Toward a Model for Verification of Business Logic Layer in 3-Layer Architecture: CPN-ECA Model

Thanh Tuan Nguyen, Nhan Le Thanh, Thi Thanh Ha Hoang

Toward a Model for Verification of Business Logic Layer in 3-Layer Architecture: CPN-ECA Model

Nguyen Thanh Tuan
University of Science and Education
The University of Danang, Vietnam
nttuan@ued.udn.vn

Le Thanh Nhan
WIMMICS-INRIA, France
nhan.le-thanh@unice.fr

Hoanh Thi Thanh Ha
University of Economic
The University of Danang, Vietnam
ha.htt@due.edu.vn

***Abstract*—This paper proposes a model for building a flexible system, which accepts and verifies the change on business logic, including both business processes and business rules, while the system has to cover the properties as reliability and reuse. In this model, the business process will be designed with Colour Petri Net and translated into a set of Event-Condition-Action rules, this set will be combined with business rules for checking the respect of a business process to the business rules in design and modifying the process. Hierarchical Colour Petri Net is also used to guarantee the reliability and to reuse properties of the system.**

I. Introduction

Software engineering presents several problems that can be resolved with many different techniques and methodologies. The challenge for system design is not only how to build a system, which accepts the change on business logic, consisting of business processes and business rules, but also the correctness of the system. The combination between Petri Nets and rule-based language in description and modeling business logic of software system is one solution for this challenge.

Our approach takes advantage of both the capacities in modeling and verification of Coloured Petri Nets (CPN) and Event-Condition-Action (ECA) rules. It allows designing a business process by CPN and translating the model to a set of ECA rules. The combination of business rules and business process as a set of rules, will be checked the conflict of them. Hierarchical CPN is also used to design a reusable system. The main contributions of this paper are:

- Formalization a business logic layer using CPN and Hierarchical CPN.
- Verification of business logic with ECA rules.

The rest of the paper is organized as follows: Section II identifies current works and solutions related to the CPN application and the use of ECA and ECAE languages to describe workflow for business process. In section III, a brief introduction of Coloured Petri Net and ECA rule and 3-layer architecture are given. Section IV introduces, through a case study, our solution on modeling and verification Business Logic layer. Then, we conclude on the outcomes of this experimentation and the ongoing works. V.

II. Related works

The positive aspect of CPN has been recognized as a modeling and verification tool for software engineering [1]. In [2], Denaro and Pezze presented some useful application of Petri Nets in Software engineering, as modeling and analysis of safety critical systems, distributed systems, real time systems, multimedia systems or software performance evaluation. Some papers describe CPN as aN effective tool for modeling and verification of SOA-based system [3], [4].

By the way, many authors have proposed using ECA rules as a tool for business process modeling and execution, e.g. [5]–[8]. In [8] ECAE (Event-Condition-Action-Event), an ECA-like language, were introduced as a method to easily represent the business rules found in a workflow, while the workflow, modeled by CPN, is translated to ECAE rules. The set of ECAE rules is used for verification of the respect of a business process to the business rules automatically when user upgrades the workflow. In [9], the authors propose a scheme to generate the functions for a CPN model from the business rules. It's mean, business rules will be added to CPN model as the functions and CPN tools is used to verify the model. But this solution does not cover the checking for conflict between business rule and business process on the system.

By contrast, our solution uses ECA rule for presenting and checking semantic aspect of business logic, at the same time, CPN is used for verification and validation of the constructed model.

III. Backgrounds

A. Coloured Petri Net

Coloured Petri Net (CPN) is an established concept for formal modeling of concurrent and distributed systems. It is based on a functional language called Standard ML (SML) **Milner1997** CPN has the capacities of both Petri Nets and programming language. Jensen and Kristiansen [10] have defined CPN formally as follows:

Definition 1. A Coloured Petri Net is a nine-tuple [10], $CPN = (P; T; A; \Sigma; C; N; E; G; I)$, where:

1) P is a finite set of **places**.
2) T is a finite set of **transitions** such that $P \cap T = \phi$.
3) $A \subseteq P \times T \cup T \times P$ is set of directed **arcs**.
4) Σ is a finite of non-empty **color sets**.
5) V is a finite set of typed variables such that $Type[v] \in \Sigma$ for all variables $v \in V$.

6) $C : P \rightarrow \Sigma$ is a colour set function that assigns a colour set to each place.
7) $G : T \rightarrow EXPR_V$ is a guard function that assigns a guard to each transition t such that $Type[G(t)] = Bool$.
8) $E : A \rightarrow EXPR_V$ is an arc expression function that assigns an arc expression to each arc a such that $Type[E(a)] = C(p)_{MS}$, where p is the place connected to the arc a.
9) $I : P \rightarrow EXPR_\phi$ is an initialization function that assigns an initialization expression to each place p such that $Type[I(p)] = C(p)_{MS}$.

The graph of CPN is a bipartite directed graph. It consists of vertices of two types: places drawn as circles or ovals and transitions drawn as bars. An example of CPN is described in figure 1.

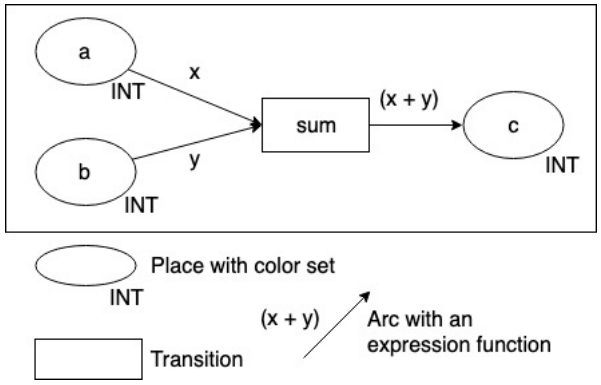

Fig. 1. An example of CPN

Though there are many ways to represent a color, colors and their associated operations are frequently expressed as SML data types and functions.

B. Event-Condition-Action Rule

Event-Condition-Action (ECA) rule is a language that has been usually used for specification and implementation of business processes [11], [12]. ECA is also used to describe rules in setting system for active databases [13] or personalizing, and publish/subscribe technology [14]–[17]. The general syntax of an ECA rule is:

On event **If** condition **Do** actions

The *event* specifies a condition for triggering the rule. The *condition* is a query, which determines if the information system is in a particular state, in which case the rule fires. The *action* states the actions to be performed if the rule is fired. These actions may in turn cause further events to occur, which may lead them to create more ECA rules to fire.

In [8], authors defined a language exhibiting both the advantages of ECA and of Logic Programming, called ECAE (Event-Condition-Action-Event) and the algorithm to translate CPN to ECA rules, which has 4 steps as follows:

Algorithm 1.

Step 1: The **C**ondition part of ECA rule is a collection of places, color sets, guard function, input arc expression related to a transition. The **E**vent part of ECA rule are transitions, the **A**ction is a group of the output arc expression and color sets.
Step 2: Translate each transition to an ECA rule.
Step 3: Add starting condition and ending condition.
Step 4: Connect all ECA rule transition as their triggered sequence.

According to the algorithm 1, the BP which is modeled by CPN, and BR are both translated to ECA rules and merged together into a set (set of ECA rule). The rule set will be verified to carry out conflicts between BP against BR by a Reasoner tool.

C. Business logic layer in 3-layer model

At the highest and most abstract level, the logical architecture view of any system can be considered as a set of cooperating components grouped into layers [18]. An application can consist of a number of basic layers. The common three-layer design consists of the following layers:

- **Presentation layer**: The presentation layer provides the user interface (UI) of the application. Typically, this is an application form or an interface, as well as the HTML document created from the web server to the client.
- **Business Logic (BL) layer**: The business logic layer applies the logic of the application using the Business Processes (BP) and the Business Rules (BR) of the request.
- **Data layer**: The data layer provides ability to access databases.

In this paper, we only focus on modeling and verification of the BL layer.

IV. Modeling and verification BL layer

A. CPN-ECA Model

In this section, we describe the CPN-ECA model, a solution for modeling and verification BL layer. It consists of two parts:

- **Business Logic Part:** In this part, BP are modeled by a CPN (Sub-part 1.1). Besides that, BR are described by ECA rules (Sub-part 1.2). After that, the BP, modeled by CPN, can be translated to a set of ECA rules with algorithm 1, in II. In order to ensure the correctness of BP, users need to provide rules (called business rule - BR) to control the correctness. The BR is also written by ECA. the BR rules and the BP rules are in integrated into a set (Sub-part 1.3).
- **Verification Part:** Here, BP, modeled by CPN, will be verified as: deadlock, infinity cycle or missing synchronization (Sub-part 2.1). In addition, BR and BP were described by ECA rules, they can be verified by a Reasoner tool [19] to detect the conflict (Sub-part 2.2).

The CPN-ECA model is presented in Figure 2. CPN Tools [20], [21]) is used for editing CPN models.

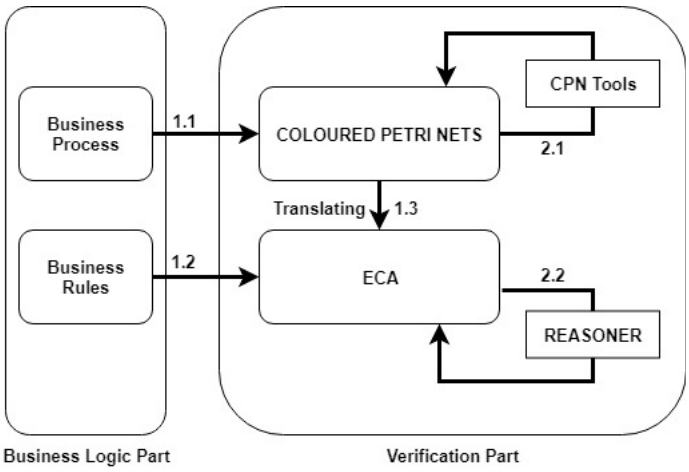

Fig. 2. CPN-ECA model

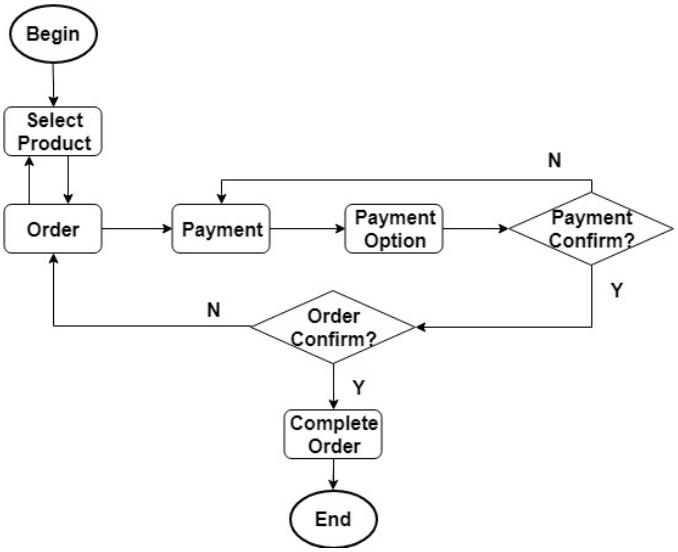

Fig. 3. Flowchart of Order Business in the Online Shopping System

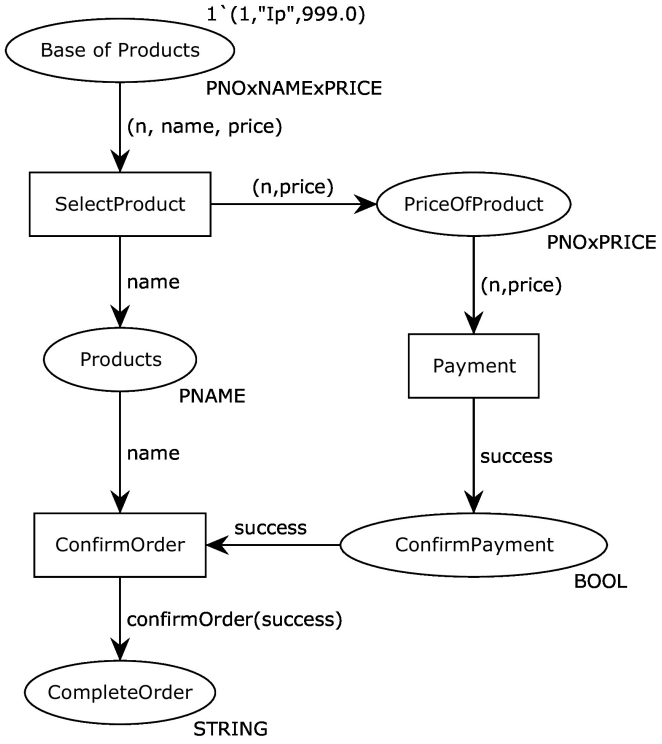

Fig. 4. CPN model of Order Business in an Online Shopping System

```
colset STRING = string;
colset PNO = INT;
colset PNAME = STRING;
colset PPRICE = REAL;
colset PNOxPRICE = product PNO * PPRICE;
colset PNOxNAMExPRICE = product PNO * PNAME * PPRICE;
var n: PNO;
var name: PNAME;
var price: PPRICE;
var success: BOOL;
fun confirmOrder(success);
```

Fig. 5. Declarations of CPN model of Order Business

B. Experimentation of the model

The Online Shopping System is a typical e-commerce system, designed to help customers can find and buy items faster and more convenient. The Order Business process is the core business of this system. To demonstrate the CPN-ECA model, we take the Order Business process as an example of this paper, the flowchart of Order Business is shown in figure 3. The goal of this process is to take orders from customers and confirm orders after payment is done. This module is modeled by CPN and presented in example 1.

Example 1. In figure 4, we design a CPN graph to represent the Order Business process of an Online Shopping System. This provides a simple example, there are three main transitions in this CPN graph. The first transition, **SelectProduct**, allows users to choose the product they want to buy, while the second transition, **Payment**, allows the user to pay for his order, and the last transition, **ConfirmOrder**, allows the user to confirm his order. Declarations of model are presented in figure 5.

The CPN graph in figure 4 can be translated to a set of ECA rules:

BP1-R1: **If** BaseOfProducts &PNOxNAMExPRICE(n,name,price) **Do** SelectProduct&PNAME(name) &PNOxPRICE(n,price)

BP1-R2: **On** SelectProduct&PNAME(name) &PNOxPRICE(n,price) **If** Done **Do** Products&PNAME(name) &PriceOfProducts&PNOxPRICE(n,price)

BP1-R3: **If** Products&PNAME(name)&ConfirmPayment &BOOL(success) **Do** ConfirmOrder

```
colset PTYPE = STRING;
colset PPRICExPTYPE = product PPRICE * PTYPE;
var sum: PPRICE;
var paytype: PTYPE;
var info: STRING;
fun payCOD(sum,paytype)
fun payCard(sum,paytype)
```

Fig. 6. Declarations of Payment Business sub-model

&confirmOrder(success)

BP1-R4: **If** PriceOfProducts&PNOxPRICE(n,price) **Do** Payment&BOOL(success)

BP1-R5: **On** Payment&PNOxPRICE(n,price) **If** Done **Do** ConfirmPayment&BOOL(success)

BP1-R6: **On** ConfirmOrder&confirmOrder(success) **If** Done **Do** CompleteOrder

When a business process is executed, it must respect a set of business rules. We add a set of business rules, for example 1:

BR1-R1: If $(n \leq 0)$ do not SelectProduct &PNO(n)&PNAME(name)&PPRICE(price)

BR1-R2: If $(name = "")$ do not ConfirmOrder&confirmOrder(success)

We merge two sets of ECA rules into a single knowledge base. Therefore, we can easily check the compliance of BP with a set of BR by detecting the conflict between the rules in one knowledge base using a reasoner.

C. Using Hierarchical CPN for reusing of model

Hierarchy is a powerful concept in CPN, with that, a CPN can be organized as a set of sub-model, in a way similar to that in which programs are organized into modules. In this way, a model can be defined one and used repeatedly.

In example 2 we extend example 1 to build the new Order Business model by adding Payment Business sub-model.

Example 2. Using Hierarchical CPN, Order Business model is added an sub-model Payment Business. Declarations of the sub-model are presented in figure 6.

The communication between the model and the sub-model takes place at input port, output port or input/output port that must to be defined and labelled at some places in sub-model. In Payment Business sub-model (see figure 8), the input port (labelled **In**) and output port (labelled **Out**) are defined at **PriceOfProduct** place and **ConfirmPayment**. In this example, two transitions **valideCODInfo** and **valideCardInfo** in the sub-model get control from the transition **SelectProduct** (in the main model) through the input port **PriceOfProduct** and send the control to transition **ConfirmOrder** (in the main model) through the output port at **ConfirmPayment**.

About BP modeled by a Hierarchy CPN, each sub-model will have a set of BP rules by ECA. Each set can be verified in-dependency with others.

The CPN graph of Payment Business sub-model in example 2 will be translated to a set of ECA rules:

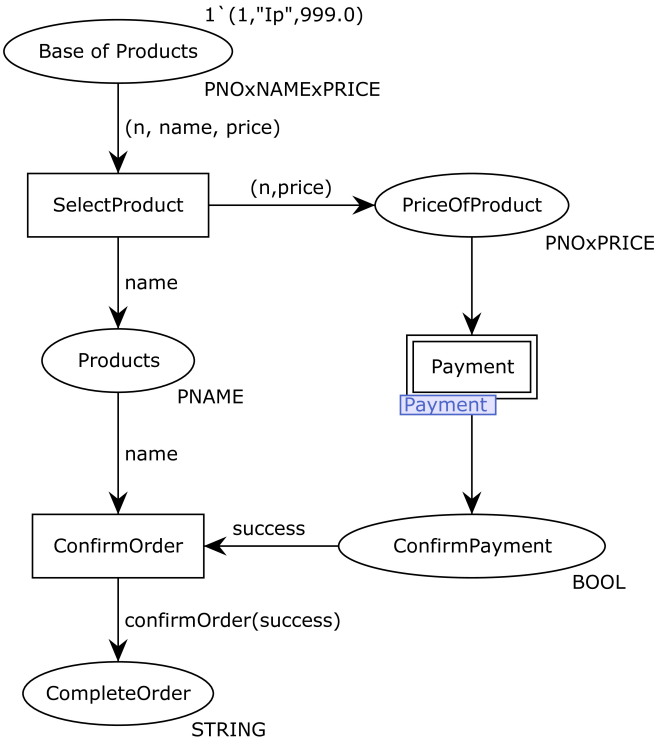

Fig. 7. Hierarchical CPN model of Order Business

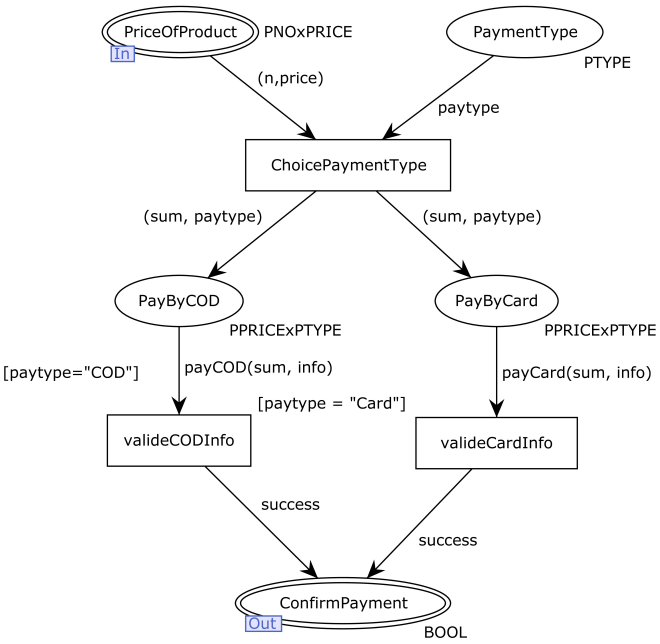

Fig. 8. Sub-model of Payment Business on Order Business.

BP21-R1: **If** PriceOfProduct&PNOxPRICE(n,price) &PaymenType&PTYPE(paytype) **Do** ChoicePaymentType &PPRICExPTYPE(sum,paytype)

BP21-R2: **On** ChoicePaymentType &PPRICExPTYPE(sum,price) **If** Done **Do** PayByCOD&payCOD(sum,info) &payByCard&payCard(sum,info)

BP21-R3: **If** PayByCOD &payCOD(sum,info)&(paytype = "COD") **Do** valideCODInfo&BOOL(success)

BP21-R4: **If** PayByCard &payCard(sum,info)&(paytype = "Card") **Do** valideCardInfo&BOOL(success)

BP21-R5: **On** valideCODInfo &BOOL(success) **If** Done **Do** ConfirmPayment

BP21-R6: **On** valideCardInfo &BOOL(success) **If** Done **Do** ConfirmPayment

Integration between this set of rules with the set of rules of main model, in figure 7, will create a set of ECA rules which will describe our Business Logic.

BP2-R1: **If** BaseOfProducts &PNOxNAMExPRICE(n,name,price) **Do** SelectProduct &PNAME(name)&PNOxPRICE(n,price)

BP2-R2: **On** SelectProduct&PNAME(name) &PNOxPRICE(n,price) **If** Done **Do** Products&PNAME(name) &PriceOfProducts&PNOxPRICE(n,price)

BP2-R3: **If** Products&PNAME(name)&ConfirmPayment &BOOL(success) **Do** ConfirmOrder&confirmOrder(success)

BP2-R4: **On** ConfirmOrder&confirmOrder(success) **If** Done **Do** CompleteOrder

BP2-R5: **If** PriceOfProduct &PNOxPRICE(n,price)&PaymenType &PTYPE(paytype) **Do** ChoicePaymentType &PPRICExPTYPE(sum,paytype)

BR2-R1: If ($n \leq 0$) do not SelectProduct&PNO(n) &PNAME(name)&PPRICE(price)

BR2-R2: If ($name$"") do not ConfirmOrder &confirmOrder(success)

BP21-R1: **If** PriceOfProduct&PNOxPRICE(n,price) &PaymenType&PTYPE(paytype) **Do** ChoicePaymentType &PPRICExPTYPE(sum,paytype)

BP21-R2: **On** ChoicePaymentType &PPRICExPTYPE(sum,price) **If** Done **Do** PayByCOD&payCOD(sum,info) &payByCard&payCard(sum,info)

BP21-R3: **If** PayByCOD &payCOD(sum,info)&(paytype = "COD") **Do** valideCODInfo&BOOL(success)

BP21-R4: **If** PayByCard &payCard(sum,info)&(paytype = "Card") **Do** valideCardInfo&BOOL(success)

BP21-R5: **On** valideCODInfo &BOOL(success) **If** Done **Do** ConfirmPayment

BP21-R6: **On** valideCardInfo &BOOL(success) **If** Done **Do** ConfirmPayment

V. Conclusions

CPN and ECA rule play an important role in designing and modeling a software system. Coloured Petri Net, inherited from the traditional Petri Net, have a better ability on expressiveness because of their color sets and guard functions. In this paper, we proposed the CPN-ECA model that exhibits the advantages of both ECA and CPN, for the problem of modeling and verification the business logic in the Business Logic layer of a software system. Our CPN-ECA model contains two parts: (1) The BL part lets user define the BP by CPN, the BP are converted to ECA rules. User defines also the BR by ECA rules. (2) The verification part takes care of checking the consistency of the rules written by ECA that are produced in the BL part. We experimented the model on Shoping Online case study.

In future work, we will focus on enhancing the integration between CPN and ECA rule, which will make our method more suitable for designing and developing software. We will also consider the problem, how to implement and verify automatically the software based on CPN and ECA rule.

References

[1] R. Gold. "Petri nets in software engineering". In: *Petri Nets: Applications and Relationships to Other Models of Concurrency* (2004), pp. 62–96. URL: http://www.springerlink.com/index/P50746R643246142.pdf.

[2] Giovanni Denaro and Mauro Pezze. *Petri Nets and Software Engineering*. 2004. DOI: 10.1007/978-3-540-27755-2_12.

[3] Gero Decker, Alexander Lüders, Hagen Overdick, et al. *RESTful petri net execution*. 2009. DOI: 10.1007/978-3-642-01364-5_10.

[4] Pawan Kumar and Ratneshwer Gupta. "Dependency Modeling of a SOA Based System Through Colored Petri Nets". In: *CIT* 24 (2016), pp. 253–269.

[5] Joonsoo Bae, Hyerim Bae, Suk Ho Kang, et al. "Automatic control of workflow processes using ECA rules". In: *IEEE Transactions on Knowledge and Data Engineering* 16.8 (2004), pp. 1010–1023. ISSN: 10414347. DOI: 10.1109/TKDE.2004.20.

[6] J. J. Alferes, F Banti, and A Brogi. “An Event-Condition-Action Logic Programming Language”. In: *Logics in Artificial Intelligence*. 2006, pp. 29–42. DOI: 10.1007/11853886_5. URL: http://www.springerlink.com/index/b27213007lg32723.pdf%7B%5C%%7D5Cnhttp://link.springer.com/10.1007/11853886%7B%5C_%7D5.

[7] Zhou Guo-xiang Zhou Guo-xiang and Gao De-ping Gao De-ping. “ECA Rule and Colored Petri Nets Based Workflow Modeling Research”. In: *Management and Service Science (MASS), 2010 International Conference on* 60633060 (2010), pp. 1–4. DOI: 10.1109/ICMSS.2010.5575732.

[8] Tuan Anh Pham and Nhan Le Thanh. “A Rule-Based Language for Integrating Business Processes and Business Rules”. In: 14178.1 (2015). URL: https://hal.inria.fr/hal-01184256.

[9] Jatuporn Deesukying and Wiwat Vatanawood. “Generating of business rules for Coloured Petri Nets”. In: *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)* (2016). DOI: 10.1109/ICIS.2016.7550824.

[10] Kurt Jensen and Lars M Kristensen. *Coloured Petri Nets Modelling and Validation of Concurrent Systems*. Vol. 9. 3-4. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. DOI: 10.1007/b95112.

[11] Serge Abiteboul, Victor Vianu, Brad Fordham, et al. “Relational transducers for electronic commerce”. In: (2003), pp. 179–187. DOI: 10.1145/275487.275507.

[12] Stefano Ceri. *Designing database applications with objects and rules: the IDEA Methodology*. 1997.

[13] Norman W. Paton and Oscar Díaz. *Active database systems*. Vol. 31. 1. Morgan-Kaufmann, San Mateo, California., 1999, pp. 63–103. DOI: 10.1145/311531.311623.

[14] Asaf Adi, David Botzer, Opher Etzion, et al. “Push technology personalization through event correlation”. In: *Vldb* (2000), pp. 643–645. URL: ftp://ftp10.us.freebsd.org/users/azhang/disc/disc01/cd1/out/papers/vldb/pushtechnologypasdao.pdf%7B%5C%%7D5Cnpapers2://publication/uuid/AAA5F251-CB50-4F68-875C-3AA590735810.

[15] Angela Bonifati, Stefano Ceri, and Stefano Paraboschi. “Active rules for {XML}: {A} new paradigm for {E}-services”. In: *The {VLDB} {J}ournal* 10.1 (2001), pp. 39–47. ISSN: 1066-8888.

[16] Angela Bonifati, Stefano Ceri, and Stefano Paraboschi. “Pushing reactive services to XML repositories using active rules”. In: *Computer Networks* 39.5 (2002), pp. 645–660. ISSN: 13891286. DOI: 10.1016/S1389-1286(02)00226-8.

[17] João Pereira, Françoise Fabret, François Llirbat, et al. “Efficient Matching for Web-Based Publish/Subscribe Systems”. In: (2006), pp. 162–173. DOI: 10.1007/10722620_17.

[18] Microsoft Patterns Practices Team. *Microsoftfffdfffd Application Architecture Guide, 2nd Edition (Patterns Practices)*. 2nd ed. Microsoft Press, 2008. ISBN: 073562710X,9780735627109.

[19] Erik Behrends, Oliver Fritzen, Wolfgang May, et al. “An ECA Engine for Deploying Heterogeneous Component Languages in the Semantic Web”. In: (2006), pp. 887–898. DOI: 10.1007/11896548_67.

[20] CPNTools. *CPN Tools A tool for editing, simulating, and analyzing Colored Petri nets*. URL: http://cpntools.org/.

[21] Anne Vinter Ratzer, Lisa Wells, Henry Michael Lassen, et al. “CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets”. In: 29.4 (2003), pp. 450–462. DOI: 10.1007/3-540-44919-1_28. URL: http://cpntools.org/.