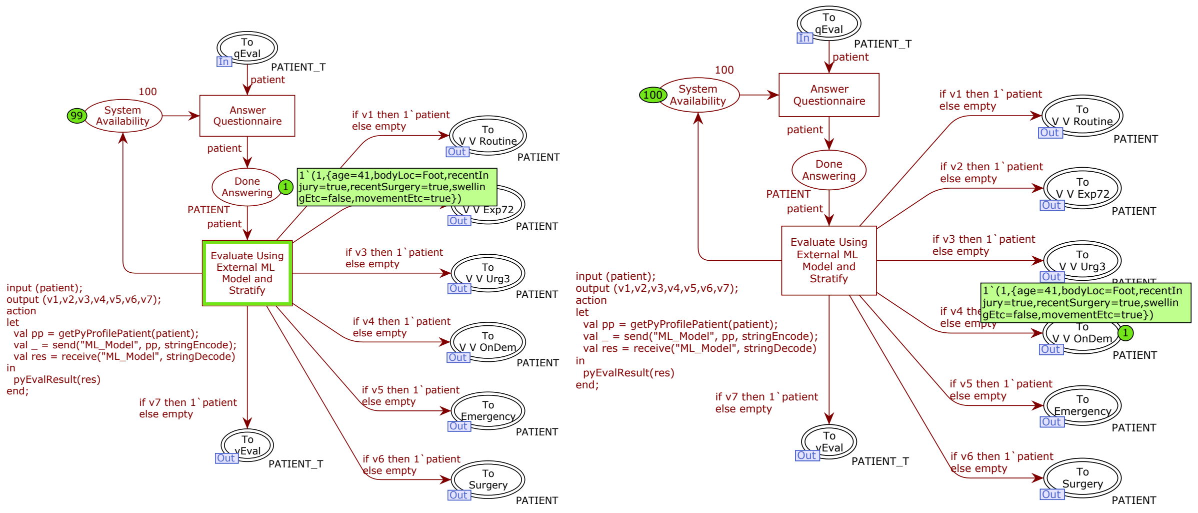
```
output (name);
action
receive("Con1", stringDecode);
```

```
action
closeConnection("Con1");
```

```
input(name)
action
let
   val result = String.map Char.toUpper name
in
   send("Con1", result, stringEncode)
end;
```

Figure 8: MAS$_{\text{sync}}$ example with Python client (left) connecting to CPN server (right).

communicated via synchronous channel to the AI/ML model and the result can then be communicated to the simulation model to proceed with the rest of the simulation steps. Of course, the AI/ML model is not capable of accounting for various resources, patient arrival rates, etc. So, we do need a simulation model to optimize the process side.



```
input (patient);
output (v1,v2,v3,v4,v5,v6,v7);
action
let
  val pp = getPyProfilePatient(patient);
  val _ = send("ML_Model", pp, stringEncode);
  val res = receive("ML_Model", stringDecode)
in
  pyEvalResult(res)
end;
```

To qEval
In
PATIENT_T
patient
100
99
System Availability
Answer Questionnaire
patient
Done Answering
1
1`(1,{age=41,bodyLoc=Foot,recentInjury=true,recentSurgery=true,swellingEtc=false,movementEtc=true})
PATIENT
patient
Evaluate Using External ML Model and Stratify
if v1 then 1`patient else empty
To V V Routine
Out
PATIENT
V V Exp72
Out
PATIENT
if v3 then 1`patient else empty
To V V Urg3
Out
PATIENT
if v4 then 1`patient else empty
To V V OnDem
Out
PATIENT
if v5 then 1`patient else empty
To Emergency
Out
PATIENT
if v6 then 1`patient else empty
To Surgery
Out
PATIENT
if v7 then 1`patient else empty
To vEval
Out
PATIENT_T
To qEval
In
PATIENT_T
patient
100
100
System Availability
Answer Questionnaire
patient
Done Answering
PATIENT
patient
Evaluate Using External ML Model and Stratify
if v1 then 1`patient else empty
To V V Routine
Out
PATIENT
if v2 then 1`patient else empty
To V V Exp72
Out
PATIENT
if v3 then 1`patient else empty
To V V Urg3
Out
PATIENT
1`(1,{age=41,bodyLoc=Foot,recentInjury=true,recentSurgery=true,swellingEtc=false,movementEtc=true})
if v4 then else empty
To V V OnDem
1
Out
PATIENT
if v5 then 1`patient else empty
To Emergency
Out
PATIENT
if v6 then 1`patient else empty
To Surgery
Out
PATIENT
if v7 then 1`patient else empty
To vEval
Out
PATIENT_T

Figure 9: Snapshot of the CPN simulation model sending patient profile to the AI/ML model (left) and routing of the patient based on the received result (right).

Utilizing the synchronous communication library and primitives described above, we open a connection to the Python process running the AI/ML model as illustrated in previous examples. Our hierarchical CPN simulation model is divided into separate sub-modules (Jensen and Kristensen 2009, Gehlot 2019). Figure 9 shows two snapshots of the sub-module responsible for handing the questionnaire. Once the patient is done answering the questionnaire, the highlighted code segment in the figure on the left gets executed when the associated transition highlighted in green is fired. As a result, the collected patient profile is converted into a string and sent to the Python process. At this point, the simulation is stopped until a result is received from the AI/ML model. The patient is then routed to the appropriate next stage by the simulation model based on the result received from the AI/ML model. This is depicted in the figure on the right. For depicted case, the patient had the following profile: age 41 with a recent injury to the foot and a recent surgery but has no swelling and can move. Thus, per the result from AI/ML model, the patient is sent for an on-demand virtual visit.

SAM$_{\text{async}}$ Coupling Example—Blood Sample Matching Lab Process

As part of our ongoing efforts focused on improving patient safety and creating a value proposition for healthcare, especially in the context of episode and continuum of care, we are creating several other models of various hospital and clinical flows and operations. We use one such model to illustrate the concept of the asynchronous coupling described above. In particular, we give details of this coupling using the model of a lab process for matching blood. In this case, we have a data analytics model that can give the frequencies of various markers of interest in blood samples based on some given profile. The simulation model then reads this data asynchronously to initialize its blood sample generator function. In this case, there is no need for the data model to be coupled synchronously with the simulation model.

In the simulation model, the representation of the blood sample consists of a list of segments. Each segment contains a list of marker positions that indicates a marker of interest at that position. The frequencies of these markers vary. For example, the notation [(1,[]),(2,[]),((3,[8]),(4,[11,12])] represents a blood sample where there are no markers of interest in segments 1 and 2 but it has a marker of interest at position 8 in segment 3 and positions 11 and 12 in segment 4. The frequency data obtained from the data analytics model is shown in Table 2.

Table 2: Marker frequency data for blood samples obtained from the data analytics model.

Segment	**Position:Frequency List**
1	1:0.0000001, 2:0.0000001, 3:0.0000001, 4:0.0005, 5:0.0000001
2	6:0.006, 7:0.0045
3	8:0.14, 9:0.135
4	10: 0.0485,11:0.4055, 12:0.468

The simulation model reads this data during the initialization step and generates a pool of blood samples to be used in the matching process. This is depicted by the model snapshots in Figure 10. It is worth noting the savings in terms of both cost and time with the simulation model if one were to optimize or re-engineer the underlying process using a wet lab with real blood samples! Table 3 summarizes results when this model is run with different samples and blood match requests.

7 CONCLUSION

We demonstrated the benefits of coupled simulation and AI/ML modeling and illustrated with a use case of musculoskeletal virtual care (MSKVC) which is integrated with knee arthroplasty. In this way, we show how it is possible to yield real-time integration of critical data and information, while simultaneously capturing tacit knowledge that can enhance optimal clinical outcomes and patient experience.

Healthcare delivery has been facing several challenges to provide high value care. Since the COVID-19 pandemic began in 2020 this challenge has become more intense. The proposed framework offers a solution to facilitate identification relatively rapidly of best treatment pathways that can yield high value outcomes for healthcare delivery and thus has significant practical application in various healthcare contexts.

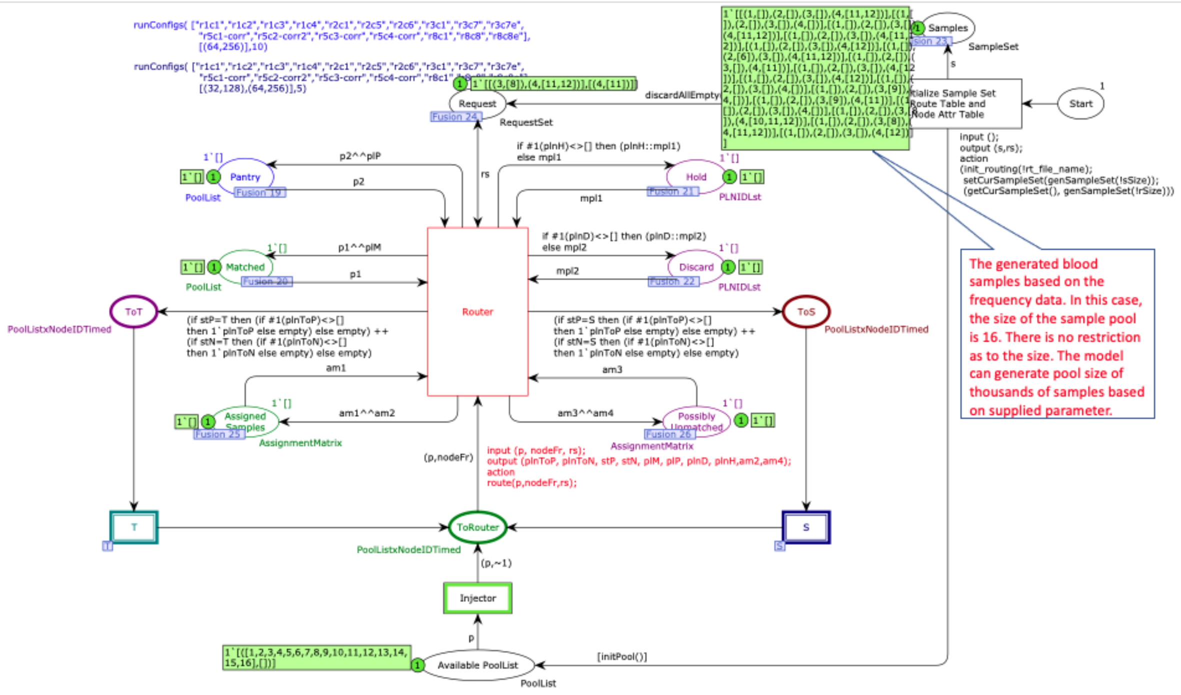

Samples
SampleSet
Initialize Sample Set Route Table and Node Attr Table
Start
Request
RequestSet
discardAllEmpty
Pantry
PoolList
Hold
PLNIDLst
Matched
Discard
Router
ToT
ToS
PoolListxNodeIDTimed
Assigned Samples
AssignmentMatrix
Possibly Unmatched
T
S
ToRouter
Injector
Available PoolList
[initPool()]
The generated blood samples based on the frequency data. In this case, the size of the sample pool is 16. There is no restriction as to the size. The model can generate pool size of thousands of samples based on supplied parameter.

Figure 10: Snapshot of the lab process simulation model showing generation of blood samples based on data supplied by a data analytics model.

We proposed a taxonomy and an approach for coupling simulation models with AI/ML and data analytics models for better clinical informatics. We presented details of Comms/CPN communication framework and a recently developed PyCPN library by us that allows CPN-based simulation models to be coupled with Python-based AI/ML models. We included two simple examples illustrating how to realize the coupling. We included details of two examples from healthcare applications—one illustrating a synchronous (tight) coupling and another illustrating an asynchronous (loose) coupling. However, it should be noted that such couplings are a part of a continuum rather than isolated instances. The conceptual diagram in Figure 11 illustrates the various facets of these couplings that can co-exist. When dealing with an episode of care and continuum of care, all possible combinations of these couplings may need to be realized.

Table 3: Results of successful matches for different samples and blood match requests.

RunNum	SampleSize	RequestSize	SuccCount	FailCount
1	128	9	9	0
2	128	10	10	0
3	256	15	14	1
4	256	15	13	2
5	128	12	11	1
6	128	10	10	0
7	256	13	11	2
8	256	14	14	0
9	128	9	9	0
10	128	9	8	1

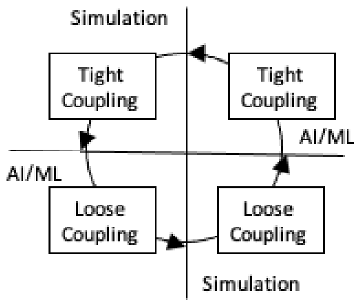


Figure 11: A conceptual diagram representing a continuum of synchronous (tight) and asynchronous (loose) coupling between simulation models and AI/ML models.

The PyCPN library is general purpose and can be used outside of the CPN context. This library can be downloaded for free, including the CPN models and associated Python code discussed in this paper. Similar communication libraries in Java and C are available here.

REFERENCES

Abohamad, W., A. Ramy, and A. Arisha. 2017, Dec. "A hybrid process-mining approach for simulation modeling". In *2017 Winter Simulation Conference (WSC)*, pp. 1527–1538.

Aqlan, F., S. Ramakrishnan, and A. Shamsan. 2017, Dec. "Integrating data analytics and simulation for defect management in manufacturing environments". In *2017 Winter Simulation Conference (WSC)*, pp. 3940–3951.

Ceglowski, R., L. Churilov, and J. Wasserthiel. 2007. "Combining Data Mining and Discrete Event Simulation for a value-added view of a hospital emergency department". *Journal of the Operational Research Society* vol. 58 (2), pp. 246–254.

Christov, S., B. Chen, G. S. Avrunin, L. A. Clarke, L. J. Osterweil, D. Brown, L. Cassells, and W. Mertens. 2008. "Rigorously Defining and Analyzing Medical Processes: An Experience Report". In *Models in Software Engineering*, edited by H. Giese, pp. 118–131. Berlin, Heidelberg, Springer Berlin Heidelberg.

Feldkamp, N., S. Bergmann, and S. Strassburger. 2017, May. "Online Analysis of Simulation Data with Stream-based Data Mining". In *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pp. 241–248.

Gallasch, G. E., and L. M. Kristensen. 2001. "Comms/CPN: A communication infrastructure for external communication with design/CPN". In *Proceedings of the Third Workshop and Tutorial on CPNs and CPN Tools*, Number DAIMI PB-554, pp. 79–93. Department of Computer Science, Aarhus University.

Gehlot, V. 2019. "From Petri Nets to Colored Petri Nets: A Tutorial Introduction to Nets Based Formalism for Modeling and Simulation". In *2019 Winter Simulation Conference (WSC)*, pp. 1519–1533.

Glowacka, K. J., R. M. Henry, and J. H. May. 2009. "A hybrid data mining/simulation approach for modelling outpatient no-shows in clinic scheduling". *Journal of the Operational Research Society* vol. 60 (8), pp. 1056–1068.

Goodall, P., R. Sharpe, and A. West. 2019. "A data-driven simulation to support remanufacturing operations". *Computers in Industry* vol. 105, pp. 48–60.

Greasley, A., and J. S. Edwards. 2021. "Enhancing discrete-event simulation with big data analytics: A review". *Journal of the Operational Research Society* vol. 72 (2), pp. 247–267.

Gyulai, D., B. Kádár, and L. Monostori. 2014. “Capacity Planning and Resource Allocation in Assembly Systems Consisting of Dedicated and Reconfigurable Lines”. *Procedia CIRP* vol. 25, pp. 185–191. 8th International Conference on Digital Enterprise Technology - DET 2014 Disruptive Innovation in Manufacturing Engineering towards the 4th Industrial Revolution.

Jensen, K., and L. M. Kristensen. 2009. *Coloured Petri Nets. Modelling and Validation of Concurrent Systems*. Berlin-Heidelberg, Springer-Verlag.

King, D., A. K. Emara, M. K. Ng, P. J. Evans, K. Estes, K. P. Spindler, T. Mroz, B. M. Patterson, V. E. Krebs, S. Pinney, N. S. Piuzzi, and J. L. Schaffer. 2020. “Transformation from a traditional model to a virtual model of care in orthopaedic surgery: COVID-19 experience and beyond”. *Bone & Joint Open* vol. 1 (6), pp. 272–280.

Laker, L. F., E. Torabi, D. J. France, C. M. Froehle, E. J. Goldlust, N. R. Hoot, P. Kasaie, M. S. Lyons, L. H. Barg-Walkow, M. J. Ward, and R. L. Wears. 2018. “Understanding Emergency Care Delivery Through Computer Simulation Modeling”. *Acad Emerg Med.* vol. 25 (2), pp. 116–127.

Liberatore, M. J., and W. Luo. 2010. “The Analytics Movement: Implications for Operations Research”. *INFORMS Journal on Applied Analytics* vol. 40 (4), pp. 313–324.

Osterweil, L. J., G. S. Avrunin, B. Chen, L. A. Clarke, R. Cobleigh, E. A. Henneman, and P. L. Henneman. 2007. “Engineering Medical Processes to Improve Their Safety”. In *Situational Method Engineering: Fundamentals and Experiences*, edited by J. Ralyté, S. Brinkkemper, and B. Henderson-Sellers, pp. 267–282. Boston, MA, Springer US.

Reid, P. P., W. D. Compton, J. H. Grossman, and G. Fanjiang. (Eds.) 2005. *Building a Better Delivery System: A New Engineering/Healthcare Partnership*, Washington D.C. National Academies Press.

Umscheid, C. A., J. Bleich, M. Draugelis, and S. Saria. 2015, October. “The Use of Predictive Analytics and Machine Learning Approaches to Improve the Quality of Inpatient Care”. In *Proceedings of the 8th Annual Mid-Atlantic Healthcare Informatics Symposium*. Philadelphia.

AUTHOR BIOGRAPHIES

VIJAY GEHLOT, Ph.D., is a Full Professor and Graduate Program Director in the Computing Sciences Department at Villanova University. His research interests include applications of Colored Petri Nets (CPNs) in the modeling and analysis of healthcare systems. He is a member of the ACM, ACM SIGSim, and Sigma Xi. His email address is vijay.gehlot@villanova.edu.

PETER ROKOWSKI is a Ph.D. student in the Department of Electrical and Computer Engineering at Villanova University. He received his BS in Computer Science and MS in Software Engineering from Villanova University. His e-mail address is prokow01@villanova.edu.

ELLIOT SLOANE, Ph.D., is a practitioner, researcher, and educator in the Clinical Engineering and MIS fields. He is the Emeritus Co-Chair of Integrating the Healthcare Enterprise International (IHE) standards organization. He has over 40 years of experience in the healthcare industry focused on patient safety, health-care technology management, and privacy and security. His email address is ebsloane@gmail.com.

NILMINI WICKRAMASINGHE, Ph.D., is an active leader in designing and developing health informatics content for many years. She has simultaneous appointments at the Swinburne University of Technology where she is the Professor of Digital Health and Deputy Director of the Iverson Health Innovation Research Institute as well as Epworth Healthcare where she is Professor and Director of Health Informatics Management. Her research interests focus on the design, development, and deployment of digital health solutions for high-value patient-centered care delivery. Her email address is nilmini.work@gmail.com.