საქართველოს ტექნიკური უნივერსიტეტი

გია სურგულაძე, დავით გულუა

ობიექტ-ორიენტირებული მოდელირება უნიფიცირებული პეტრის ქსელებით PNML

(Petri Net Marcup Language)

დამტკიცებულია:
სტუ-ს სარედაქციო-
საგამომცემლო
საბჭოს მიერ

თბილისი – 2007

უაკ 681.3.06

გადმოცემულია პეტრის ქსელების კლასიფიკაცია და მათი კვლევის ასპექტები უნიფიცირებისა და ინტერნეტში მათი შესაბამისი აღწერის პროგრამების გადაცემის თვალსაზრისით. ამოცანა დაკავშირებულია პეტრის ქსელებისთვის ერთიანი ინფრასტრუქტურის შექმნის პრობლემებთან, რაც მათი სტანდარტიზაციის ბაზის შექმნის მნიშვნელოვანი ეტაპია.

მონოგრაფიაში შემოთავაზებულია დიდი პეტრის ქსელების ასახვისთვის გვერდებად და მოდულებად ორგანიზების კონცეფცია, განსაზღვრულია მათი ძირითადი თვისებები და ასახვის მეთოდები. ჩამოყალიბებულია **PNML**-ენის გამოყენების ინტერნეტ-ტექნოლოგია **XML**-ის სინტაქსის საფუძველზე. ასეთი მიდგომით შესაძლებელი იქნება პეტრის ქსელის სხვადასხვა სიმულატორებს შორის მონაცემთა გაცვლის თავსებადობის უზრუნველყოფა.

განკუთვნილია თეორიული და პრაქტიკული ინფორმატიკისა და სხვა დარგების სპეციალისტებისათვის, დოქტორანდების, მაგისტრანტებისა და სტუდენტებისათვის, რომლებიც დინამიკური ტექნოლოგიური პროცესების მოდელირებასა და კვლევას აწარმოებენ.

რეცენზენტები:

- საქ.მეცნ.აკადემიის წევრ-კორესპოდენტი, ტ.მ.დ., პროფესორი **ა. ფრანგიშვილი**
- ტ.მ.დ., პროფესორი **ნ. ჯიბლაძე**

პროფ. გ. სურგულაძის რედაქციით

ISBN 99940-48-07-4

Georgian Technical University

GIA SURGULADZE, DAVID GULUA

OBJECT-ORIENTED SIMULATION WITH UNIFIED PETRI NETS

Supported by DAAD (Germany)

The classification of Petri Nets is made in order to elaborate common interchange format of Petri Nets, so called Petri Net Markup Language (PNML), which is based on Extensible Markup Language (XML) and makes possible flexible data transfer between different Petri Net types. Common interchange format is used in design process of different Information Management systems.

Tbilisi 2005

წინასიტყვაობა

წარმოდგენილი ნაშრომი ეძღვნება პეტრის ქსელების გამოყენების საკითხების განვითარებას განაწილებული სისტემების დაპროექტების ამოცანების გადასაწყვეტად, კერძოდ ობიექტ-ორიენტირებული მოდელირებისა და პროგრამირებისათვის.

მონოგრაფიაში შემოთავაზებულია პეტრის ქსელების გაფართოებული ვარიანტი ობიექტ-ორიენტირებული მიდგომით, მისი ინსტრუმენტის სტანდარტულ-უნიფიცირებულ ფორმამდე მისაყვანად. ასეთი მიდგომა ახალია და ბოლო წლებში აქტიურად ვითარდება ამერიკისა და გერმანიის წამყვან უნივერსიტეტებში.

პეტრის ქსელის ფორმატირების ენა (Petri Net Markup Language) არის სისტემური პეტრის ქსელისა და ობიექტ-ორიენტირებული მეთოდოლოგიის (უნიფიცირებული მოდელირების ენის-UML) შერწყმით მიღებული ინსტრუმენტი, რომელიც უზრუნველყოფს დიდი პეტრის ქსელების დეკომპოზიციისა და ინტერნეტით მათი გადაცემის შესაძლებლობას, აგრეთვე პეტრის ქსელების სხვადასხვა გრაფო-ანალიზური სიმულატორების თავსებადობას.

ავტორებს ნაშრომში შემოთავაზებული აქვთ პეტრის ქსელის ფორმატირების ენისათვის XML და Java ენების გამოყენება. წიგნში მრავლადაა საილუსტრაციო მაგალითები, რაც ხელს უწყობს მასალის აღქმას.

ნაშრომი უდავოდ საინტერესო და სასარგებლო იქნება როგორც ინფორმატიკის, ასევე ტექნიკისა და ტექნოლოგიების სხვა დარგის მეცნიერებისა და სპეციალისტებისთვის, რომელთაც მიზნად ქსელური, დინამიკური პროცესების მოდელირება და კვლევა განუზრახავთ.

არჩილ ფრანგიშვილი
საქ.მეცნ.აკადემიის წევრ-კორესპონდენტი,
ტექნიკის მეცნიერებათა დოქტორი,
პროფესორი

ს ა რ ჩ ე ვ ი

თავი II
პეტრის ქსელების ფორმატირების ენა (PNML)

2.1 პეტრის ქსელის ინსტრუმენტების (სიმულატორების) ანალიზი

წიგნში დასმული ამოცანის აქტუალურობას დავასაბუთებთ იმით, რომ ავსახავთ სადღეისო ვითარებას პეტრის ქსელების სიმულატორთა სამყაროში, რათა პეტრის ქსელის სხვადასხვა ტიპებისა და სიმულატორებისთვის **ერთიანი გაცვლითი ფორმატის** შექმნის საჭიროება გამოიკვეთოს.

პეტრის ქსელების ინსტრუმენტების, ეგრეთ წოდებული **სიმულატორების** სრული და მუდმივგანახლებადი მონაცემთა ბაზა პეტრის ქსელების ოფიციალურ ვებ-გვერდზე მოიპოვება [1]. 2005 წლის 1 ოქტომბრის მონაცემებით, ბაზაში 50 რეგისტრირებული სიმულატორი შედის, აქედან 11 კომერციული პროდუქტია (სასწავლო დაწესებულებების-თვის ფასდაკლებით), ხოლო 39 თავისუფლად ვრცელდება ინტერნეტის ქსელში.

პლატფორმებიდან (კომპიუტერის არქიტექტურა პლუს ოპერაციული სისტემა) **PC**-არქიტექტურისთვის (**Windows** ოპერაციული სისტემით) ყველაზე მეტი სიმულატორია შექმნილი, თუმცა პეტრის ქსელის სიმულატორთა გავრცელების არე ძალიან ფართოა.

არსებობს სიმულატორები თვით **SiliconGraphics** (ოპერაციული სისტემა **IRIX**) ფირმის სუპერ-კომპიუტერებისა და **HewletPackard**-ის (ოპერაციული სისტემა **HP-UX**) პლატფორმებისთვის, აგრეთვე **OS2**-სა და **BSD**-ოჯახის ოპერაციული სისტემებისთვის **(FreeBSD, NetBSD, OpenBSD).** პეტრის ქსელის რეგისტრირებულ სიმულატორთა გავრცელების სადღეისო ვითარება 2.1 ცხრილშია მოცემული.

პეტრის ქსელების სიმულატორები და პროგრამული პლატფორმები ცხრ.2.1

არქიტექტურა	ოპერაციული სისტემა	სიმულატორთა რაოდენობა
PC	Ms Dos	4
	Windows 95/98/NT/2000/XP	31
	LINUX	22
SUN	SUN OS, SOLARIS	21
Macintosh	MAC OS, MAC OS X	5
JAVA		18

ამ ცხრილში არ იანგარიშება სიმულატორთა ის ნაწილი, რომლებიც მორალურად მოძველდა და მონაცემთა ბაზიდან წაშლილ იქნა (როგორც, მაგალითად, ადრე ცნობილი გერმანული სიმულატორი **Pepsi**). კლასიკურ პეტრის ქსელებთან ერთად სულ უფრო მეტი ახალი სიმულატორი იძენს მაღალი დონის პეტრის ქსელების აგებისა და ანალიზის საშუალებებს. სიმულატორთა რაოდენობრივი განაწილება პეტრის ქსელის ტიპების მიხედვით 2.2 ცხრილში აისახება.

პეტრის ქსელების სიმულატორთა განაწილება ტიპების მიხედვით. ცხრ.2.2

პეტრის ქსელის ტიპი	P/T	დროითი	სტოქას-ტური	ობიექ-ტური	მაღალი დონის
სიმულა-ტორთა რაოდ.	36	33	16	5	31

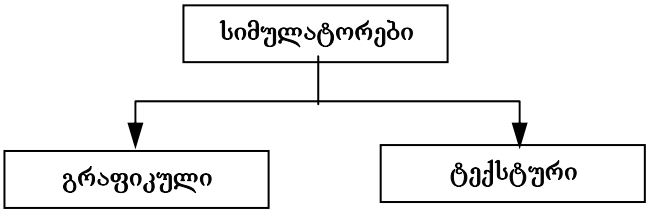
ზემოთ მოყვანილი ცხრილებიდან ნათელი ხდება, რომ სხვადასხვა სიმულატორები ხშირად ერთსა და იმავე სამუშაოს განმეორებით ასრულებს ერთმანეთთან შეუთავსებლობის გამო, რაც ასევე ახალი, ერთიანი გაცვლითი ფორმატის შექმნის აუცილებლობაზე მიუთითებს.

ჩამოვთვალოთ მთავარი თვისებები, რომელიც პეტრის ქსელის სიმულატორებს გააჩნიათ ან უნდა გააჩნდეთ:

ყოველი სიმულატორის უპირველესი კომპონენტი პეტრის ქსელის **აგების** და **გამოსახვის** საშუალებაა. ქსელი შეიძლება ფაილიდან ჩაიტვირთოს ან მომხმარებელთან ინტერაქტიულ დიალოგში შეიქმნას. მრავალი სიმულატორი დამატებით მარკერთა სიმულაციის ვიზუალური გრაფიკული საშუალებებითაცაა აღჭურვილი (**HPSim, CPN-Tools, VisualObjectNet++** და სხვ.). ანიმაციური ელემენტების ჩართვა სიმულატორს სიცხადეს ჰმატებს.

ანალიზის საშუალებები პეტრის ქსელის სიმულატორებისთვის ასევე მნიშვნელოვანი კომპონენტია. იგულისხმება ინვარიანტულობის მატრიცების და მიღწევადობათა „ხის“ აგება და დამუშავება, აგრეთვე მათი საშუალებით პეტრის ქსელების სხვადასხვა ამოცანათა (**მიღწევადობა, უსაფრთხოება, შეზღუდულობა, აქტიურობა, სხვადასხვა პეტრის ქსელთა ორეულობა, ინვერსულობა** და ასე შემდეგ) გადაწყვეტა.

მხოლოდ გარეგნული თვალსაზრისით თუ გავარჩევთ, სიმულატორების 2 კლასი გამოიყოფა (ნახ.2.1).



ნახ.2.1.

გრაფიკული სიმულატორებია **CPN-Tools, Renew, PEP** და სხვ., ხოლო ტექსტური - **INA, MARIA, LoLA**.

პირველი უფრო პეტრის ქსელების დიზაინერებსა და მომხმარებლებზეა გათვლილი, მეორე – ექსპერტებზე. ეს უკანასკნელი კლასი გრაფიკული ინტერფეისის უარყოფის სანაცვლოდ პეტრის ქსელების ანალიზის კარგად განვითარებულ საშუალებებს შეიცავს.

უკვე არის ორი კლასის სიმულატორების ინტეგრირების ცალკეული მცდელობებიც.

პეტრის ქსელების ერთიან მონაცემთა ფორმატზე მუშაობა მხოლოდ რამდენიმე წლის წინ დაიწყო. უკვე შექმნილ სიმულატორებში მონაცემთა ფორმატი უმრავლეს შემთხვევაში დამოუკიდებლად განისაზღვრება. დამპროგრამებელი თვითონ განსაზღვრავს პეტრის ქსელის ელემენტების შენახვისა და ასახვის მექანიზმს. შესაბამისად, სხვადასხვა ინსტრუმენტებს შორის მონაცემთა გაცვლა დამატებითი დაპროგრამების გარეშე ვერ ხერხდება. იყო მხოლოდ რამდენიმე მცდელობა ერთიანი ფორმატის შემოღებისა, ოღონდ არა პეტრის ქსელის ყველა, არამედ ზოგიერთი კონკრეტულ ტიპებს შორის (მაგალითად **PEP** და **Design/CPN**), რომლებიც საერთო ამოცანებმა დაუკავშირა ერთმანეთს. საერთო ჯამში, პეტრის ქსელის სიმულატორთა ურთიერთ-დამოუკიდებლობის და არათავსებადობის დონე მაინც უაღრესად მაღალია.

2.2 PNML-ის მოდელი

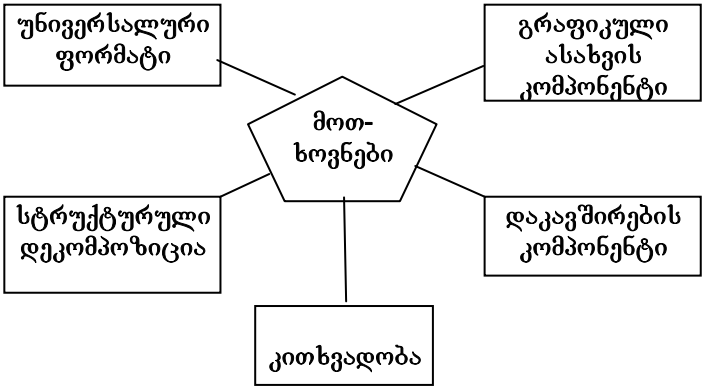
2.2.1 ძირითადი მოთხოვნები

წინა თავში პეტრის ქსელების ერთიანი კონცეფციის ერთერთი ვარიანტი შემოვიტანეთ. მასში მთავარი მახვილი **პეტრის ქსელის ტიპის** განსაზღვრაზეა დასმული, რომელიც **ქსელის**, **ჭდეებისა** და **გახსნის წესების** ერთობლიობას წარმოადგენს. ამათგან ქსელის საბაზო სტრუქტურა (**პოზიციები, გადასასვლელები, რკალები**) ფუნდამენტური და მსგავსია ყველა ტიპის

პეტრის ქსელისთვის, ხოლო **ჭდეების სიმრავლე** და **გადასასვლელთა გახსნის წესები** – განსხვავებული.

წინამდებარე თავში პეტრის ქსელების სინტაკსური ასახვის პრობლემებს შევეხებით.

საჭიროა ჩამოყალიბდეს უნივერსალური აღწერის ენა, რომელიც პეტრის ქსელის ნებისმიერი ტიპის აღწერისთვის იქნება გამოსადეგი (ნახ.2.2.).



ნახ.2.2.

- პირველი მოთხოვნა, რომელიც ამგვარ ენას წაეყენება, არის **მონაცემთა უნივერსალური ფორმატი**, რომელიც ამავდროულად **მოქნილი**, **თავსებადი** და **გაფართოებადიც** იქნება პეტრის ქსელის ახალი ტიპების უმტკივნეულოდ ინტეგრირებისთვის.

მოქნილობა გულისხმობს, რომ ენამ უნდა ასახოს პეტრის ქსელის ყოველი ტიპი თავის უნიკალური თვისებებით. მან არ უნდა შეზღუდოს ან დამალოს პეტრის ქსელის ზოგი თვისება მისი კონვერტირების პროცესში.

თავსებადობა ნიშნავს მაქსიმალური ოდენობის ინფორმაციის გაცვლის შესაძლებლობას პეტრის ქსელების სხვადასხვა ტიპებს შორის, ხოლო

გაფართოებადობა პეტრის ქსელის ახალი ტიპების განსაზღვრის შესაძლებლობას.

- მეორე მოთხოვნას ენის **გრაფიკული** კომპონენტი წარმოადგენს ქსელის ელემენტების კორექტული ასახვისთვის, რადგან გრაფიკული ასახვა უმნიშვნელოვანესია პეტრის ქსელის პრაგმატიკის გასაგებად.

- ენის მესამე მოთხოვნად **დაკავშირება** მოიაზრება: ენამ უნდა უზრუნველყოს, რომ ყოველი განსაზღვრული **ჭდე** აუცილებლად ქსელის რომელიმე კომპონენტთან (**კვანძი**, **რკალი**) იყოს დაკავშირებული.

- მეოთხე მოთხოვნა **სტრუქტურიზაცია** იქნება: დიდი პეტრის ქსელურ მოდელთა ცალკეულ **მოდულებად** დაყოფა და მოდულთაშორის კავშირების განსაზღვრა

- **წაკითხვადობის** მოთხოვნა გულისხმობს, რომ ახალი ენით აღწერილი ქსელის წაკითხვა „შეუიარაღებელი“ თვალით იყოს შესაძლებელი, შუალედური ტრანსლატორების გარეშე.

ჩამოთვლილ მოთხოვნათა ხორცშესხმის ყველაზე ეფექტურ საშუალებად გვეჩვენება **პეტრის ქსელის აღწერის ენისთვის პეტრის ქსელის სხვადასხვა ტიპების ერთიანი გრამატიკის ფორმირება**.

ამგვარი მიდგომის მიზანია პეტრის ქსელის შიგთავსის გამოყოფა დამმუშავებელი პროგრამისგან და მისი შენახვა არა ჩვეულებრივი, არამედ სტრუქტურული (იერარქიული) ფაილის ფორმით, რაც მოხერხებულია პეტრის ქსელების ერთიანი გაცვლითი ფორმატის ასაგებადაც, რომელიც კონკრეტულ ინსტრუმენტზე აღარ იქნება დამოკიდებული და თავად მიაწვდის მას ინფორმაციას პეტრის ქსელის ელემენტების ფორმისა და შინაარსის შესახებ.

ამრიგად, საუბარი გვაქვს ერთგვარ **მონიშვნათა** ახალ ენაზე, რომელსაც **პეტრის ქსელის ფორმატირების (მონიშვნათა) ენას (Petri Net Markup Language - PNML)** ვუწოდებთ.

ფორმატირების ენები ინტერნეტის პროგრესთან ერთად სულ უფრო მეტ მნიშვნელობას იძენს. **PNML**-ის საფუძვლად **მონიშვნების გაფართოებული ენა XML** არის აღებული, რომელიც სადღეისოდ ყველაზე პოპულარულ ინტერნეტ-ტექნოლოგიას წარმოადგენს.

ერთერთ მომდევნო პარაგრაფში მოკლედ გვაქვს განხილული **XML** ენა, რადგან **PNML**-ის საფუძველს სწორედ მისი სინტაკსი წარმოადგენს.

2.2.2. PNML-ის კომპონენტები

ინფორმაციის ტიპები PNML-ში. **PNML**-ენის განსაზღვრისას 2.1 ქვეთავში მოცემულ განსაზღვრებათა თანახმად 2 ტიპის ინფორმაცია იქნება საჭირო: ქსელის ტიპისგან **დამოუკიდებელ** (ყველა ქსელისთვის უნიკალურ) და **დამოკიდებულ** ქსელის ელემენტებზე.

დამოუკიდებელ ელემენტებს მივაკუთვნებთ **კვანძებს (პოზიციები, გადასასვლელები)** და **რკალებს**. პეტრის ქსელის ტიპისგან დამოუკიდებლად განისაზღვრება აგრეთვე **გვერდები** და **მოდულები**.

ყოველი ელემენტისთვის, აგრეთვე მთლიანად პეტრის ქსელისთვის განვსაზღვრავთ უნიკალურ **იდენტიფიკატორს** (კოდი). იდენტიფიკატორების გამოყენების არე საკმაოდ ფართო იქნება: მაგალითად, რკალები გამოიყენებს პოზიციების და გადასასვლელთა იდენტიფიკატორებს ქსელში თავიანთი კავშირების იდენტიფიცირებისთვის.

პეტრის ქსელის ტიპზე დამოკიდებულ ელემენტს ვუწოდებთ **ჭდეს**, რომელიც ყველა ელემენტს შეიძლება ერთვოდეს და მის შინაარსს განსაზღვრავდეს.

თავისთავად ჭდის სინტაქსი პეტრის ქსელის ბირთვის შემადგენელი ნაწილია, ხოლო სემანტიკა (ჭდეების სახეობები, რაოდენობა და ნებადართული

კომბინაციები) პეტრის ქსელის ყოველი კონკრეტული ტიპისთვის ცალ-ცალკე უნდა განისაზღვროს.

ზოგადად, ჭდეების 2 ტიპის განსაზღვრა იქნება საჭირო: **წარწერებისა** და **ატრიბუტების**, სადაც წარწერა მნიშვნელობათა შეუზღუდავი სიმრავლის მქონე ჭდეს წარმოადგენს და ტექსტური ფორმით გამოისახება პეტრის ქსელის საბაზო ელემენტის გვერდით, ხოლო ატრიბუტის მნიშვნელობათა სიმრავლე შეზღუდულია.

პირველის მაგალითებია ჭდეები გადასასვლელთა გახსნის პირობის, პოზიციების სახელებისა და მარკირებათა ასახვისთვის, მეორის მაგალითად რკალის ტიპის **(**რომლის მნიშვნელობათა სიმრავლეა **IN, OUT, READ, INHIBITOR)** ამსახველი ჭდის დასახელება შეიძლება განვიხილოთ.

ამასთან ყოველ წარწერას თანმხლები გრაფიკული ინფორმაცია (ეკრანული კოორდინატები, დაშორება დაკავშირებული ელემენტიდან) ახლავს, ხოლო ატრიბუტს იგი არ გააჩნია, ატრიბუტი თვით ელემენტის გამოსახვის ფორმას განსაზღვრავს (მაგალითად, რკალი ატრიბუტით **INHIBITOR** შემაკავებელ რკალს განსაზღვრავს, რომელიც პატარა წრით ბოლოვდება).

გრაფიკული ინფორმაცია სხვადასხვა ტიპის შეიძლება იყოს: კვანძისთვის - პოზიციის კოორდინატები, რკალისთვის – საშუალედო პოზიციების კოორდინატთა სია, ანოტაციისთვის – ფართობითი პოზიცია შესაბამისი ობიექტის მიმართ (დაშორება).

გრაფიკულ ინფორმაციად ითვლება აგრეთვე მონაცემები ზომის, ფერის, კვანძების და რკალების ფორმის შესახებ ან ჭდეების ფერზე, შრიფტზე და შრიფტის ზომაზე.

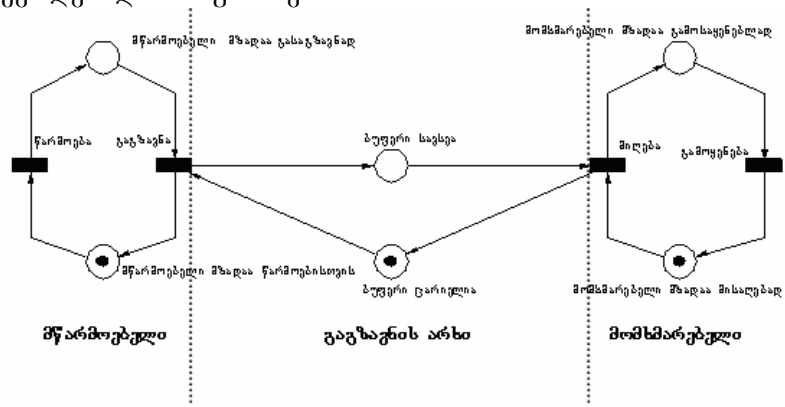
პეტრის ქსელის ზოგი ინსტრუმენტი შეიძლება შეიცავდეს **ინსტრუმენტის სპეციფიკურ ინფორმაციას**, რომელსაც სხვა ინსტრუმენტები ვერ ცნობს. ამგვარი ინფორმაციის შესანახად ყოველი ობიექტი და ჭდე უნდა აღიჭურვოს **ინსტრუმენტის სპეციფიკური ინფორმაციის**

ბლოკით, რომლის ფორმატი კონკრეტულ ინსტრუმენტზეა დამოკიდებული და **PNML**-ით არ აღიწერება.

სტრუქტურიზაცია PNML-ენის სინტაქსში აუცილებელია, იგი პეტრის ქსელის ცალკეულ ნაწილებად, ანუ **გვერდებად** დაყოფას გულისხმობს. **გვერდი** არის ობიექტი, რომელიც პეტრის ქსელის ელემენტების ქვესიმრავლეს ან/და სხვა გვერდებს შეიცავს. ამასთან, გვერდის საზღვარი შეიძლება გადიოდეს კვანძებზე, მაგრამ არა რკალებზე ანუ რკალს მხოლოდ ერთი გვერდის ფარგლებში მოთავსებული კვანძების დაკავშირება შეუძლია, როგორც ეს 2.3 ნახაზზე „მწარმოებელ-მომხმარებლის“ სისტემისთვისაა ნაჩვენები.

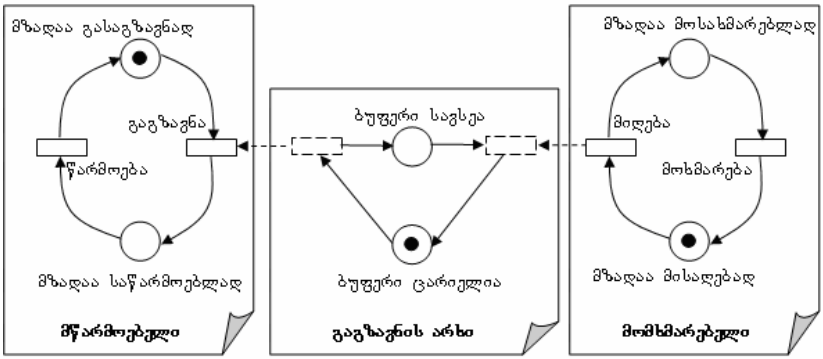
მიუხედავად იმისა, რომ გვერდების საზღვრები ყოველთვის კვანძებზე გადის, არ შეიძლება სასაზღვრო კვანძი ორივე გვერდს ეკუთვნოდეს, ამიტომ შემოგვაქვს **მაჩვენებელი კვანძის** ცნება, რომელიც **მიზნის კვანძზე** მაჩვენებელს წარმოადგენს და ზუსტად ერთ პოზიციაზე (**მაჩვენებელი პოზიცია**) ან გადასასვლელზე (**მაჩვენებელი გადასასვლელი**) მიუთითებს.

სასაზღვრო კვანძი ერთ რომელიმე გვერდზე იქნება განთავსებული, ხოლო მაჩვენებელი კვანძები მასზე სხვა გვერდებიდან მიუთითებს.



ნახ.2.3. პეტრის ქსელის დაგვერდვის მაგალითი

ამგვარად მოდერნიზებული „მწარმოებელ-მომხმარებლის“ სისტემა 2.4 ნახაზზეა წარმოდგენილი, სადაც მაჩვენებელი გადასასვლელები წყვეტილხარჩოიანი მართკუთხედებით გამოისახება.



ნახ.2.4. გვერდებად დაყოფილი პეტრის ქსელი „მწარმოებელ-მომხმარებლის“ სისტემისთვის

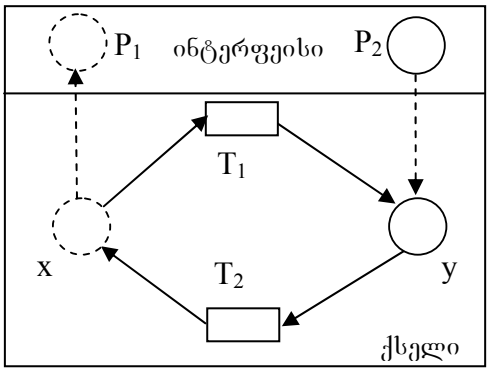
გვერდებად დაყოფილი პეტრის ქსელიდან საწყისი ქსელის მიღება ადვილია. საამისოდ მაჩვენებელი კვანძები მიზნის კვანძებს ერწყმის და გვერდების საზღვრები უგულვებელიყოფა.

უკვე არსებული პეტრის ქსელის სიმულატორებიდან დაგვერდვის ფუნქციას შეიცავს ინსტრუმენტი **CPN-Tools.**

პეტრის ქსელების სტრუქტურიზების უფრო სრულყოფილ მეთოდად **მოდულარიზაციას** წარმოვადგენთ, რაც დაგვერდვაზე რთული პროცედურაა, რადგან მოდულთაშორის კავშირები უფრო რთული ხასიათისაა.

მოდულს განვიხილავთ, როგორც პეტრის ქსელის ნაწილს დანარჩენ პეტრის ქსელთან (მოდულის **გარემო**) ინფორმაციის გაცვლის საშუალებით. მოდული შედგება **მოდულ-ქსელური** და **ინტერფეისის** ნაწილებისგან.

პირველი მოდულში შემავალ პეტრის ქსელის ფრაგმენტს შეიცავს, მეორე – მოდულის გარემოსთან კავშირის საშუალებებს. მოდულის მაგალითი 2.5 ნახაზზეა მოცემული.



ნახ.2.5. მოდული ქსელისა და ინტერფეისის ნაწილებით

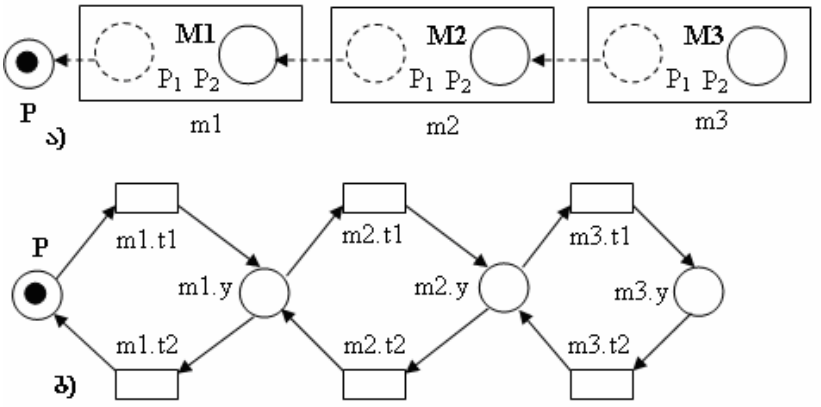
მოდულის ქსელური ნაწილიდან გარემოსთან პირდაპირი კავშირი აკრძალულია, ანუ არ შეიძლება მოდულის ქსელში იმგვარი კვანძის არსებობა, რომელზეც გარედან რომელიმე მაჩვენებელი კვანძი მიუთითებს და პირიქით, მოდულის ქსელი არ უნდა შეიცავდეს მაჩვენებელ კვანძებს, რომლებიც გარემოს კვანძებზე მაჩვენებლები იქნება.

მოდული გარემოს მხოლოდ ინტერფეისის გავლით შეიძლება დაუკავშირდეს. ჩვენს მაგალითში მოდულის ქსელი შეიცავს 2 გადასასვლელს (**T_1** და **T_2**), 1 პოზიციას (**y**) და 1 მაჩვენებელ პოზიციას (**x**). ინტერფეისისთვის განვსაზღვრავთ 2 პოზიციას: **P_1**-ს **იმპორტის პოზიციას** ვუწოდებთ და წყვეტილი წრეწირით გამოვსახავთ, **P_2 ექსპორტის პოზიცია** იქნება და ჩვეულებრივი პოზიციის რგოლის სახით გამოიტანება.

პირველი გარემოდან მოდულში მონაცემთა იმპორტის ფუნქციას შეასრულებს, მეორე – მოდულიდან გარემოში ექსპორტისა. მაჩვენებელი პოზიცია **x** წარმოადგენს მაჩვენებელს იმპორტის პოზიცია **P_1**-ზე, ხოლო ექსპორტის პოზიცია **P_2** მოდულური ქსელის **y**-პოზიციაზე.

ამგვარი მექანიზმით მოდულის შემავალი და

გამომავალი პარამეტრები განისაზღვრება, რაც კომპლექსურ პეტრის ქსელში მოდულის მრავალჯერადი გამოყენების საშუალებას იძლევა (მოდულის **ეგზემპლარების** სახით). შესაბამისი მაგალითი 2.6 ნახაზზეა ნაჩვენები.



ნახ.2.6. ა) მოდულური პეტრის ქსელი;
ბ) მოდულური პეტრის ქსელის სემანტიკა

ნახაზზე პეტრის ქსელის ასაგებად დამოუკიდებელი პოზიცია **P** და **M1**-მოდულის 3 ეგზემპლარი გამოიყენება: **m1, m2** და **m3.** ამასთან, მოდულის ქსელური ნაწილის გამოსახვისას საკმარისია მხოლოდ მოდულის ინტერფეისული ნაწილის გამოსახვა, რადგან სწორედ იგი შეიცავს მოდულის იმპორტის და ექსპორტის **პარამეტრებს** და მის გარეშე გარემოდან მოდულზე არავითარი ზემოქმედება არ ხდება.

მოდულის ეგზემპლარი **m1** პოზიცია **p**-ს თავისი იმპორტის პოზიცია $\mathbf{P_1}$-ის **პარამეტრად** იყენებს, რასაც მოდულურ პეტრის ქსელზე გრაფიკულად $\mathbf{P_1}$-დან **P**-ზე მიმართული მაჩვენებლის რკალით გამოვსახავთ. ეგზემპლარი **m2** თავისი იმპორტის პოზიცია $\mathbf{P_1}$-ის **პარამეტრად m1**-ის ექსპორტის პოზიცია $\mathbf{P_2}$-ს იღებს და ასე შემდეგ.

ნახაზის მეორე ნაწილში მოდულური პეტრის ქსელი „გაშლილი“ სახითაა წარმოდგენილი და მისი სემანტიკაა გადმოცემული.

სხვათა შორის, ამ ქსელის ყოველი კომპონენტის უნიკალურობის დასაცავად კომპონენტები მოდულის ეგზემპლარისა და კომპონენტის სახელებით აღიწერება, რომლებიც ერთმანეთისგან წერტილითაა გამოყოფილი.

2.2.3 PNML-ის სტრუქტურა

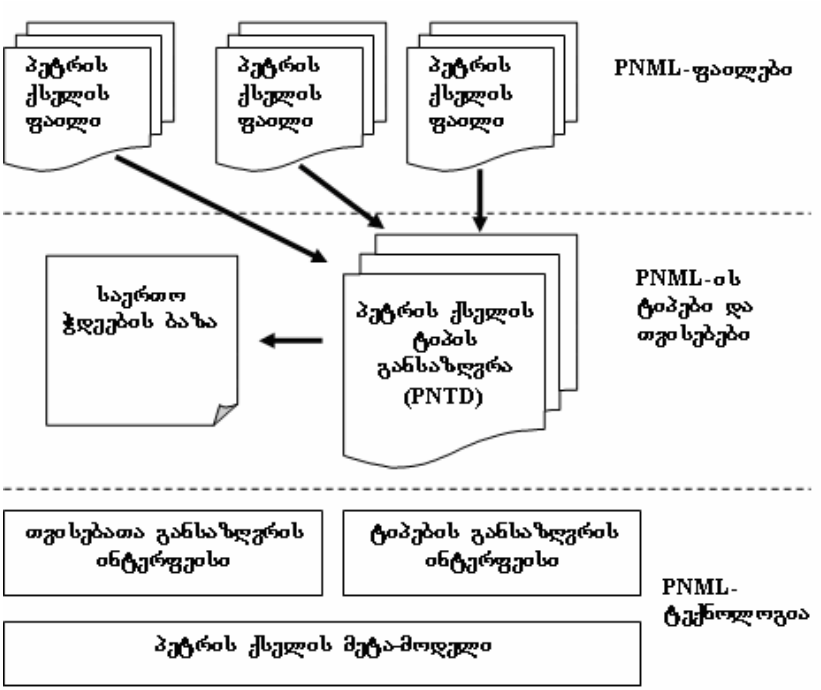
პეტრის ქსელების მონიშვნათა ენის ერთიანი სტრუქტურა 3 დონისგან შედგება. ყველაზე ქვედა დონე პეტრის ქსელის ფუნდამენტური კომპონენტების აღწერას მოიცავს, რომელი ნებისმიერი კლასისა და ტიპის პეტრის ქსელებისთვის საერთოა და განისაზღვრება მხოლოდ ერთხელ.

ამგვარ კომპონენტებს მივაკუთვნებთ პეტრის ქსელის **მეტა-მოდელს**, **თვისებათა განსაზღვრის ინტერფეისსა** და **ტიპების განსაზღვრის ინტერფეისს**.

შუალედურ დონეზე პეტრის ქსელების ენა პეტრის ქსელის ახალი ტიპების განსაზღვრის მექანიზმებს შეიცავს **საერთო ჭდეების ბაზითა** და **პეტრის ქსელის ტიპის განსაზღვრის** მექანიზმით, რომლებიც ერთმანეთთან მჭიდრო კავშირშია.

მესამე, გამოყენებითი დონე ქვედა დონეების საფუძველზე აგებულ პეტრის ქსელის სრულ ინფრასტრუქტურას, ე.წ. **პეტრის ქსელის ფაილებს** შეიცავს (ნახ.2.7). განვიხილოთ **PNML**-ის სტრუქტურული კომპონენტები დეტალურად.

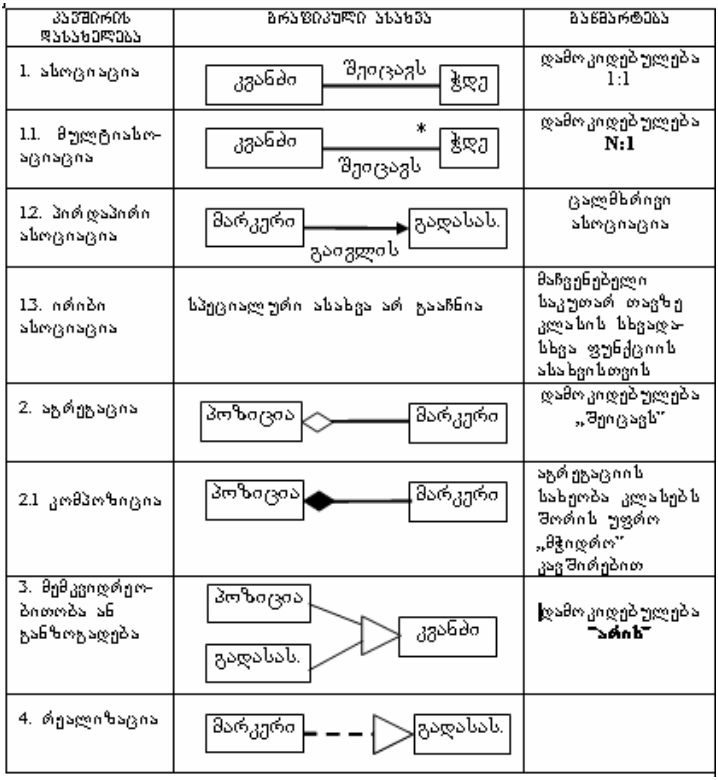
პეტრის ქსელის ფორმატირების ენის მეტა-მოდელი მის საწყის სტრუქტურას წარმოადგენს და შესრულებულია **UML**-ტექნოლოგიის გამოყენებით. **UML** იშიფრება როგორც **Unified Modelling Language** – უნიფიცირებული მოდელირების ენა.



ნახ.2.7. **PNML**-ენის ზოგადი სქემა

იგი ობიექტ-ორიენტირებული იდეოლოგიის სტანდარტიზაციის პროდუქტია და შეიცავს ობიექტ-ორიენტირებული მოდელირების სხვადასხვა საშუალებათა სიმრავლეს 4 კლასში განაწილებული 9 სხვადასხვა ტიპის დიაგრამების სახით, რომლებსაც განსხვავებული ფუნქციონალური დატვირთვები გააჩნია [2].

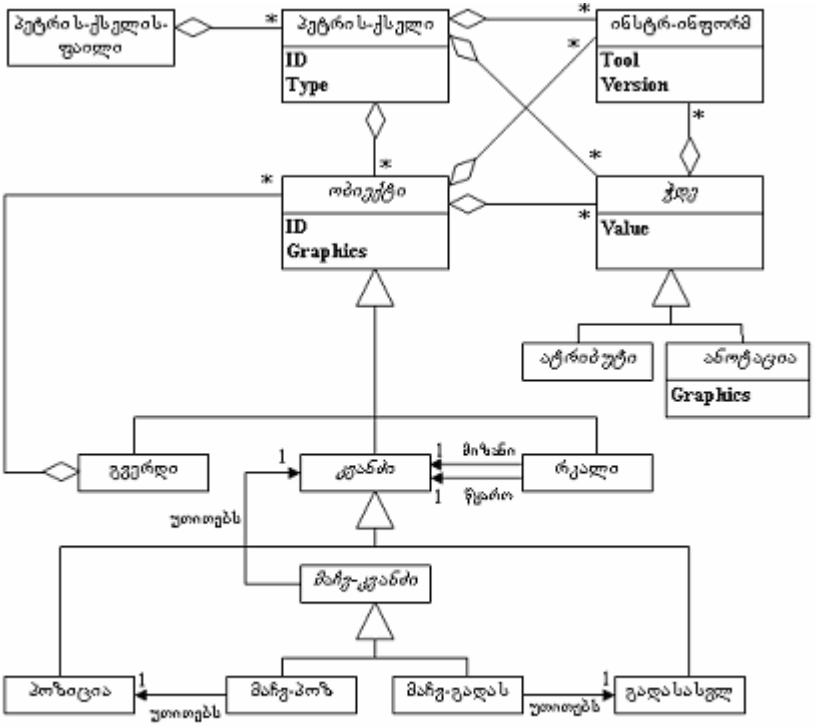
პეტრის ქსელის ენის მეტა-მოდელისთვის **კლასების დიაგრამაა** გამოყენებული, რომლის ელემენტებს **კლასები**, **ინტერფეისები** და მათ შორის დამოკიდებულებები წარმოადგენს. კლასების დიაგრამაში კავშირთა ტიპები 2.8 ნახაზზეა მოცემული პეტრის ქსელების ელემენტთა მაგალითზე.



ნახ.2.8. კავშირის ტიპები კლასების დიაგრამაში

სიტუაციის მიხედვით შესაძლებელია სხვადასხვა კლასებს შორის ერთზე მეტი დამოკიდებულების არსებობა, რაც გვაქვს კიდეც კლასების დიაგრამაზე პეტრის ქსელების მეტა-მოდელისათვის (ნახ.2.9).

ზემოთ უკვე განვსაზღვრეთ პეტრის ქსელების მონიშვნათა ენის ცალკეული კომპონენტების შინაარსი და დანიშნულება, ახლა მათ შორის კავშირების სემანტიკას აღვწერთ.



ნახ 2.9. პეტრის ქსელების (**PNML**) ენის მეტა-მოდელი

ყოველი **მაჩვენებელი პოზიცია** ან **მაჩვენებლი გადასასვლელი** წარმოადგენს მაჩვენებელ კვანძს და უთითებს შესაბამისად ერთ და მხოლოდ ერთ **პოზიციაზე** ან **გადასასვლელზე**.

პოზიცია და **გადასასვლელი** წარმოადგენს **კვანძს**, ხოლო ყოველი **რკალისთვის** არსებობს ერთი საწყისი (“**წყარო**”) და ერთი მიზნის (“**მიზანი**”) **კვანძი**.

გვერდები, **კვანძები** და **რკალები** წარმოადგენს პეტრის ქსელის **ობიექტებს**, ამასთან, **გვერდი** მოიცავს 1 ან მეტ ობიექტს (ანუ გვერდებს, კვანძებს და რკალებს).

ატრიბუტები და **ანოტაციები** არის **ჭდეები**. ობიექტი 1 ან მეტ ჭდეს, აგრეთვე პეტრის ქსელის კონკრეტული

ინსტრუმენტისთვის დამახასიათებელ ინფორმაციას შეიცავს (მინიმუმ ინსტრუმენტის დასახელებას და ვერსიას).

ყველაზე ზედა დონეს წარმოადგენს **პეტრის ქსელი**, რომელიც შედგება 1 ან მეტი ობიექტის (გვერდები, კვანძები და რკალები შესაბამისი ჭდეებით) და 1 ან მეტი საკუთარი ჭდისაგან.

და ბოლოს, **პეტრის ქსელის ფაილი** 1 ან მეტი პეტრის ქსელისგან შედგება. **ტიპების განსაზღვრის ინტერფეისი** ახალ პეტრის ქსელის ტიპს განსაზღვრავს, რომელიც მეტა მოდელის ფაილებიდან თავისთვის სასურველ სიმრავლეს იღებს.

თვისებათა განსაზღვრის ინტერფეისი პეტრის ქსელების ახალ თვისებებს განსაზღვრავს.

საერთო ჭდეების ბაზა შეიცავს პეტრის ქსელის სტანდარტულ ჭდეთა სიმრავლეს, რომლებსაც ნებისმიერი ტიპის პეტრის ქსელში უცვლელი სემანტიკა გააჩნია. საერთო ჭდეების ბაზა პეტრის ქსელის ჭდეების უმრავლესობის თავიდან განსაზღვრის პროცედურას არასაჭიროს ხდის. პეტრის ქსელის ახალი ტიპის განსაზღვრისას სტანდარტული ჭდეები შეიძლება საერთო ჭდეების ბაზიდან იქნეს აღებული, ხოლო სპეციფიკური ჭდეები პეტრის ქსელის ტიპის განსაზღვრის პროცედურით განისაზღვრება.

საერთო ჭდეების ბაზა დინამიკურად განახლებადი დოკუმენტია, რომელიც სპეციალისტთა ჯგუფის მიერ პეტრის ქსელების თეორიის განვითარებასთან ერთად ახალი სტანდარტული ჭდეებით შეიძლება შეივსოს.

პეტრის ქსელის ტიპის განსაზღვრა (PNTD – Petri Net Type Definition) წარმოადგენს პეტრის ქსელის ახალი ტიპის განსაზღვრისთვის აუცილებელ დირექტივათა ნაკრებს პეტრის ქსელის მონიშვნათა ენაზე.

პეტრის ქსელის ფაილი უკვე განსაზღვრული ტიპის პეტრის ქსელებს შეიცავს, რომლებიც თავიანთ კომპონენტებს შესაბამისი **PNTD**-ს მიხედვით აგებს.

2.3 პეტრის ქსელების ფორმატირების ენის სინტაქსი

2.3.1 XML - საფუძველი პეტრის ქსელების ფორმატირების ენისთვის

1998 წელს საერთაშორისო ორგანიზაცია **W3C**-ს მიერ ოფიციალურ სპეციფიკაციად დამტკიცების შემდეგ **XML**-ტექნოლოგია (**XML** იშიფრება, როგორც **Extensible Markup Language – მონიშვნათა გაფართოებადი ენა**) ინტერნეტის დაპყრობას აგრძელებს. სადღეისოდ არსებობს უამრავი ვებ-სერვერი, სადაც ინფორმაციის შესანახად და ასახვისთვის სწორედ **XML**-ს იყენებენ და შეიძლება ითქვას, ამ ტექნოლოგიამ ინტერნეტი თვისობრივადაც გარდაქმნა [3].

XML ჰიპერტექსტური მონიშვნების სისტემაა, ისევე, როგორც მისი წინამორბედი **HTML (Hyper Text Markup Language – ჰიპერტექსტების მონიშვნის ენა)**. ორივე ენა **სტანდარტული განზოგადებული მონიშვნების ენა SGML**-ის ქვეენას წარმოადგენს. ჰიპერტექსტის ცნება თეორიულად ჯერ კიდევ 1945 წელს ამერიკელმა მეცნიერმა ვანევარ ბუშმა დაამკვიდრა, ხოლო 60-იან წლებში ჰიპერტექსტებზე მომუშავე პროგრამებიც გამოჩნდა, რომლებიც ინფორმაციის არაწრფივ ასახვას და დამუშავებას უზრუნველყოფდა. ჰიპერტექსტური ტექნოლოგიის მასობრივი გავრცელება მხოლოდ მსოფლიო აბლაბუდის (**WWW – World Wide Web**) აგების პროცესში გახდა შესაძლებელი.

HTML-მა ინტერნეტის მსოფლიო ქსელად ქცევაში გადამწყვეტი როლი შეასრულა, მისი საშუალებით ინტერნეტში განთავსებული უზარმაზარი მოცულობის ინფორმაციის ძებნის პროცედურა მოწესრიგდა და გამარტივდა, მაგრამ ახალი ამოცანებისთვის საჭირო სიმძლავრეები **HTML**-ს არ გააჩნია. იგი წარმოადგენს სპეციალურ ინსტრუქციათა - **ტეგების** შეზღუდულ

ნაკრებს, რომლებიც, როგორც წესი, ჩვეულებრივ ტექსტურ (**.HTML**) ფაილში ინახება.

პროგრამა-ბრაუზერები (**Internet Explorer, Netscape, Mozilla, Quanta, Opera**) წაიკითხავს **HTML**-ფაილს და მასში აღწერილი ტეგების მეშვეობით ასახავს ინფორმაციას მომხმარებლის ეკრანზე. მაგრამ თანამედროვე ინტერნეტ-სერვისებისთვის პრინციპულ ამოცანად იქცა ინფორმაციის არამარტო ასახვა, არამედ სტრუქტურიზაციაც, რაც **HTML**-ს ფაქტობრივად არ შეუძლია. მასში მონაცემებს და ტეგებს ერთმანეთთან კავშირი არ გააჩნიათ, რაც ძლიერ ართულებს ინფორმაციის ანალიზს. მაგალითად, ინსტრუქცია:

```
<font color="red">Text</font>
```

ადვილი გასაგებია ნებისმიერი ბრაუზერისთვის, რომელიც ხვდება, რომ საჭიროა ტექსტის "**Text**" წითელი შრიფტით ასახვა, მაგრამ ამასთან მისთვის სულერთია, **HTML**-ფაილის რა ადგილას იქნება ზემორე ტეგი (`<font></font>`) განთავსებული, არის თუ არა იგი სხვა ტეგების ფრაგმენტი, ან თვითონ დაქვემდებარებულ ტეგებს თუ მოიცავს.

ფაქტობრივად, ინფორმაციის ძებნა და ანალიზი ჩვეულებრივი ტექსტური ფაილის ანალოგიურად უნდა მოხდეს, რაც ინფორმაციასთან მუშაობის არაეფექტურ მეთოდს წარმოადგენს.

HTML-ის მეორე ნაკლად მოუქნელობა უნდა ჩაითვალოს. მასში ტეგების სტანდარტული ნაკრები შეზღუდულია (მიუხედავად პერიოდული შევსებისა) და ბოლომდე ვერ პასუხობს სხვადასხვა ტიპის ინფორმაციულ მოთხოვნებს (მულტიმედია, მათემატიკური და ქიმიური ფორმულები და სხვა).

XML-მა ზემორე პრობლემები წარმატებით გადაჭრა. მასში არ არის წინასწარ განსაზღვრული ტეგების ნაკრები, **XML** წარმოადგენს საწყის ბაზისს (ნოტაციას) მომხმარებლის საკუთარი მონიშვნების ენისთვის [4]. **XML**-დოკუმენტიც **HTML**-ის მსგავსად ტეგების

საფუძველზე აიგება, მაგრამ მისგან პრინციპულად განსხვავდება. თავისთავად **XML**-ფაილის სტრუქტურა გაურკვეველია სტანდარტული ინტერნეტ-ბრაუზერებისთვის, მისი ასახვისთვის სპეციალური პროგრამული უზრუნველყოფაა საჭირო (მაგალითად, შუალედური პროგრამა **XML**-დოკუმენტსა და ინტერნეტ-ბრაუზერს შორის).

XML-ს გააჩნია მონაცემთა საცავებთან მიმართვის შესაძლებლობები, სადღეისოდ ინტერნეტ-პროგრამირების თითქმის ყველა გავრცელებული სისტემა (**JAVA Script, VisualBasic Script, PHP, Perl** და სხვა სკრიპტული და არა მარტო სკრიპტული ენები) ფლობს **XML**-დოკუმენტების დამუშავების საშუალებებს, რაც მანქანურ-დამოუკიდებელი პროგრამების დასაწერად საუკეთესო კომბინაციას წარმოადგენს და ინფორმაციის გაცვლის უნივერსალურ ფორმატს გვთავაზობს.

ამასთან **XML**-ში შესაძლებელია დოკუმენტებში იერარქიულად შენახული ინფორმაციის კორექტულობის კონტროლი, რომელიც სრულიად სხვადასხვა ტიპის მონაცემებისგან შეიძლება შედგებოდეს. ერთიანი კონტროლის ქონა ძალიან სასარგებლოა ინფორმაციული სისტემის შექმნის საწყის ეტაპზევე, რადგან ამით სისტემის სხვადასხვა კომპონენტების მიერ მონაცემთა სხვადასხვა ფორმატების გამოყენებით გამოწვეული შეუთავსებლობა იმთავითვე აღმოიფხვრება.

სკრიპტული ენები **XML**-დოკუმენტის დამუშავებისთვის დამატებითი პროგრამების (სკრიპტების) დაწერას მოითხოვს. ასახვის უფრო ეფექტური მეთოდია ეგრეთ წოდებული **სტილური ცხრილების გაფართოებადი ნაკრები XSL (Extensible Stylesheet Language)**, რომელიც **XML**-ის სპეციალურ ინსტრუქციათა ნაკრებს წარმოადგენს და შეიცავს **XML**-დოკუმენტის ფილტრაციის, მარტივი და რთული ძებნის და სხვა მრავალ ფუნქციას, რომლებიც **წესების** (**Rules**)

მექანიზმზეა დაფუძნებული. **XSL**-ტექნოლოგია უკვე ჩანერგილია ყველა გავრცელებულ ინტერნეტ-ბრაუზერში.

XML-დოკუმენტების დამმუშავებელი პროგრამების (მარტივად რომ ვთქვათ, **XML**-ბრაუზერების) აგება და გამოყენება, როგორ წესი, შედარებით მცირე დროს და ხარჯებს მოითხოვს. ენის ნაკლოვანებებიდან პირველ რიგში მუშაობის დაბალი სიჩქარე და მეხსიერების მაღალი ხარჯი გამოირჩევა, რაც ოპტიმიზაციის სპეციალური მეთოდებით (**Compressed XML**) ნაწილობრივ გამოსწორებადია.

ზოგად შემთხვევაში **XML**-დოკუმენტი შემდეგ ფორმატს უნდა დაექვემდებაროს:

1. დოკუმენტის სათაურში ცხადდება თვით **XML**-დოკუმეტი ვერსიის ნომრითა და სხვა დამატებითი ინფორმაციით;

2. ყოველ „გამღებ" ტეგს **აუცილებლად** უნდა მოჰყვებოდეს „დამხურავი" ტეგი (**HTML**-ში ამის აუცილებლობა არ არის).

3. **XML**-ში სიმბოლოთა რეგისტრი განირჩევა;

4. ატრიბუტების მნიშვნელობები ყოველთვის ბრჭყალებში თავსდება.

5. ტეგების „ჩადგმულობა" (იერარქია) მკაცრად კონტროლირებადია, ამიტომ "გამღები" და "დამხურავი" ტეგების მიმდევრობას გადამწყვეტი მნიშვნელობა აქვს.

საწყის და საბოლოო ტეგებს შორის განთავსებულ **მთელ** ინფორმაციას **XML** განიხილავს, როგორც მონაცემებს, ამიტომ **HTML**-ისგან განსხვავებით გაითვალისწინება ფორმატირების ელემეტებიც (ცარიელი სიმბოლო, მომდევნო სტრიქონის ნიშანი, ტაბულაცია და სხვა).

თუ **XML**-დოკუმენტი ზემორე წესებს არ არღვევს, მას **ფორმალურ-სწორი** (სინტაქსურად სწორი) დოკუმენტი ეწოდება.

ენის კონსტრუქცია. ზოგადად, **XML**-დოკუმენტი ორი მთავარი კომპონენტის, **მონიშვნის ელემენტებისა**

(**Markup**) და **მონაცემების** (**Content**) ერთობლიობას წარმოადგენს. თუ განვავრცობთ, **XML**-დოკუმენტი შედგება **ელემენტების ნაკრების, PCDATA და CDATA-სექციების, ანალიზატორის დირექტივების, კომენტარების, სპეცსიმბოლოებისა** და **ტექსტური მონაცემებისგან.**

ელემენტი **XML**-დოკუმენტის სტრუქტურული ბლოკია. ყოველი არაცარიელი ელემენტი აუცილებლად შედგება საწყისი და საბოლოო ტეგების, აგრეთვე მათ შორის მოთავსებული მონაცემებისგან. მონაცემი შეიძლება წარმოადგენდეს უბრალო ტექსტს, დოკუმენტის ჩადგმულ ელემენტს, **PCDATA** ან **CDATA**-სექციას, დამმუშავებელ ინსტრუქციას ან კომენტარს – ანუ პრაქტიკულად **XML**-დოკუმენტის ნებისმიერ ნაწილს. ელემენტების ნაკრები **XML**-დოკუმენტის იერარქიულ სტრუქტურას განსაზღვრავს. ერთი ელემენტი **ფესვურია**, პროგრამა-ანალიზატორი დოკუმენტის წაკითხვას მისგან იწყებს.

თუ ელემენტი ცარიელია (მონაცემებს არ შეიცავს), მისი საწყისი და საბოლოო ტეგები ერთიანდება შემდეგი ფორმატით: **<ტეგი/>**

კომენტარი ამოიცნობა ფორმატით: **<!--კომენტარი-->**

ატრიბუტი ელემენტის თვისებებს განსაზღვრავს, მაგალითად, მას მნიშვნელობას ანიჭებს. იგი საწყის ტეგში აისახება ფორმატით:

<ტეგი ატრიბუტი=მნიშვნელობა>.

სპეცსიმბოლოები ენის კონსტრუქციას განსაზღვრავს და მონაცემთა ბლოკში მათი პირდაპირი ჩართვა დოკუმენტის სტრუქტურას არევს, ამიტომ მისი რიცხვითი ან სიმბოლური იდენტიფიკატორის გამოყენება ხდება საჭირო.

ანალიზატორების დირექტივები გამოისახება ფორმატით: **<?დირექტივა?>**

CDATA (Character Data) მონაცემთა არეა, რომელსაც ანალიზატორი განიხილავს, როგორც უბრალო ტექსტს მასში ინსტრუქციებისა და

სპეცსიმბოლოების განურჩევლად, მაგრამ კომენტარებისგან განსხვავებით, გამოიყენებს მას სხვადასხვა მიზნებით, მაგალითად, კლიენტ-პროგრამის შესასრულებლად. **CDATA** ბლოკში ხშირად **JAVA**-სკრიპტის ინსტრუქციები თავსდება. ფორმატი:

<![CDATA] მონაცემები]]>.

PCDATA (Parseable Character Data) – ნებისმიერი ინფორმაცია, რომელთანაც პროგრამა-ანალიზატორს მუშაობა შეუძლია.

ელემენტების განსაზღვრისას მათი იერარქიაც უნდა განისაზღვროს. მშობელი ელემენტის განსაზღვრის ბლოკში მისი შიდა ელემენტები იქვე ფრჩხილებში აღიწერება სპეციალური არააუცილებელი სიმბოლოების თანხლებით: **„+”, „*”, „?”,** რომელთაგან “+” უთითებს, რომ ქვეელემენტი ელემენტში რამდენიმე ეგზემპლარად შეიძლება შედიოდეს, “?” – ელემენტი ოფციონალურია (შეიძლება საერთოდ არ იყოს წარმოდგენილი), “ * ” – შიდა ელემენტი ან ელემენტთა მიმდევრობა წარმოდგენილია რამდენჯერმე ან საერთოდ გამოიტოვება.

ენის გრამატიკასთან **ფორმალური** შესაბამისობის გარდა დოკუმენტში შეიძლება (და სასურველია) **შინაარსის** კონტროლის საშუალებებიც იყოს ჩადებული, რომლებსაც ცალკე **პროგრამა-ანალიზატორები** (ან მათი სპეციალური მოდულები) ანუ **ვერიფიკატორები** ამუშავებენ (მათ სხვანაირად **პარსერებსაც** უწოდებენ, ინგლისური სიტყვის **parse** – “ანალიზი, გარჩევა”, საფუძველზე).

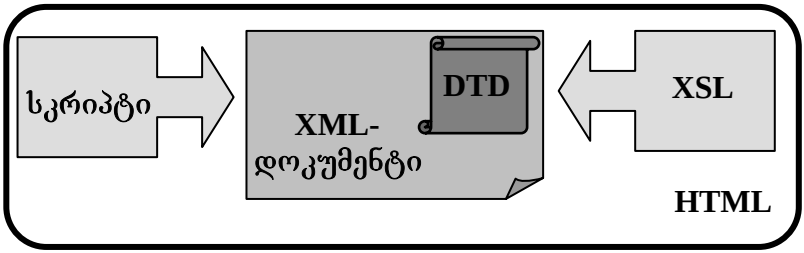
შინაარსის კონტროლი გულისხმობს **XML**-დოკუმენტის შესაბამისობის გარკვევას წინასწარ განსაზღვრულ მონაცემთა სქემებთან.

სადღეისოდ **XML**-დოკუმენტის სისწორის შემოწმების ორი გავრცელებული მეთოდი არსებობს:

- DTD-განსაზღვრებები (**Document Type Definition** – დოკუმენტის ტიპის განსაზღვრა) და

- მონაცემთა სემანტიკური სქემა (**Semantic Schema**), რომლებიც შეიძლება **XML**-ფაილშივე ჩაისვას ან ცალკე ფაილის სახით გაფორმდეს (მაგალითად, **DTD**-სთვის გამოიყენება ტეგი **<!DOCTYPE "ფესვური ელემენტი" SYSTEM "DTD-ფაილის სახელი"/>).**

ზემოთ მოცემული აღწერილობიდან გამოდინარე, **XML**-დოკუმენტის (და შესაბამისად, **პეტრის ქსელების მონიშვნათა ენაზე** აღწერილი დოკუმენტის) კავშირი სხვა ინტერნეტ-ტექნოლოგიებთან 2.10 ნახაზზე მოცემული სქემით შეიძლება გამოისახოს.



ნახ.2.10. **XML**-დოკუმენტის
დამუშავების სქემის ერთი ვარიანტი

ქვემოთ, 1-ელ ლისტინგში მოცემულია მარტივი **XML**-დოკუმენტის მაგალითი.

ფრაგმენტის პირველი ნაწილი დოკუმენტის ტიპების განსაზღვრის ინტერფეისია (**DTD**), რომელიც **XML**-დოკუმენტში გარე ფაილის სახით ჩაისმება.

დირექტივა:

```
<!DOCTYPE Saqartvelo SYSTEM "Saqartvelo.dtd">
```

განსაზღვრავს იმ ელემენტთა და მათი ატრიბუტების სიმრავლეს, რომლებიც შემდეგ **XML**-დოკუმენტში ნებადართული იქნება.

```
<!-- DTD-gansazRvrebebi -->
<!ELEMENT qvekana
          (fartobi,mosaxleoba,qalaqi)>
<!ATTLIST qvekana
Saxeli CDATA #REQUIRED>
<!ELEMENT fartobi (PCDATA)>
<!ATTLIST fartobi KvKm CDATA #REQUIRED>
<!ELEMENT mosaxleoba (PCDATA)>
<!ATTLIST mosaxleoba raodenoba CDATA
          #REQUIRED>
<!ELEMENT qalaqi (PCDATA,raioni)>
<!ATTLIST qalaqi id ID #REQUIRED>
<!ELEMENT raioni (PCDATA)*>
<!ATTLIST raioni id ID #REQUIRED>
<!-- XML-dokumenti -->
<?xml version="1.0" encoding="ISO-8859-1"
          standalone="no"?>
<!DOCTYPE Saqartvelo SYSTEM
          "Saqartvelo.dtd">
<qvekana Saxeli="Saqartvelo">
<fartobi KvKm="69700">afxazeTis da samxreT
          oseTis CaTvliT</fartobi>
<mosaxleoba raodenoba="3500000">faqtobrivi
          migraciis
          gaTvaliswinebiT</mosaxleoba>
<qalaqi id="1">Tbilisi
          <raioni id="1-
          1">Saburtalo</raioni>
          <raioni id="1-2">Vake</raioni>
</qalaqi>
<qalaqi id="3">Kutaisi</qalaqi>
<qalaqi id="4">Zugdidi</qalaqi>
</qvekana>
```

ლისტინგი-1. **XML**-დოკუმენტი გარე **DTD**-განსაზღვრებებით

პეტრის ქსელების მონიშვნის ენისთვის მისაღები ანალიზატორის შესარჩევად ჩვენ გავაანალიზეთ როგორც დე-ფაქტო სტანდარტებად მიღებული (**DTD**, მონაცემთა სემანტიკური სქემები), ასევე სხვა დამოუკიდებელი ანალიზატორები. საბოლოოდ არჩეულ იქნა ანალიზატორი **TREX (Tree Regular Expressions for XML),** რომელიც **DTD**-ზე ნაკლებ კომპაქტურია, სამაგიეროდ მოდულურობის თვისება გააჩნია, რაც მნიშვნელოვანია პეტრის ქსელების ერთიანი გაცვლითი ფორმატის შექმნისათვის [5]. მე-2 ლისტინგში მოცემულია **DTD**-განსაზღვრებათა ფრაგმენტი და მისი ექვივალენტური ფრაგმენტი **TREX**-ზე.

პეტრის ქსელის ტიპის ცნების განსაზღვრისა და **XML**-ენის ძირითადი ელემენტების განხილვის შემდეგ უკვე შეიძლება **PNML**-ენის სინტაკსის განსაზღვრა.

PNML-ს **(Petri Net Markup Language – პეტრის ქსელების მონიშვნის ენა)** განვმარტავთ, როგორც **XML**-ზე დაფუძნებულ პეტრის ქსელების დოკუმენტების აღწერის ენას, ანუ ყოველი **PNML**-დოკუმენტი იმავდროულად **XML**-დოკუმენტს წარმოადგენს.

შემდგომ პარაგრაფებში განვსაზღვრავთ პეტრის ქსელების ფორმატირების (მონიშვნათა) ენის სინტაკსს მისი სტრუქტურის შემადგენელი ნაწილებისთვის.

```
<!ELEMENT D (A | (B*, C)+)>
<!ELEMENT A EMPTY>
<!ELEMENT B EMPTY>
<!ELEMENT C EMPTY>
```

```
<grammar>
    <start>
        <element name="D">
            <choice>
                <ref name="A"/>
                    <oneOrMore>
                       <zeroOrMore>
                          <ref name="B"/>
                       </zeroOrMore>
                       <ref name="C"/>
                    </oneOrMore>
            </choice>
        </element>
    </start>
    <define name="C">
       <element name="C"><empty/></element>
    </define>
    <define name="B">
       <element name="B"><empty/></element>
     </define>
     <define name="A">
       <element name="A"><empty/></element>
     </define>
</grammar>
```

ლისტინგი 2. მონაცემთა კლასების განსაზღვრა **DTD** და **TREX**-ზე

2.3.2. პეტრის ქსელების მეტა-მოდელის სინტაკსი

მეტა-მოდელის სინტაქსის ელემენტები (**PNML**-ის **გასაღებური** ელემენტები) 2.7 ნახაზზე აგებული კლასების დიაგრამის საფუძველზე განისაზღვრება. პირდაპირი შრიფტით გამოსახული კლასებისთვის ("კონკრეტული კლასები") ექვივალენტური **XML**-ელემენტები 2.3 ცხრილშია მოცემული.

მეტა-მოდელის ტრანსლაცია PNML-ის ელემენტებში ცხრ2.3.

კლასი	XML-ელემენტი	XML-ატრიბუტი
პეტრის-ქსელის-ფაილი	<pnml>	
პეტრის-ქსელი	<net>	id: ID type: anyURI
პოზიცია	<place>	id: ID
გადასასვლელი	<transition>	id: ID
რკალი	<arc>	id: ID source: IDRef(Node) target: IDRef(Node)
გვერდი	<page>	id: ID
მოდული	<module>	id: ID
მაჩვენებელი-პოზიცია	<referencePlace>	id: ID ref: IDRef(Place or RefPlace)
მაჩვენებელი-გადასასვლელი	<referenceTransition>	id: ID ref: IDRef(Transition or RefTrans)
ინსტრუმენტის-ინფორმაცია	<toolspecific>	tool: string version: string
მნიშვნელობა	<value>	
გრაფიკა	<graphics>	

პეტრის ქსელის ფაილის აღმნიშვნელი ელემენტი **<pnml>** პეტრის ქსელის ენაზე შედგენილი ყველა დოკუმენტის ფესვური ელემენტია, საიდანაც პროგრამა-ანალიზატორი დოკუმენტის ანალიზს იწყებს.

როგორც აღვნიშნეთ, პეტრის ქსელის დოკუმენტი შეიძლება ერთზე მეტი პეტრის ქსელის აღწერას

შეიცავდეს, ყოველი ცალკეული პეტრის ქსელი **<net>...</net>** ელემენტის ფარგლებშია მოქცეული.

კლასთა სახელები პეტრის ქსელის მეტა-მოდელის **კონკრეტული** ელემენტებისთვის მეტა-მოდელის სქემაზე (ნახ.2.9) პირდაპირი შრიფტით გვაქვს გამოსახული.

კონკრეტულ ელემენტებს მივაკუთვნებთ მათ, რომლებიც პეტრის ქსელის გრაფში კონკრეტულადაა წარმოდგენილი და რომლებიც პეტრის ქსელის შექმნის პროცესში წინასწარ განისაზღვრება: **პოზიციებს, გადასასვლელებს, რკალებს, გვერდებს, მაჩვენებელ-პოზიციებს** და **მაჩვენებელ-გადასასვლელებს**.

კურსივით გამოვსახავთ იმ ელემენტებს, რომლებსაც პეტრის ქსელში ან კონკრეტული გამოსახულება არ გააჩნია (**კვანძები, ობიექტები**), ან მათი განსაზღვრა დამატებით უნდა შესრულდეს **პეტრის ქსელის ტიპის განსაზღვრის** (**PNTD**) მექანიზმით (**ჭდეები**).

კონკრეტულ ელემენტებს **PNML**-ში ექვივალენტური ელემენტები განესაზღვრებათ (**<place>, <transition>, <arc>, <page>**). ამათგან პირველ ორს მხოლოდ იდენტიფიკაციის ატრიბუტი (**id**) გააჩნია, ისევე როგორც გვერდის ამსახველ ელემენტს, როცა **რკალის** ამსახველი ელემენტი **<arc>** ფლობს დამატებით ორ ატრიბუტს: **source** რკალის საწყის კვანძს განსაზღვრავს ("წყარო"), **target** – საბოლოოს ("მიზანი"). **მაჩვენებელი პოზიცია** (**<referencePlace>**) და **მაჩვენებელი გადასასვლელი** (**<referenceTransition>**) შეიცავს სპეციალურ ატრიბუტს **ref**, რომელიც წარმოადგენს მაჩვენებელს პოზიციაზე ან გადასასვლელზე, ან მაჩვენებელს სხვა მაჩვენებლებზე.

ელემენტი **<toolspecific>** გამოსახავს კონკრეტული ინსტრუმენტის მომსახურე ინფორმაციას პეტრის ქსელის მოცემული ელემენტისთვის (პირველ რიგში მოიცემა ინსტრუმენტის დასახელება და ვერსია, ხოლო შემდეგ მხოლოდ მითითებული ინსტრუმენტისთვის დამახასიათებელი ქმედებები);

ელემენტი **<value>** პეტრის ქსელის კომპონენტის მნიშვნელობას ინახავს (მაგალითად, პოზიციის მარკირების ან რკალის ანოტაციის მიმდინარე მნიშვნელობებს).

მეტა-მოდელის განსაზღვრაში ვათავსებთ პეტრის ქსელების მონიშვნის ენის ყველა კონკრეტული ელემენტისთვის აუცილებელ გრაფიკულ კომპონენტს (ელემენტი **<graphics>**), რომელიც თავის მომსახურე ელემენტებთან და ატრიბუტებთან ერთად ცალკე იქნება განხილული. მაგალითებში **PNML**-ის გასაღებური ელემენტები ხაზგასმით გვექნება გამოყოფილი.

2.3.3. ჭდეების განსაზღვრის სინტაკსი

PNML-ის ყველა ელემენტი, რომელიც მეტა-მოდელში არ არის განსაზღვრული (1-ელ ცხრილში არ არის მოცემული), განიხილება როგორც ჭდე ან ჭდეების კომბინაცია **PNML**-ის მოცემული ელემენტისთვის.

მაგალითად, **<initialMarking>** შეიძლება იყოს ჭდე პოზიციისთვის და მის საწყის მარკირებას ასახავდეს, **<name>** ობიექტის სახელის ამსახველი ჭდე იქნება, **<inscription>** - რკალის წარწერა.

ჭდე კომბინირებული ელემენტია და შეიძლება ქვეელემენტებს შეიცავდეს. მაგალითად, ჭდის მნიშვნელობა ინახება ელემენტში **<text>,** მაგრამ ამასთან, მისი გამოსახვა შეიძლება **XML**-ის ხითაც, თუკი მას რთული სტრუქტურა გააჩნია (ელემენტი **<structure>).**

PNML-ის ოფციონალური ელემენტი **<graphics>** განსაზღვრავს ჭდის გრაფიკულ პარამეტრებს.

ასევე ოფციონალურმა **<toolspecific>** ელემენტმა ჭდეს შეიძლება დამატებითი, კონკრეტული ინსტრუმენტისთვის ინფორმაცია შეჰმატოს.

2.3.4. გრაფიკული ელემენტების განსაზღვრის სინტაკსი

PNML-ის გრაფიკული გარსი ელემენტ **<graphics>**-ის ფარგლებში აღიწერება და შედგება რამდენიმე ქვეელემენტისგან, რომლებიც 2.4 ცხრილშია მოცემული.

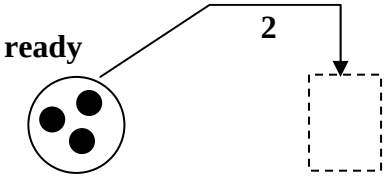
PNML-ის გრაფიკული ელემენტები ცხრ.2.4

XML ელემენტი	ატრიბუტი	დომენი
<position>	x	decimal
	y	decimal
<offset>	x	decimal
	y	decimal
<dimension>	x	nonNegativeDecimal
	y	nonNegativeDecimal
<fill>	color	RBG-color
	image	anyURI
	gradient-color	RBG-color
	gradient-rotation	{vertical, horizontal, diagonal}
<line>	shape	{line, curve}
	color	RBG-color
	width	nonNegativeDecimal
	style	{solid, dash, dot}
<font>	family	CSS2-font-family
	style	CSS2-font-style
	weight	CSS2-font-weight
	size	CSS2-font-size
	decoration	{underline, overline, line-through}
	align	{left, center, right}
	rotation	decimal

<position> პეტრის ქსელის ელემენტის აბსოლუტურ კოორდინატებს განსაზღვრავს, **<offset>** - ელემენტის წარწერის დაშორებას ელემენტისგან. **<dimension>** წარმოადგენს კვანძის ზომებს, ხოლო **<fill>** – კვანძის გაფერადების პარამეტრებს ასახავს. **<line>** – რკალის მახასიათებელი ელემენტია მისი ფორმის, ფერის, სიგანის და სტილის პარამეტრებით და ბოლოს, ელემენტი **<font>** ანოტაციათა შრიფტის დაყენებას ემსახურება.

2.3.5. პეტრის ქსელის (PNML-) ფაილის მაგალითები

წინა პარაგრაფებში მოცემული პეტრის ქსელების **მეტა მოდელის**, **ჭდეებისა** და **გრაფიკული ელემენტების** განსაზღვრებათა ერთობლიობა საშუალებას გვაძლევს ავაგოთ "სუფთა" **PNML**-ფაილი მოცემული **P/T** (კლასიკური) კლასის პეტრის ქსელისთვის (ნახ. 2.11).



ნახ.2.11. საილუსტრაციო პეტრის ქსელი **PNML**-ფაილის ასაგებად

```
<pnml>
      <net id="n1" type="PTNet">
          <name>
              <value>P/T-qselis nimuSi</value>
          </name>
            <place id="p1">
                  <graphics>
                      <position x="20" y="20"/>
                  </graphics>
                  <name>
                        <value>ready</value>
                  <graphics>
                        <offset x="10" y="-8"/>
                  </graphics>
                  </name>
                  <initialMarking>
```

```
                    <value>3</value>
                </initialMarking>
            </place>
            <transition id="t1">
                <graphics>
                    <position x="60" y="20"/>
                </graphics>
  <toolspecific tool="PetriSim" version="1.1">
                    <hidden/>
   </toolspecific>
      </transition>
       <arc id="a1" source="p1" target="t1">
              <graphics>
                  <position x="30" y="25"/>
                  <position x="60" y="25"/>
              </graphics>
              <inscription>
                   <value>2</value>
                   <graphics>
                      <offset x="15" y="-2"/>
                   </graphics>
              </inscription>
        </arc>
    </net>
</pnml>
```

ლისტინგი 3. PNML-კოდი 22-ე სურათზე ასახული კლასიკური პეტრის ქსელისთვის

ამრიგად, ყოველი **PNML**-ფაილი იწყება ტეგით **<pnml>** და სრულდება ტეგით **</pnml>**. შემდეგი ტეგია კონკრეტული პეტრის ქსელის განმსაზღვრელი **<net>**, რომელიც შეიცავს 2 ატრიბუტს: უნიკალურ იდენტიფიკატორს **(id="n1")** და **პეტრის ქსელის ტიპის** დასახელებას.

პეტრის ქსელის პირველი ელემენტია **<name>,** რომელიც ქსელს სახელს ანიჭებს, რის შემდეგაც ქსელის კომპონენტების აღწერა იწყება.

ჩვენს მაგალითში გვაქვს პეტრის ქსელის სამივე საბაზო კომპონენტი: პოზიცია (**<place>**), გადასასვლელი (**<transition>**) და რკალი (**<arc>**), რომლებსაც იდენტიფიკაციის აუცილებელ ატრიბუტებთან ერთად

საკუთარი (მაგალითად, გრაფიკული) ატრიბუტებიც გააჩნია.

კერძოდ, პოზიციისთვის ფრაგმენტი :

```
<graphics>
    <position x="20" y="20"/>
</graphics>
```

უჩვენებს, რომ პოზიციის გრაფიკული გამოსახულება (უფრო ზუსტად, გამოსახულების ანუ რგოლის ცენტრი), მოთავსებულია დეკარტეს კოორდინატთა სისტემის (20,20) კოორდინატზე, ხოლო ფრაგმენტი :

```
<name>
   <value>ready</value>
   <graphics>
      <offset x="10" y="-8"/>
   </graphics>
</name>
```

განსაზღვრავს პოზიციის სახელს (“**ready**”) და მის დაშორებას პოზიციის რგოლის ცენტრიდან (ელემენტი **<offset>**). და ბოლოს, ფრაგმენტი:

```
<initialMarking>
    <value>3</value>
</initialMarking>
```

პოზიციის საწყის მარკირებას განსაზღვრავს.

იგივე პრინციპით განისაზღვრება გადასასვლელიც, ჩვენი მაგალითისთვის მას ემატება მხოლოდ ინსტრუმენტის სპეციფიკური ინფორმაცია (ტეგი <**toolspecific**>), რომელშიც განსაზღვრული გვაქვს პეტრის ქსელის ფიქტიური ინსტრუმენტი **PetriSim** თავისი ვერსიის ნომრით.

სპეციფიკური ინფორმაციის განყოფილება პეტრის ქსელის სიმულატორთა დამპროგრამებლებისთვისაა გამოყოფილი მხოლოდ კონკრეტული ინსტრუმენტისთვის საჭირო მოქმედებათა შესასრულებლად.

ჩვენს მაგალითში ფიქტიური ინსტრუმენტი **PetriSim** გადასასვლელის დამალვის **<hidden/>-**ელემენტს შეიცავს (შესაბამისად, გადასასვლელი სურათზე წყვეტილი მართკუთხედითაა გამოსახული).

რკალის განსაზღვრას ორი დამატებითი ატრიბუტიც თან ახლავს:

source – რკალის საწყისი კვანძის უნიკალურ იდენტიფიკატორს შეიცავს,

target – საბოლოოსას.

საყურადღებოა, რომ ჯერჯერობით რკალის განსაზღვრა არ შეიცავს ინფორმაციას რკალის ტიპის შესახებ. რკალის **ტიპი** განისაზღვრება სპეციალური უხილავი ჭდის საშუალებით, რომელსაც **რკალის ატრიბუტს** ვუწოდებთ (იხილეთ პეტრის ქსელის მეტა-მოდელის აღწერა 2.3.3 პარაგრაფში) და რომელიც რკალის გრაფიკულ ფორმას განაპირობებს.

მისი ექვივალენტური ატრიბუტი **PNML**-ის რკალის განსაზღვრის არეში იქნება **type**. ქვემოთ მოცემული ფრაგმენტი რკალის ტიპს ატრიბუტის დამატებით განსაზღვრავს:

```
<arc id="a1" place="p1" transition="t1" type="in">
```

მეორე ვარიანტში **<type>** განისაზღვრება, როგორც **<arc>-**ელემენტის ქვეელემენტი:

```
<arc id="a2" place="p2" transition="t2">
        <type value="in"/>
</arc>
```

შემაკავებელი რკალისთვის (**Inhibitor Arc**):

```
<arc id="a2" place="p2" transition="t2">
        <type value="inhibitor"/>
</arc>
```

ზემო 2 ფრაგმენტში ელემენტი **<type>** ხაზგასმული არ არის, რადგან იგი არ შედის **PNML**-ის გასაღებური ელემენტების რიცხვში და **პეტრის ქსელის ტიპების განსაზღვრის** მექანიზმით დამატებით უნდა იქნეს

განსაზღვრული (ისევე, როგორც საწყისი მარკირების ელემენტი **<initialMarking>** მე-3 ლისტინგიდან).

შედარებით კომპლექსური **PNML**-ფაილის მაგალითი მოგვყავს 2.4 ნახაზზე გამოსახული, მრავალგვერდიანი მწარმოებელ-მომხმარებლის სისტემისათვის. იგი მე-4 ლისტინგზეა აღწერილი.

განსხვავება მხოლოდ პოზიციების და გადასასვლელების სახელებში იქნება, სადაც ქართული დასახელებები ინგლისური ექვივალენტებითაა ჩანაცვლებული.

```
<pnml>
 <net id="n1" type="PTNet">
   <name>
      <value>Consumer-Produser System</value>
   </name>
   <page id="pg1">
   <name><value>Producer</value></name>
   <place id="p1">
   <name><value>Ready to produce</value></name>
<initialMarking><value>0</value></initialMarking>
   </place>
      <transition id="t1">
        <name><value>produce</value></name>
      </transition>
   <place id="p2">
   <name><value>Ready to deliver</value></name>
<initialMarking><value>1</value></initialMarking>
   </place>
      <transition id="t2">
        <name><value>deliver</value></name>
      </transition>
      <arc id="a1" source="p1" target="t1">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a2" source="t1" target="p2">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a3" source="p2" target="t2">
        <inscription><value>1</value></inscription>
```

```
      </arc>
      <arc id="a4" source="t2" target="p1">
        <inscription><value>1</value></inscription>
      </arc>
    </page>
    <page id="pg2">
      <name><value>Delivery Channel</value></name>
      <referenceTransition id="rt1" ref="t2"/>
      <place id="p5">
        <name><value>Empty</value></name>
<initialMarking><value>1</value></initialMarking>
      </place>
      <place id="p6">
        <name><value>full</value></name>
<initialMarking><value>0</value></initialMarking>
      </place>
      <referenceTransition id="rt2" ref="t3"/>
      <arc id="a5" source="rt1" target="p6">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a6" source="p6" target="rt2">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a7" source="rt2" target="p5">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a8" source="p5" target="rt1">
        <inscription><value>1</value></inscription>
      </arc>
    </page>
    <page id="pg3">
      <name><value>Consumer</value></name>
      <place id="p3">
      <name><value>Ready to recieve</value></name>
<initialMarking><value>1</value></initialMarking>
      </place>
      <transition id="t3">
        <name><value>recieve</value></name>
      </transition>
      <place id="p4">
      <name><value>Ready to consume</value></name>
```

```
<initialMarking><value>0</value></initialMarking>
      </place>
      <transition id="t4">
        <name><value>consume</value></name>
      </transition>
      <arc id="a9" source="p3" target="t3">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a10" source="t3" target="p4">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a11" source="p4" target="t4">
        <inscription><value>1</value></inscription>
      </arc>
      <arc id="a12" source="t4" target="p3">
        <inscription><value>1</value></inscription>
      </arc>
    </page>
  </net>
</pnml>
```

ლისტინგი 4. PNML-ფაილი „მწარმოებელ-მომხმარებლის“ სისტემისთვის

მოცემულ **PNML**-ფაილში ელემენტი **<page>** გვერდის ასახვას ემსახურება და საკუთარი უნიკალური იდენტიფიკატორი გააჩნია. იგი სხვა ელემენტებიდან შეიძლება გამოვყოთ **<referenceTransition>**, რომელიც ერთი გვერდიდან მეორე გვერდზე განთავსებულ გადასასვლელზე მაჩვენებელს აღწერს.

რკალის ანოტაცია **<inscription>** განსაზღვრავს რკალის ანოტაციის ჭდეს და პეტრის ქსელის მონიშვნათა ენის გასაღებურ ელემენტებს არ მიეკუთვნება (უნდა განისაზღვროს **PNTD**-თი).

PNML-ენა **მოდულების** განსაზღვრის მექანიზმსაც შეიცავს, რომლის არსი 2.3.2 პარაგრაფში გადმოვეცით. პრაქტიკული რეალიზაციის თვალსაზრისით **მოდულს** განვიხილავთ, როგორც **დამოუკიდებელ პეტრის ქსელის (PNML-) ფაილს**, ხოლო მოდულთაშორის კავშირებს – პეტრის ქსელის ფაილებს შორის კავშირების სახით გამოვსახავთ, რომელთა რეალიზაცია **მაჩვენებლების**

მექანიზმით ხდება. შესაბამისად, საჭიროა **PNML**-ის ახალი ელემენტების განსაზღვრა **PNML**-ფაილში სხვა, გარე **PNML**-ფაილების იმპორტისთვის.

მოდულს განსაზღვრავს გასაღებური ელემენტი **<module>**. 2.2.2 პარაგრაფში მოცემული განსაზღვრების თანახმად, მოდული შედგება ინტერფეისისა და მოდულის ქსელისგან. **ინტერფეისს** ცალკე ელემენტი განესაზღვრება იგივე სახელით **<interface>**, ხოლო **მოდულის ქსელური** ნაწილი

<interface>…</interface>

ტეგების გარეთაა განთავსებული. ამგვარად აგებული მოდულის შესაბამისი **PNML**-ფაილი 2.5 ნახაზზე აგებული მოდულური პეტრის ქსელისთვის მე-5 ლისტინგშია ნაჩვენები. ადგილის დაზოგვის მიზნით ფაილიდან ჭდეებისა (საწყისი მარკირება, ანოტაციები) და გრაფიკული ელემენტები ამოღებულია.

```
<module id="M1">
  <name><value>M1</value></name>
  <interface>
    <importPlace id="p1"/>
    <exportPlace id="p2" ref="y"/>
  </interface>
  <referencePlace id="x" ref="p1"/>
  <transition id="t1"/>
  <transition id="t2"/>
  <place id="y"/>
  <arc source="x" target="t1"/>
  <arc source="t1" target="y"/>
  <arc source="y" target="t2"/>
  <arc source="t2" target="x"/>
</module>
```

ლისტინგი 5. **PNML**-ფაილი **M1** მოდულისთვის

როგორც ლისტინგი ცხადჰყოფს, **მოდულის ინტერფეისისთვის** განისაზღვრება **იმპორტის პოზიცია p1** (ელემენტი <**importPlace>**), რომელიც გარემოდან

მოდულში მონაცემთა იმპორტს ემსახურება და ექსპორტის პოზიცია **p2** (ელემენტი <**exportPlace>**), რომელიც მოდულის ქსელიდან გარემოში მონაცემთა ექსპორტს ასრულებს და უთითებს მოდულური ქსელის პოზიცია **y**-ზე. დავუშვათ, მოცემული მოდული შენახულია მიმდინარე კატალოგის **moduleM1.pnml** ფაილში და საჭიროა მისი ეგზემპლარების გამოყენება 2.6 ნახაზზე მოცემული მოდულური პეტრის ქსელის **PNML**-ფორმატში შესანახად. შესაბამის **PNML**-ფაილს მე-6 ლისტინგზე ნაჩვენები სახე ექნება.

```
<pnml>
  <net id="n1">
    <place id="p">
    <initialMarking>
      <value>1</value>
    </initialMarking>
    </place>
    <instance id="m1" ref="M1"
uri="file:moduleM1.pnml">
      <importPlace parameter="p1" ref="p"/>
    </instance>
    <instance id="m2" ref="M1"
uri="file:moduleM1.pnml">
      <importPlace parameter="p1" instance="m1"
ref="p2"/>
    </instance>
    <instance id="m3" ref="M1"
uri="file:moduleM1.pnml">
      <importPlace parameter="p1" instance="m2"
ref="p2"/>
    </instance>
  </net>
</pnml>
```

ლისტინგი 6. PNML-ფაილი მოდულური პეტრის ქსელისთვის

ელემენტი **<instance>** მოდულის ეგზემპლარს განსაზღვრავს. მისი ატრიბუტი **ref** მაჩვენებელია პეტრის ქსელის შესაბამის მოდულზე, ხოლო ატრიბუტი **uri**

(Unified Resource Identifier) რესურსის, ამ შემთხვევაში პეტრის ქსელის მოდულის მისამართია.

იმპორტის პოზიციისთვის განისაზღვრება **პარამეტრი** (**parameter**) და **მაჩვენებელი** (**ref**) იმპორტის კვანძზე (`<importPlace parameter="p1" ref="p"/>`) ან

მოდულის ეგზემპლარზე
(`<importPlace parameter="p1" instance="m1" ref="p2"/>`),

საიდანაც მოდულში მონაცემთა იმპორტი სრულდება.

2.3.6. პეტრის ქსელის ტიპის განსაზღვრა – PNTD

წინა პარაგრაფებში ჩვენს მიერ შემოთავაზებულ და განხილულ იქნა "სუფთა" **PNML**-ტექნოლოგია, რომელშიც ხორციელდება პეტრის ქსელების ფორმატირების ენის **გასაღებური ელემენტების** სინტაკსური კონტროლი, რითაც შესაძლებელი ხდება კორექტული საბაზო სტრუქტურის აგება პეტრის ქსელის ნებისმიერი ტიპისთვის.

მომდევნო ნაბიჯი უნდა იყოს ინფრასტრუქტურის შექმნა პეტრის ქსელის კონკრეტული ტიპის განსაზღვრისთვის, რაც **პეტრის ქსელის ტიპის განსაზღვრის მექანიზმით** (**PNTD – Petri Net Type Definition**) მიიღწევა.

ეს მექანიზმი აზრობრივად **XML**-ის სემანტიკური კონტროლის საშუალებებს (**DTD**, მონაცემთა სემანტიკური სქემები, **TREX**) ეფუძნება, რომლებსაც 2.4.1 პარაგრაფში შევეხეთ, ანუ **PNTD** წარმოადგენს საწყისი წესების სიმრავლეს (**გრამატიკას**) **სემანტიკურად სწორი პეტრის ქსელის (PNML-) ფაილების** ასაგებად.

გრაფიკულად ეს დამოკიდებულება 2.3.3 პარაგრაფში, 2.7 ნახაზზეა ნაჩვენები, სადაც **PNML**-ენის ზოგადი სტრუქტურაა მოცემული. პეტრის ქსელის ფაილები მიმართავენ **PNTD**-ს პეტრის ქსელის

კონკრეტული ტიპისთვის ნებადართულ ჭდეებზე ინფორმაციის მისაღებად, რის შემდეგაც მოცემული ტიპის პეტრის ქსელთან მუშაობისას მხოლოდ ამ ჭდეებით მანიპულირება შეუძლიათ.

მე-7 ლისტინგი **TREX**-გრამატიკის მეშვეობით (იხილეთ პარაგრაფი 2.4.1) შედგენილი პეტრის ქსელის ტიპის განსაზღვრას ასახავს **P/T** (კლასიკური) პეტრის ქსელისთვის.

```
<grammar ns="http://www.informatik.hu-
berlin.de/top/pnml"
   xmlns="http://www.thaiopensource.com/trex">
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/pnml.trex"/>
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/conv.trex"/>
      <define name="NetType" combine="replace">
            <string>PTNet</string>
      </define>
      <define name="Place" combine="interleave">
          <optional>
              <ref name="InitialMarkingString"/>
          </optional>
      </define>
      <define name="Arc" combine="interleave">
            <optional>
                  <ref
            name="InscriptionString"/>
            </optional>
      </define>
</grammar>
```

ლისტინგი 7. P/T (კლასიკური) პეტრის ქსელის ტიპის განსაზღვრა

TREX-გრამატიკის სინტაქსი **XML**-ორიენტირებულია (**DTD**-სგან განსხვავებით). მისი ფესვურია ელემენტია **<grammar>**, რომლის ატრიბუტი **ns** (**XML**-ის გასაღებური სიტყვიდან **namespace**) **PNTD**-დოკუმენტის წყაროს განსაზღვრავს (მის მისამართს ინტერნეტში), ხოლო **xmlns (XML namespace)** – ანალიზატორის მისამართს.

ელემენტი **<include>** ორჯერ გამოიყენება: პირველად პეტრის ქსელის მეტა მოდელის ფაილის, ხოლო მეორედ – საერთო ჭდეთა ბაზის (იხ.§-2.4.3) იმპორტისთვის ინტერნეტის მისამართებიდან.

ელემენტ **<define>**-ით ახალი ჭდე შემოიტანება სხვადასხვა ოფციების თანხლებით. მაგალითად, ფრაგმენტი:

```
<define name="NetType" combine="replace">
    <string>PTNet</string>
</define>
```

განსაზღვრავს ჭდეს **P/T** (კლასიკური) პეტრის ქსელისთვის, რომელიც შეიძლება **PNML**-ფაილში იქნეს იმპორტირებული, ისე, რომ ფაილში ქსელის განსაზღვრის საწყის ჭდეს (რომელიც ელემენტ **<net>**-ში განისაზღვრებოდა. იხ. ლისტინგი-3) ჩაანაცვლებს, რაზეც **combine**-ატრიბუტის მნიშვნელობა "**replace**" მიუთითებს. ხოლო ფრაგმენტით:

```
<define name="Place" combine="interleave">
    <optional>
        <ref name="initialMarkingString"/>
    </optional>
</define>
```

განისაზღვრება პოზიციის ჭდე **P/T**-ქსელში, სადაც ატრიბუტის მნიშვნელობა **combine="interleave"** უთითებს, რომ **PNTD**-ს ჭდის განსაზღვრა კი არ ჩაანაცვლებს, არამედ შეუჯერდება **PNML**-ფაილში მოცემული ანალოგიური ჭდის განსაზღვრას. ესე იგი, თუ **PNML**-ფაილში პოზიცია განისაზღვრებოდა ფრაგმენტით (პეტრის ქსელების ენის მეტა-მოდელიდან):

```
<place id="p"></place>
```

ხოლო **PNTD**-ში დაემატა ჩვენს მიერ ზემოთ მოცემული განსაზღვრება, მაშინ მისი საბოლოო სახე იქნება:

```
<place id="p">
    <initialMarkingString>
        <value>1</value>
    </initialMarkingString>
</place>
```

ამასთან ერთად, **PNTD**-ში მოცემული ელემენტი **<optional>** აზუსტებს, რომ ელემენტი ოფციონალურია, ანუ იგი შეიძლება ყველა პოზიციისთვის არ განისაზღვროს (რაც ზუსტად შეესაბამება პოზიციის მარკირების ლოგიკას).

PNTD-ში განსაზღვრული ჭდეების იმპორტი **PNML**-ფაილში შესაძლებელი რომ გახდეს, **PNML**-ის ფესვურ ელემენტ **<pnml>**-ს და ცალკეული პეტრის ქსელის საწყის ელემენტ **<net>**-ს შესაბამისი მაჩვენებლები უნდა ჩაემატოს, მაგალითად, ქვემოთ მოცემული ფრაგმენტის სახით:

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
<net id="n1" type="http://www.informatik.hu-
                    berlin.de/top/pntd/ptNetb">
```

2.3.7. საერთო ჭდეთა ბაზის სინტაკსი

საერთო ჭდეთა ბაზაში სტანდარტული **ჭდეები** განისაზღვრება, რომლებიც საწყის ეტაპზე პეტრის ქსელის არცერთ საბაზო ელემენტს არ ეკუთვნის.

საერთო ჭდება ბაზა, ფაქტობრივად, წარმოადგენს შეთანხმებას იმ ჭდეების შესახებ, რომლებსაც სხვადასხვა ტიპის პეტრის ქსელები შეიძლება შეიცავდეს.

განსაზღვრული ჭდეების გარკვეული ქვესიმრავლე შემდგომ **პეტრის ქსელის ტიპების განსაზღვრის**

მექანიზმს (**PNTD**) მიეწოდება პეტრის ქსელის ტიპის მიხედვით. **PNTD**, თავის მხრივ, ამ ჭდეს პეტრის ქსელის კონკრეტულ ელემენტს მიამაგრებს და საჭიროების შემთხვევაში **PNML**-დოკუმენტს გადასცემს.

თუ **PNTD** იყენებს ჭდეებს საერთო ჭდეების ბაზიდან, მასში ბაზის ფაილის (მაგალითად, **conv.trex**) იმპორტი აუცილებელია. შემდგომ, ახალი ჭდის განსაზღვრისას შეიძლება გამოყენებულ იქნას შესაბამისი მითითება საერთო ჭდეების ბაზაზე, როგორც ქვემოთმოყვანილ ფრაგმენტში:

```
<grammar ns="http://www.informatik.hu-
                                    berlin.de/top/pnml"
      xmlns="http://www.thaiopensource.com/trex">
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/pnml.trex"/>
      <include href="http://www.informatik.hu-
            berlin.de/top/pnml/conv.trex"/>
...
<define name="Place" combine="interleave">
      <interleave>
            <optional>
                  <ref name="conv:PTMarking"/>
            </optional>
            <optional>
                  <ref name="conv:Name"/>
            </optional>
      </interleave>
</define>
```

ფორმალურად, საერთო ჭდეების ბაზა შეიცავს ერთმანეთისგან მეტნაკლებად დამოუკიდებელი ჭდეების განსაზღვრებათა მიმდევრობას.

ყოველ ჭდეს უნიკალური **სახელი** ენიშნება. ამასთან ჭდის სახელი შეიძლება შესაბამისი **XML**-ელემენტის სახელს არ ემთხვეოდეს.

2.5 ცხრილში საერთო ჭდეების ბაზაში განსაზღვრულ ჭდეთა სიმრავლის ქვესიმრავლეა მოცემული.

საერთო ჭდეების ბაზის ქვესიმრავლის აღწერა PNML-ით. ცხრ2.5.

ჭდის სახელი	XML ელემენტი	დომენი (ტიპი)
Name	<name>	string
PTMarking	<initialMarking>	nonNegativeInteger
ENMarking	<initialMarking>	—
HLMarking	<initialMarking>	structured
PTCapacity	<capacity>	nonNegativeInteger
HLSort	<sort>	structured

ცხრილის პირველ სვეტში ჭდის სახელია გამოტანილი, მეორეში – შესაბამისი **XML**-ელემენტი. მე-3 სვეტი ჭდის მონაცემთა ტიპს ასახავს.

ზემოაღწერილ ჭდეთა დანიშნულება შემდეგია:

Name - წარმოადგენს მომხმარებლის განსაზღვრულ იდენტიფიკატორს ელემენტის (პოზიციის, გადასასვლელის, რკალის) სემანტიკის აღსაწერად, რომლის მნიშვნელობა ჩვეულებრივი ტექსტური სტრიქონის ტიპისაა (**string**). მისი ასახვისთვის სპეციალური ელემენტი **<text>** გამოიყენება;

PTMarking - პოზიციების საწყისი მარკირებაა **P/T (Place/Transition)** ტიპის პეტრის ქსელებში, რომელიც მნიშვნელობებს ნატურალურ (არაუარყოფით) რიცხვთა სიმრავლიდან იღებს (**nonNegativeInteger**);

ENMarking - ელემენტარულ სისტემურ ქსელებში **(EN Petri Nets)** პოზიციების საწყისი მარკირებას აღწერს და მისი ტიპი არ არის განსაზღვრული, რადგან ელემენტარულ სისტემურ ქსელებში პოზიციათა მარკირება ან ცარიელია ან შეიცავს შავი რგოლის სახის მქონე მარკერს, რომელიც გრაფიკულად უნდა განისაზღვროს;

HLMarking არის **თერმი** პოზიციების საწყისი მარკირებისთვის მაღალი დონის ქსელებში, რომელსაც კომპლექსური სტრუქტურა გააჩნია და აღიწერება შესაბამისი ტიპით **structured**. იგი შეიცავს **<text>** ელემენტს რამდენიმე სტრიქონად ან **XML**-ხის განშტოებას ანოტაციის ელემენტში **<structure>.**

PTCapacity წარმოადგენს **P/T** ტიპის ქსელებში პოზიციის მოცულობის აღმწერ ანოტაციას, რომელიც ნატურალური რიცხვის ტიპისაა;

HLSort მაღალი დონის პეტრის ქსელების პოზიციებში არსებულ მარკერთა სახეობებს აღწერს. იგი კომპლექსური სტრუქტურის მქონე ელემენტია.

საერთო ჭდეთა ბაზაში თავისი ადგილი გააჩნია ჭდეებს გადასასვლელებისთვის (მაგალითად, გადასასვლელის გახსნის პირობისთვის), რკალებს და პეტრის ქსელების მონიშვნათა ენის სხვა ელემენტებს.

ზემოთ მოყვანილი ცხრილიდან ისიც ჩანს, რომ საერთო ჭდეების ბაზის სხვადასხვაა ჭდეები ზოგჯერ **XML**-ის ერთი და იმავე ელემენტით გამოისახება (ცხრილში ელემენტი **<initialMArking>** ჭდეებისთვის **PTMarking**, **ENMarking** და **HLMarking**), რაც ბაზის ლოგიკურ სტრუქტურას არ არღვევს, რადგან ნახსენები ჭდეები პეტრის ქსელის სხვადასხვა ტიპებისთვისაა აღწერილი და ერთდროულად არასდროს გამოიყენება.

საერთო ჭდეების ბაზის ფრაგმენტი რკალის ატრიბუტებისა და ანოტაციებისთვის მე-8 ლისტინგშია მოცემული:

```
<grammar      ns="http://www.informatik.hu-berlin.de/top/pnml"
              xmlns="http://www.thaiopensource.com/trex">

    <include href="http://www.informatik.hu-
              berlin.de/top/pnml/pnml.trex"/>

              <define name="InitialMarkingString">

                    <element name="initialMarking">
                          <ref name="Annotation"/>
                    </element>
              </define>

              <define name="InscriptionString">
                    <element name="inscription">
                          <ref name="Annotation"/>
                    </element>
              </define>

              <define name="ArcType">
                    <element name="type">
                          <attribute name="value">
                               <ref name="ArcTypeValues"/>
                          </attribute>
                    </element>
              </define>

              <define name="ArcTypeValues">
                    <choice>
                          <string>normal</string>
                          <string>read</string>
                          <string>inhibitor</string>
                          <string>reset</string>
                    </choice>
              </define>
</grammar>
```

ლისტინგი 8. საერთო ჭდეთა ბაზის ფრაგმენტი

ლიტერატურა

1. პეტრის ქსელების ოფიციალური ვებგვერდი. URL: http://www.informatik.uni-hamburg.de/TGI/PetriNets/

2. ვ. რეისიგი, გ. სურგულაძე, დ. გულუა: ვიზუალური ობიექტორიენტირებული დაპროგრამების მეთოდები. გამომცემლობა „ტექნიკური უნივერსიტეტი“. თბილისი, 2002 წ.

3. А. Печерский: Язык XML - практическое введение. Центр Информационных Технологии – www.citforum.ru/internet/XML, 1999

4. A. Moeller: XML as an Interchange Format. Petri Nets International Conference, Aahrus, Danmark. Presentation, 27/06/2000.

5. Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara, 1996.