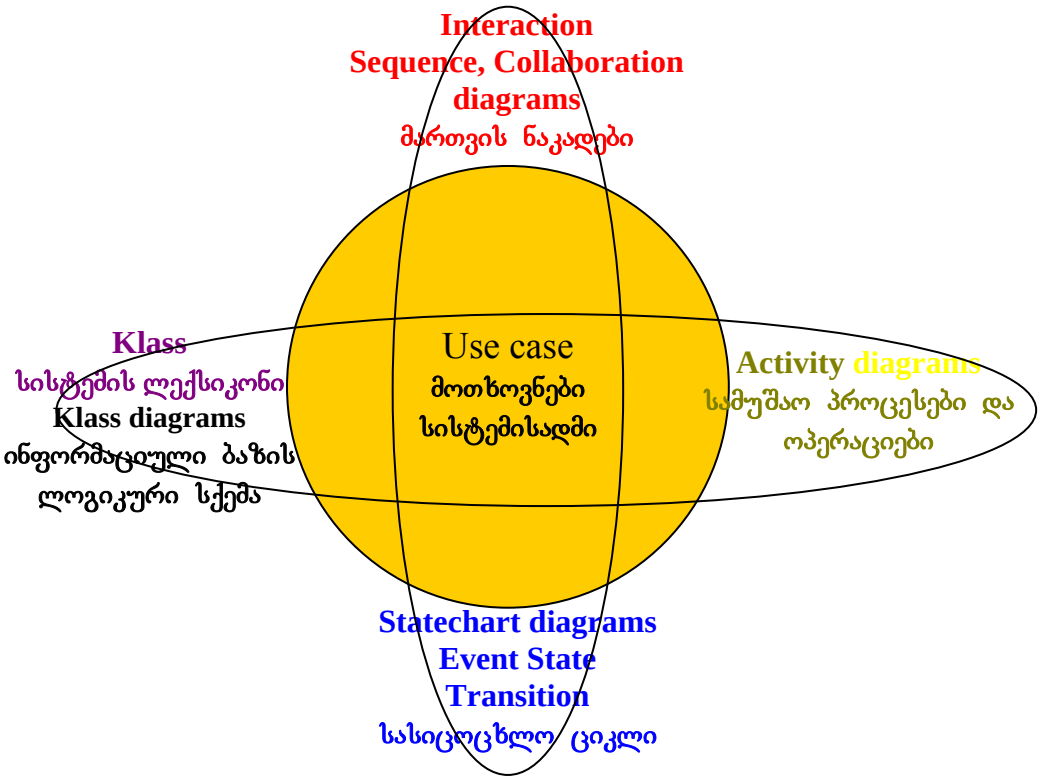
გ. გოგიჩაიშვილი, თ. სუხიაშვილი

სისტემების ობიექტ-ორიენტირებული ანალიზი და დაპროექტება

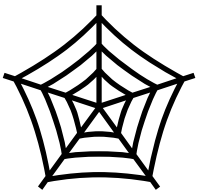


“ტექნიკური უნივერსიტეტი”

საქართველოს ტექნიკური უნივერსიტეტი

გიორგი გოგიჩაიშვილი, თეიმურაზ სუხიაშვილი

სისტემების ობიექტ-ორიენტირებული ანალიზი და დაპროექტება



დამტკიცებულია სტუ-ს სამეცნიერო-ტექნიკური საბჭოს მიერ

თბილისი 2012

უაკ 681.3.06.

მოცემულია მართვის ავტომატიზებული სისტემების აგების მეთოდიკა ობიექტ – ორიენტირებული მიდგომით თანამედროვე ინფორმაციული ტექნოლოგიების გამოყენებით.

მოყვანილია სისტემისადმი მოთხოვნების და მისი რეალიზების საშუალებების მოდელირება, რომელიც მოიცავს საპრობლემო სფეროს სტრუქტურული, ქცევითი და სისტემის არქიტექტურული მოდელირების საკითხებს. მოყვანილია მართვის ავტომატიზებული სისტემების დამუშავების ეტაპები რაციონალური უნიფიცირებული პროცესის გამოყენებით.

განკუთვნილია ინფორმატიკისა და მართვის სისტემების ფაკულტეტის სტუდენტების, მაგისტრანტებისა და სპეციალისტებისათვის.

რეცენზენტი: პროფ. ნ. ჯიბლაძე

გამომცემლობა « ტექნიკური უნივერსიტეტი », 2012

ს ა რ ჩ ე ვ ი

თ ა ვ ი 1
მართვის ორგანიზაციული სისტემების თავისებურებანი და მათი სრულყოფის ასპექტები ავტომატიზაციის საფუძველზე

მართვის ობიექტებზე გადაწყვეტილებათა მიღების პროცესების ავტომატიზაცია თანამედროვე ეტაპზე აყენებს განაწილებული ადამიანურ - მანქანური სისტემების შექმნის ამოცანას, რომელიც განკუთვნილი იქნება მრავალმიზნიანი რთული ობიექტების მართვის სრულყოფისათვის.

გამოკვლევის სიმძიმის ცენტრის აღნიშნულ პრობლემაზე გადატანა გამოწვეულია მართვის განაწილებული ობიექტების და ინფორმაციული ნაკადების დიდი დინამიკით და მეორეს მხრივ, მოდელირების სრულიად ახალი, ობიექტ – ორიენტირებული მეთოდებისა და ენების შექმნით, რაც საბოლოო ჯამში უზრუნველყოფს ასეთ ობიექტებზე გადაწყვეტილების მიღების პროცესების სრულყოფას.

განსაკუთრებულ სირთულეს წარმოადგენს ორგანიზაციული სისტემები, რომლის ელემენტია ადამიანები, ქცევის მაღალგანვითარებული ფორმებით. გარდა ამისა, ორგანიზაციულ სისტემებს ახასიათებს რამდენიმე სპეციფიკური თვისება.

პირველ რიგში, ეს მათი წარმომდგენი სტრუქტურების სირთულეა, რომლებიც შეიცავენ ელემენტებს დიდი რაოდენობის სხვადასხვა სახის მაჩვენებლებით და შესასრულებელი ფუნქციებით, მათ შორის აქტიურებს, რომლებსაც შეუძლიათ ზემოქმედება მოახდინონ თვით მართვის სისტემაზე. აქტიური ელემენტებია ადამიანები, რომლებსაც გააჩნიათ მოქმედების გარკვეული თავისუფლება სისტემის ფუნქციონირების ფარგლებში.

გარდა ამისა, სისტემისათვის განსაკუთრებულ მნიშვნელობას ღებულობს მასში შემავალ ელემენტებს შორის კავშირები. ეს კავშირები შეიძლება ატარებდეს ძალიან რთულ ხასიათს. მაგალითად, როდესაც ორ ელემენტს შორის დგინდება რამდენიმე სხვადასხვა სახის ურთიერთდამოკიდებულება.

სისტემების არსებით მხარეს წარმოადგენს მათი დინამიკურობა, ევოლუციური განვითარება. დროთა განმავლობაში იცვლება თვით ობიექტის სტრუქტურა, მისი შემცველი ელემენტების ფუნქციები. ევოლუციური განვითარება ერთის მხრივ დაკავშირებულია სისტემის გარეთ მომხდარ ცვლილებებთან, გაუთვალისწინებელი აღშფოთებითი ზემოქმედებით, ხოლო მეორეს მხრივ-შინაგანი თვითორგანიზაციით და სრულყოფით.

ავტომატიზებული სისტემის აგება გულისხმობს არსებული მართვის სისტემის შესწავლას და მისი ფუნქციონირების ადეკვატური მოდელის შექმნას. ყველა შემთხვევაში რთული სისტემების მოდელირება აუცილებელია, წინააღმდეგ შემთხვევაში მისი აღთქმა როგორც ერთიანის და მთლიანის შეუძლებელია. მოდელი იგება რათა უკეთ იქნას გაგებული დასამუშავებელი სისტემა და მისი მეშვეობით გადაწყდეს ამოცანები, რომლებიც განაპირობებენ ავტომატიზებული სისტემის ფორმირებას. ეს ამოცანებია:

- სისტემის ვიზუალირება მიმდინარე და მომხმარებლისათვის სასურველ მდგომარეობაში;
- განისაზღვროს სისტემის სტრუქტურა და ქცევა;
- შეიქმნას შაბლონი, რომელითაც შემდეგ მოხდება სისტემის კონსტრუირება;
- აგებული მოდელების გამოყენებით მოხდეს მიღებული გადაწყვეტილებების დოკუმენტირება.

§ 1.1. მოდელირებისადმი ალგორითმული და ობიექტ-ორიენტირებული მიდგომა

თანამედროვე მართვის თეორიაში ამჟამად არსებობს მრავალი მოდელი, რომლებიც აღწერს საპრობლემო სფეროს სხვადასხვა ინსტრუმენტული საშუალებით. ნებისმიერი შეიძლება მეტნაკლებად მოერგოს დასამუშავებელ სისტემას, მაგრამ ამავე დროს თითოეული მათგანი გავლენას ახდენს ავტომატიზებული სისტემის საბოლოო სახეზე და გასათვალისწინებელია, თუ რამდენად დააკმაყოფილებს იგი

მომხმარებელს. სისტემების მოდელირების არსებული მიდგომებიდან უმთავრესი და აქტუალურია ალგორითმული და ობიექტ - ორიენტირებული.

ალგორითმული მეთოდი წარმოადგენს ტრადიციულ მიდგომას ავტომატიზებული სისტემის შესაქმნელად. ძირითად სამშენებლო ბლოკებს წარმოადგენს პროცედურები და ფუნქციები, ხოლო ყურადღება პირველ რიგში ენიჭება მართვის გადაცემის საკითხებს და დიდი ალგორითმების დეკომპოზიციას მცირეზე. ცუდი ამაში არაფერია, თუ არ ჩავთვლით იმას, რომ ასეთი სისტემები არც თუ ისე მარტივად ადაპტირდებიან მოთხოვნილების ცვლილებისას ან განზომილების გაზრდისას, რაც მთავარია ამ შემთხვევაში მათი გამოყენებაც საკმაოდ რთულდება.

ავტომატიზებული სისტემის დამუშავების თანამედროვე მიდგომა არის ობიექტ-ორიენტირებული. აქ ძირითადი სამშენებლო ბლოკის სახით გამოდის ობიექტი ან კლასი. ყველაზე ზოგადი აზრით ობიექტი არსია საპრობლემო სფეროს ლექსიკონიდან, ხოლო კლასი წარმოადგენს ერთი ტიპის მქონე ობიექტთა სიმრავლის აღწერას. ყოველი ობიექტი ხასიათდება მდგომარეობით (მას უკავშირებენ გარკვეულ მონაცემებს) და ქცევით (მასზე შეიძლება რაიმეს გაკეთება ან მას თვითონ შეუძლია გააკეთოს რაიმე სხვა ობიექტზე).

ორგანიზაციული სისტემების მოყვანილი თვისებებიდან გამომდინარე, ავტომატიზაციისას გარკვეულ შემთხვევაში შეიძლება საჭირო გახდეს საპრობლემო სფეროს განზოგადებული წარმოდგენა, ხოლო სხვა შემთხვევაში მისი დეტალური აღწერა. ნებისმიერ შემთხვევაში უკეთესი მოდელი იქნება ის, რომელიც უზრუნველყოფს დეტალიზაციის საჭირო დონის ამორჩევას იმისგან დამოკიდებულებით, თუ ვინ და რა დანიშნულებით უყურებს სისტემას. ყველა შემთხვევაში მომხმარებლისათვის უფრო მეტ ინტერეს წარმოადგენს ის, თუ რის გაკეთებას შეძლებს იგი მოცემული სისტემით, ხოლო დამმუშავებლისათვის - თუ როგორ გააკეთებს ამას.

ცხადია, ყველაზე უკეთესი შემთხვევაა, როდესაც აგებული მოდელი შეესაბამება რეალობას. მაგრამ იმ შემთხვევაში როდესაც ასეთი შესაბამობა დარღვეულია, შესაძლებელი უნდა იყოს იმის დადგენა, რაში გამოიხატება განსხვავება და რა გამომდინარეობს აქედან. რამდენადაც მოდელი ყოველთვის ამარტივებს რეალობას, ამოცანა იმაშია, რომ ამ გამარტივებამ არ გამოიწვიოს რაიმე არსებითი დანაკარგები.

ორგანიზაციული სისტემების მოდელირებისას, მის ცალკეულ შემადგენელ ელემენტებს შორის რთული სემანტიკური კავშირების არსებობის გამო, მათი სრულყოფილი ასახვისათვის მიზანშეწონილია რამდენიმე მოდელის შექმნა, რომელიც აღწერს საპრობლემო სფეროს სხვადასხვა ხედვით და აბსტრაქციით. მოდელები შეიძლება შეიქმნას და შესწავლილ იქნან ერთმანეთისაგან დამოუკიდებლად, მაგრამ ამასთან ერთად ისინი რჩებიან ურთიერთდაკავშირებულნი.

ასეთი მიდგომა სავსებით მისაღებია ობიექტ-ორიენტირებული სისტემებისათვის. ობიექტ-ორიენტირებული მიდგომა პროგრამული უზრუნველყოფის დასამუშავებლად ამჟამად ყველაზე აქტუალურია, რადგან მან დაანახა სისტემოტექნიკოსებს თავისი სარგებლიანობა ნებისმიერი განზომილებისა და სირთულის სისტემების აგებისას სხვადასხვა სფეროებში. გარდა ამისა თანამედროვე დაპროგრამების ენების უმრავლესობა, ინსტრუმენტული საშუალებები და ოპერაციული სისტემები წარმოადგენენ, ამა თუ იმ ზომით, ობიექტ-ორიენტირებულებს.

თუ მივიღებთ ობიექტ- ორიენტირებულ მიდგომას, მაშინ უნდა განისაზღვროს როგორ წარიმართოს სისტემის შესწავლა და ანალიზი ავტომატიზებული სისტემის ობიექტ-ორიენტირებული არქიტექტურიდან გამომდინარე.

§ 1.2. ობიექტ-ორიენტირებული მიდგომით მართვის ორგანიზაციული სისტემების ავტომატიზაციის ეტაპები

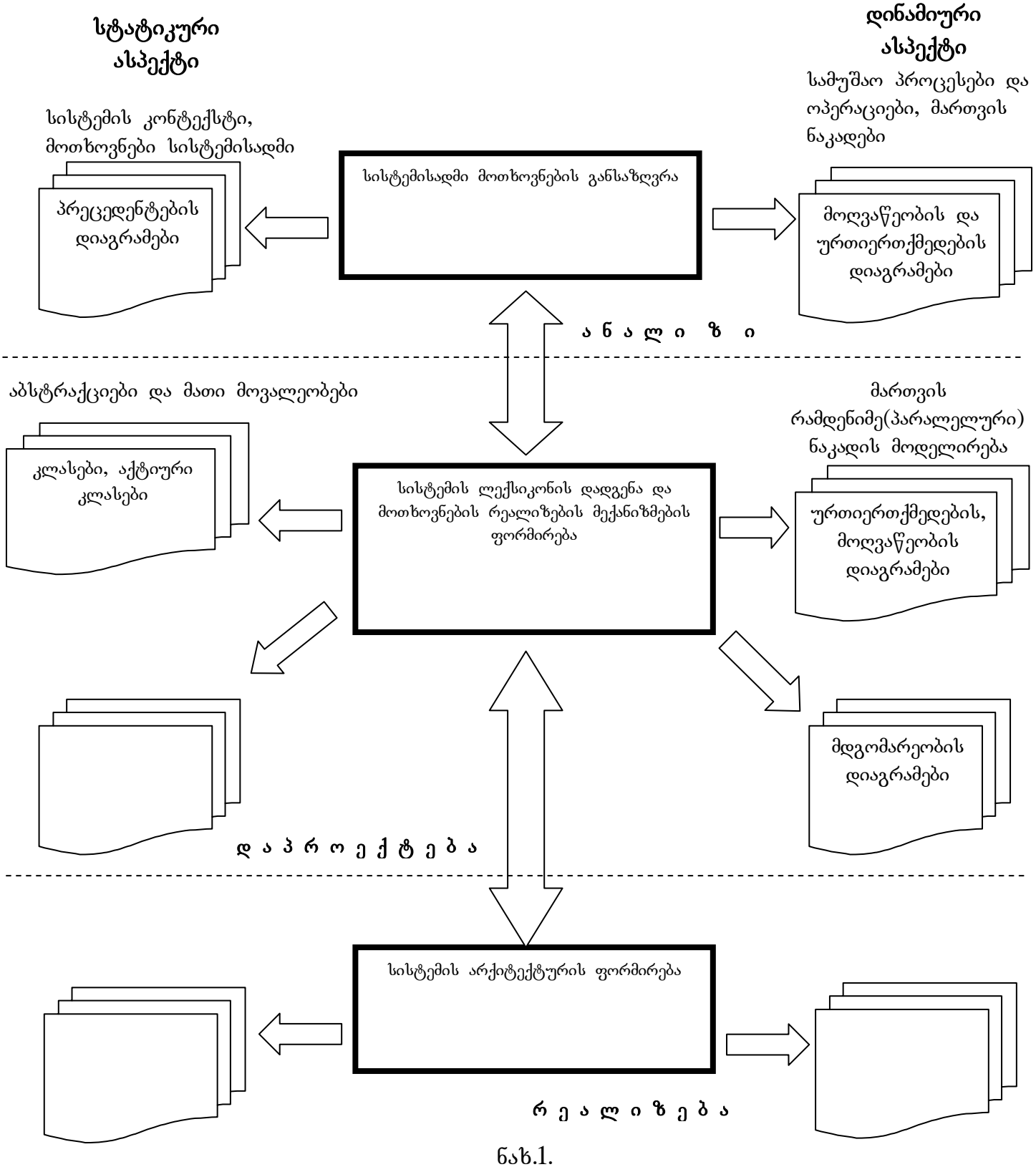
ობიექტ-ორიენტირებული მიდგომით ორგანიზაციული სისტემების ავტომატიზაციის სამი ძირითადი ეტაპი არსებობს:

- სისტემის ანალიზი, რომელიც გულისხმობს სისტემისადმი მოთხოვნების განსაზღვრას, მომხმარებელთა სამუშაო პროცესებისა და მართვის ნაკადების აღწერას;
- დაპროექტება, რომელიც გულისხმობს სისტემის ლექსიკონის დადგენას და მოთხოვნათა რეალიზების მექანიზმების ფორმირებას;
- რეალიზება, რომელიც გულისხმობს სისტემის არქიტექტურის დადგენას ანუ მარეალიზებელი კომპონენტებისა და მთლიანად სისტემის ტოპოლოგიის ფორმირებას.

მათგანი ითვალისწინებს სტრუქტურულ და ქცევით მოდელირებას (იხ.ნახ.1), ესე იგი სტატიკური და დინამიკური არსების მოდელირებას. ერთობლიობაში ეს ეტაპები საშუალებას იძლევიან გამოაშკარავდეს ყველაზე მნიშვნელოვანი გადაწყვეტილებები, რომლებიც ეხება სისტემას მთლიანობაში, ხოლო ცალკე თითოეული ყურადღებას ამახვილებს ერთ ასპექტზე, რომლის განხილვა ამ სახით მარტივდება.

ავტომატიზებული სისტემის აგებისას თავდაპირველად(***სისტემის ანალიზი***) საჭიროა დადგინდეს თუ რას უნდა აკეთებდეს სისტემა გარე მეთვალყურის თვალსაზრისით. მოთხოვნა – ეს სისტემის სასურველი ქცევაა მომხმარებელთა თვალთახედვით, რომელიც უნდა შეასრულოს სისტემამ. ამასთან მომხმარებელს მოცემულ ეტაპზე არ აინტერესებს თუ როგორ შეასრულებს სისტემა დასახულ ამოცანას. კარგათ დაპროექტებულმა სისტემამ მთლიანად უნდა შეასრულოს ყველა მოთხოვნები, ამასთან უნდა აკეთოს ეს საიმედოთ.

მოთხოვნები სისტემისადმი უნდა ჩამოყალიბდეს დამპროექტებლის მიერ იმ პირებთან(აქტიორები) ერთად, რომლებსაც უშუალო ან ირიბი კავშირი აქვთ სისტემასთან. ამ მიზნით დადგინდება სისტემის გარემოცვა – სისტემის კონტექსტი, გამოვლინდება აქტიორები და დადგინდება აქტიორებისა და მათი როლების სემანტიკა, ამასთან ერთად სისტემის სასურველი ქცევა. გამოკვლევის შედეგებს წარმოადგენენ პრეცედენტების დიაგრამის სახით.



ნახ.1.

როგორც წესი, სამუშაოს დაწყებისას მოვლენათა ნაკადებს აღწერენ ტექსტის სახით. სისტემისადმი მოთხოვნების დაზუსტების შესაბამისად გადადიან გრაფიკულ გამოსახვაზე.

აქტიორთა როლების სემანტიკის აღწერა ფაქტიურად ნიშნავს იმ სამუშაო პროცესების და ოპერაციების აღწერას, რომელსაც შესაბამისი აქტიორები ასრულებენ პრეცედენტით გათვალისწინებული მოქმედებების შესრულებისას. მათი მოდელირებისათვის გამოიყენება მოდელირება მიმდევრობითი(ზოგიერთ შემთხვევაში პარალელური) ბიჯების სახით, რომელიც აღწერს მართვის ნაკადის გადასვლას ერთი მოქმედებიდან მეორეზე (მოღვაწეობის დიაგრამა).

ყოველი მოთხოვნა რეალიზდება სხვადასხვა სცენარით. ყოველი სცენარი აღწერს ერთ გარკვეულ მართვის ნაკადს. სრულყოფილი შესწავლისათვის საჭიროა თითოეული სცენარის აღწერა. შესაბამისად, ანალიზის მოცემულ ეტაპზე უნდა მოხდეს მოთხოვნით გათვალისწინებული მართვის ნაკადების მოდელირება. ამისათვის მოცემულ ეტაპზე მოთხოვნათა აღწერისათვის მიმართავენ ქცევის მოდელირებას ურთიერთქმედებით, რომელიც გამოიხატება შეტყობინებების გაცვლაში ობიექტებს შორის (ურთიერთქმედების დიაგრამა).

დაპროექტების ეტაპზე უნდა დადგინდეს სისტემის ლექსიკონი ე.ი. დადგინდეს არსები, რომლებიც მნიშვნელოვანია როგორც მომხმარებლებისათვის ისე დამმუშავებლებისათვის. ობიექტ-ორიენტირებულ სისტემებში ასეთი აბსტრაქციები მოდელირდება კლასების სახით. ამიტომ, ყოველი აბსტრაქციისათვის უნდა დადგინდეს მისი შესაბამისი მოვალეობების სიმრავლე და დამუშავდეს ატრიბუტები და ოპერაციები, რომლებიც აუცილებელია კლასების მიერ თავიანთი მოვალეობების შესასრულებლად.

კლასები იშვიათად არსებობენ ერთმანეთისაგან დამოუკიდებლად, როგორც წესი ისინი სხვადასხვა საშუალებით ურთიერთქმედებენ ერთმანეთთან. ობიექტ - ორიეტირებული მოდელირებისას გამოყენებაშია მიმართებების სამი სახე (დამოკიდებულება, განზოგადება, ასოციაცია), რომელთა მეშვეობით უნდა აღწერილ იქნას კლასებს შორის ურთიერთ დამოკიდებულებები(კლასების დიაგრამა). რაც თავის მხრივ წარმოადგენს სისტემის მონაცემთა ბაზის კონცეპტუალური სქემის საფუძველს.

თუ გავითვალისწინებთ ორგანიზაციული სისტემების ფუნქციონირების სირთულეს, დინამიურობას და ევოლუციურ განვითარებას, უნდა გავითვალისწინოთ მოთხოვნათა ცვლილების და აგრეთვე ახალი მოთხოვნების დამატების შესაძლებლობა. ამისათვის უნდა დადგინდეს პროექტირების ნიმუშები(მექანიზმები) და არქიტექტურული ნიმუშები(კარკასები). პროექტირების ნიმუშები აღწერენ სტრუქტურას და ქცევას კლასების ნაკრებისათვის, ხოლო არქიტექტურული ნიმუშები სტრუქტურას და ქცევას მთლიანად სისტემისათვის.

მოთხოვნათა რეალიზების მექანიზმების მოდელირება ხდება კოოპერაციების მეშვეობით. კოოპერაცია მოიცავს ელემენტების(კლასების) გარკვეულ ერთობლიობას, მათ შორის დადგენილი ურთიერთქმედებით, რის შედეგადაც სრულდება შესაბამისი მოთხოვნა.

ერთობლივად მოქმედი ობიექტების ქცევის შესწავლასთან ერთად ხშირად საჭიროა ცალკეული ობიექტების ქცევის ანალიზი. ეს განსაკუთრებით ეხება ობიექტებს, რომელთა ქცევა ყველაზე კარგად გამოიხატება მათი რეაქციით საკუთარი კონტექსტის გარეთ მომხდარ მოვლენებზე(რეაქტიული ობიექტები). ასეთ ობიექტებს აქვთ მკაფიოდ გამოხატული სასიცოცხლო ციკლი. ისინი ჩნდებიან, განიცდიან ევოლუციას თავიანთი არსებობის გარკვეულ სტადიებზე და შემდეგ კვდებიან ან ქრებიან.

ასეთი ობიექტებისათვის ინტერეს წარმოადგენს პირველ რიგში - მდგრადი მდგომარეობები, - მოვლენები, რომლებიც ახდენენ გადასვლის ინიციირებას ერთი მდგომარეობიდან მეორეში და მოქმედებები, რომლებიც სრულდება მდგომარეობის შეცვლისას.

რეაქტიული ობიექტების ქცევის მოდელირებისათვის გამოიყენება მდგომარეობის დიაგრამა. იგი გვიჩვენებს ავტომატს, რომლის კერძო სახეს წარმოადგენს მოღვაწეობის დიაგრამა და გამოიყენება სამუშაო პროცესებისა და ოპერაციების მოდელირებისათვის. მაშასადამე, მოთხოვნების რეალიზების საშუალებების დადგენისას სასარგებლოა როგორც მოღვაწეობის დიაგრამა, ისე

მდგომარეობის დიაგრამა. მაგრამ თუ სამუშაო პროცესების და ოპერაციების აღწერისას (მოღვაწეობის დიაგრამა) ასახავენ მართვის ნაკადს მოღვაწეობიდან მოღვაწეობამდე, ობიექტების სასიცოცხლო ციკლების აღწერაში (მდგომარეობის დიაგრამა) წარმოდგენილია მართვის ნაკადი მდგომარეობიდან მდგომარეობამდე.

რთული ორგანიზაციული სისტემების ფუნქციონირებისას გამოიყოფა გარკვეული ეტაპები (პროცესები), რომლებიც აუცილებელია მართვის ფუნქციების განსახორციელებლად. თითოეული ამ პროცესთაგანი, ფლობს რა მოცემული პროცესისათვის დამახასიათებელ მართვის ნაკადს, თავის მხრივ შეიძლება შეიცავდეს შედარებით მცირე პროცესებს (ძაფებს). მოცემული პროცესები არ არის გამორიცხული წარიმართონ ერთდროულად (პარალელურად). განსაკუთრებით ეს მნიშვნელოვანია განაწილებული კლიენტ-სერვერული სისტემების აგებისას.

თუ მიმდევრობით სისტემებში გვაქვს მხოლოდ ერთი მართვის ნაკადი და დროის ყოველ მომენტში სრულდება ერთი და მხოლოდ ერთი მოქმედება, პარალელურ სისტემაში მართვის ნაკადები რამდენიმეა, ე.ი. დროის ერთსა და იმავე მომენტში სრულდება სხვადასხვა მოქმედება. თითოეული, ერთდროულად შესრულებადი მართვის ნაკადებიდან, იწყება დამოუკიდებელ პროცესში ან ძაფში შესვლის წერტილიდან.

პროცესის წარმოსადგენად, რომლის კონტექსტშიც სრულდება დამოუკიდებელი მართვის ნაკადი, მუშაობს სხვების პარალელურად და სარგებლობს მისი თანაბარი უფლებებით, გამოიყენება აქტიური კლასი. ჩვეულებრივი(პასიური) კლასებისაგან განსხვავებით აქტიურ კლასებს აქვთ უნარი თვითონ მოახდინონ დამოუკიდებელი მართვის ნაკადის ინიციირება.

რთული ორგანიზაციული სისტემების ავტომატიზაციისას განაწილებული სამუშაო ადგილებით უნდა დადგინდეს წესების სიმრავლე, რომლითაც დადგინდება პროცესების(აქტიური კლასების) პარალელურად და სინქრონულად შესრულება, ასევე წესების სიმრავლე, რომლითაც დადგინდება პროცესების(აქტიური კლასების) კომუნიკაცია.

ამრიგად, შესაძლებელი რომ იყოს სისტემის სასურველ ქცევაზე მსჯელობა, იქმნება პრეცედენტების დიაგრამა. საპრობლემო სფეროს ლექსიკონი აღიწერება კლასების დიაგრამის მეშვეობით. იმისათვის, რომ უჩვენონ ლექსიკონში აღწერილი არსები როგორ მუშაობენ ერთობლივად საჭირო ქცევის უზრუნველსაყოფად, სარგებლობენ მიმდევრობის, კოოპერაციის, მდგომარეობათა და მოღვაწეობის დიაგრამებით. საბოლოოდ ლოგიკური სქემები უნდა გარდაიქმნან რეალურ საგნებში, მაგალითად შესრულებად პროგრამებში, ბიბლიოთეკებში, ცხრილებში, ფაილებში და დოკუმენტებში.

ავტომატიზებული სისტემის ***რეალიზება*** ითვალისწინებს სწორედ სისტემის არქიტექტურის ფორმირებას ანუ ფიზიკური ასპექტების – კომპონენტების (ლოგიკური არსების ფიზიკური დაჯგუფება) და კვანძების (აპარატურა, რომელზედაც განლაგდებიან და სრულდებიან კომპონენტები) დაპროექტებას.

ავტომატიზებული სისტემის დამუშავება მოითხოვს მოწესრიგებულ მიდგომას იმასთან, თუ როგორ უნდა განაწილდნენ სამუშაოები და პასუხისმგებლობები ორგანიზაციაში, რომელიც დაკავებულია დამუშავების პროცესით. ობიექტ-ორიენტირებული მიდგომით პროგრამული სისტემის დამუშავება ხორციელდება ***რაციონალური უნიფიცირებული პროცესის*** საფუძველზე. აღნიშნული პროცესით ავტომატიზებული სისტემის დამუშავება უნდა განხორციელდეს ***იტერაციული*** და ***ინკრემენტული*** კვლევის საფუძველზე.

იტერაცია ეს დასრულებული ეტაპია, რომლის შედეგად გამომუშავდება პროექტის ვერსია, რომელიც ახდენს დაგეგმილი ფუნქციების ნაწილის რეალიზაციას. შემდეგ ეს ვერსია იტერაციიდან იტერაციამდე ფართოვდება მზა პროდუქციის მიღებამდე.

ინკრემენტული პროცესი გულისხმობს სისტემური არქიტექტურის მუდმივ განვითარებას ახალი ვერსიების გამოშვებისას, ამასთან ყოველი შემდეგი ვერსია წინმდებარესთან შედარებით უფრო სრულყოფილია. პროცესი, რომელიც არის ერთდროულად იტერაციული და ინკრემენტული, წარმოადგენს ***რისკით მართვადს***, რამდენადაც ამ დროს ყოველ ახალ ვერსიაში სერიოზული ყურადღება ენიჭება იმ ფაქტორების გამოვლენას, რომელიც შეიცავს უდიდეს რისკს პროექტის წარმატებით დამთავრებისათვის და მის მინიმუმამდე დასაყვანად.

თ ა ვ ი 2
სისტემის ობიექტ-ორიენტირებული ანალიზი
2.1. სისტემისადმი მოთხოვნების განსაზღვრა

ავტომატიზებული სისტემის აგებისას თავდაპირველად საჭიროა დადგინდეს თუ რას უნდა აკეთებდეს სისტემა ანუ რა მოთხოვნებს უყენებს სისტემას მომხმარებელი.

მოთხოვნა – ეს პროექტის თავისებურება, თვისება ან სისტემის ქცევაა. მოთხოვნების დადგენისას გარკვეულ წილად აღიწერება კონტრაქტების პირობები, რომელიც დაიდება სისტემასა და მის გარეთ მყოფ არსებს შორის, რომელშიც დეკლარირდება თუ რა უნდა აკეთოს სისტემამ. ამასთან, მათ არ აინტერესებს თუ როგორ შეასრულებს სისტემა დასახულ ამოცანას, არამედ ის თუ რას გააკეთებს იგი. კარგათ დაპროექტებულმა სისტემამ მთლიანად უნდა შეასრულოს ყველა მოთხოვნები, ამასთან უნდა აკეთოს ეს საიმედოთ.

ნებისმიერი სისტემა თავის შიგნით შეიცავს გარკვეულ არსებს, მაშინ როდესაც სხვა არსები რჩებიან მის გარეთ. არსები სისტემის შიგნით პასუხს აგებენ ქცევის რეალიზებაზე, რომელსაც ელოდებიან არსები გარედან. არსები, რომლებიც იმყოფებიან სისტემის გარეთ და ურთიერთქმედებენ მასთან, შეადგენენ მის კონტექსტს. მაშასადამე, კონტექსტს უწოდებენ სისტემის გარემოცვას.

მოთხოვნები სისტემისადმი უნდა ჩამოყალიბდეს ანალიტიკოსების მიერ იმ პირებთან ერთად, რომლებსაც უშუალო ან ირიბი კავშირი აქვთ სისტემასთან. მოთხოვნები, ისევე როგორც სისტემის გარემოცვა – სისტემის კონტექსტი შეიძლება გამოისახოს სხვადასხვანაირად, არასტრუქტურირებული ტექსტიდან დაწყებული, დამთავრებული ფორმალურ ენაზე წარმოდგენით. ფუნქციონალური მოთხოვნების უმეტესობა შეიძლება გამოისახოს პრეცედენტების სახით, რისთვისაც გამოიყენება პრეცედენტების დიაგრამები.

§ 2.1.1. პრეცედენტები. პრეცედენტების დიაგრამა

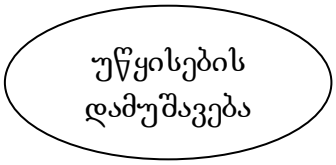
პრეცენდეტების ან გამოყენებით შემთხვევათა დიაგრამას უწოდებენ დიაგრამას, რომელზედაც ნაჩვენებია პრეცედენტებისა და აქტიორების ერთობლიობა და აგრეთვე მიმართებები მათ შორის.

პრეცედენტების დიაგრამა ჩვეულებრივ მოიცავს თავისში:

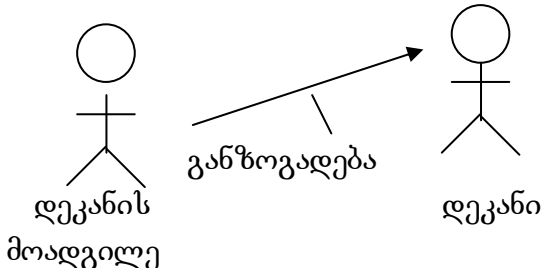
- პრეცედენტებს;
- აქტიორებს;
- მიმართებებს (დამოკიდებულების, განზოგადების და ასოციაციის).

პრეცედენტს უწოდებენ გარკვეული თანმიმდევრობის მოქმედებათა სიმრავლის აღწერას, რომელსაც ასრულებს სისტემა იმისათვის, რომ აქტიორს შეეძლოს მიიღოს განსაზღვრული შედეგი.

ყოველ პრეცედენტს უნდა გააჩნდეს სახელი, რომელიც განასხვავებს მას სხვა პრეცედენტებისაგან. გრაფიკულად პრეცედენტი გამოისახება ელიფსის სახით, რომლის შიგნით იწერება დასახელება. პრეცედენტის სახელი წარმოადგენს ტექსტურ სტრიქონს



პრეცედენტები და აქტიორები. აქტიორი წარმოადგენს დამაკავშირებელი როლების სიმრავლეს, რომელსაც პრეცედენტების მომხმარებლები ასრულებენ მათთან ურთიერთობაში. ჩვეულებრივ აქტიორი წარმოადგენს როლს, რომელსაც თამაშობს პიროვნება, მოწყობილობა ან სხვა სისტემა. აქტიორები გამოისახებიან შემდეგი სახით



აქტიორები პრეცედენტებს უკავშირდებიან მხოლოდ ასოციაციური კავშირით. ასოციაცია აქტიორსა და პრეცედენტს შორის გვიჩვენებს, რომ ისინი ურთიერთობენ, შესაძლოა უგზავნიან ან ღებულობენ ერთმანეთისაგან შეტყობინებებს.

პრეცედენტები და მოვლენათა ნაკადი. პრეცედენტი აღწერს თუ რას აკეთებს სისტემა, მაგრამ არ განსაზღვრავს თუ როგორ აკეთებს იგი ამას. მოდელირების პროცესში ყოველთვის მნიშვნელოვანია გამოვყოთ შიდა და გარე წარმოდგენა. პრეცედენტის აღწერაში სასურველია მიეთითოს, თუ როგორ და როდის იწყება და მთავრდება პრეცედენტი, როდის ურთიერთქმედებს აქტიორებთან და რომელ ობიექტებთან იცვლებიან ინფორმაციით. როგორც წესი, სამუშაოს დაწყებისას მოვლენათა ნაკადებს აღწერენ ტექსტის სახით. სისტემისადმი მოთხოვნების დაზუსტების შესაბამისად გადადიან გრაფიკულ გამოსახვაზე ურთიერთქმედების დიაგრამის მეშვეობით.

პრეცედენტები და სცენარები. ყოველ პრეცედენტს შეესაბამება მოვლენათა სიმრავლე, რომლებიც განაპირობებენ მისი განხორციელების სხვადასხვა თანმიმდევრობას, ანუ სცენარს. ამიტომ, პრეცედენტი აღიწერება არა ერთი, არამედ თანამიმდევრობათა სიმრავლით, რამდენადაც პრეცედენტისათვის დამახასიათებელი ყველა საინტერესო დეტალის გამოხატვა ერთი თანმიმდევრობით შეუძლებელია. შესაბამისად, სასურველია გამოიყოს მთავარი (ძირითადი) მოვლენათა ნაკადი ალტერნატიულებისაგან. მაგალითად, სამოქალაქო სამართალწარმოებაში პრეცედენტს "მესამე პირის ჩართვა" გააჩნია განხორციელების სხვადასხვა ვარიანტები:

- დაინტერესებული პირის მოთხოვნით, რომელიც აცხადებს დამოუკიდებელ მოთხოვნას დავის საგანზე ან მის ნაწილზე;
- დაინტერესებული პირის მოთხოვნით, რომელიც არ აცხადებს დამოუკიდებელ მოთხოვნას დავის საგანზე ან მის ნაწილზე;
- ერთ-ერთი მხარის ინიციატივით.

ყოველი ვარიანტი გამოისახება თავისი თანმიმდევრობით, ანუ სცენარით. შესაბამისად თითოეული სცენარი წარმოადგენს ერთ შესაძლო ვარიანტს მოცემულ

მოვლენათა ნაკადში. სცენარი – ეს მოქმედებათა გარკვეული თანმიმდევრობაა, რომელიც წარმოადგენს სისტემის ქცევას. სცენარები ისეთივე დამოკიდებულებაში არიან პრეცედენტებთან, როგორითაც ეგზემპლიარები კლასებთან, ე. ი. სცენარი – ეს პრეცედენტის ეგზემპლიარია.

შედარებით რთული სისტემა შეიცავს რამოდენიმე ათეულ პრეცედენტს, რომელთაგან თითოეული შეიძლება გაიშალოს რამოდენიმე ათეულ სცენარში. ყოველი პრეცედენტისათვის შეიძლება გამოიყოს ძირითადი სცენარები, რომელიც აღწერს ძირითად თანმიმდევრობას, და დამხმარე, რომლებიც აღწერენ ალტერნატიულ თანმიმდევრობებს.

პრეცედენტების ორგანიზება. პრეცედენტების ორგანიზებისათვის მათ აერთიანებენ პაკეტებში ან განსაზღვრავენ მათ შორის განზოგადების, ჩართვისა და გაფართოების კავშირებს.

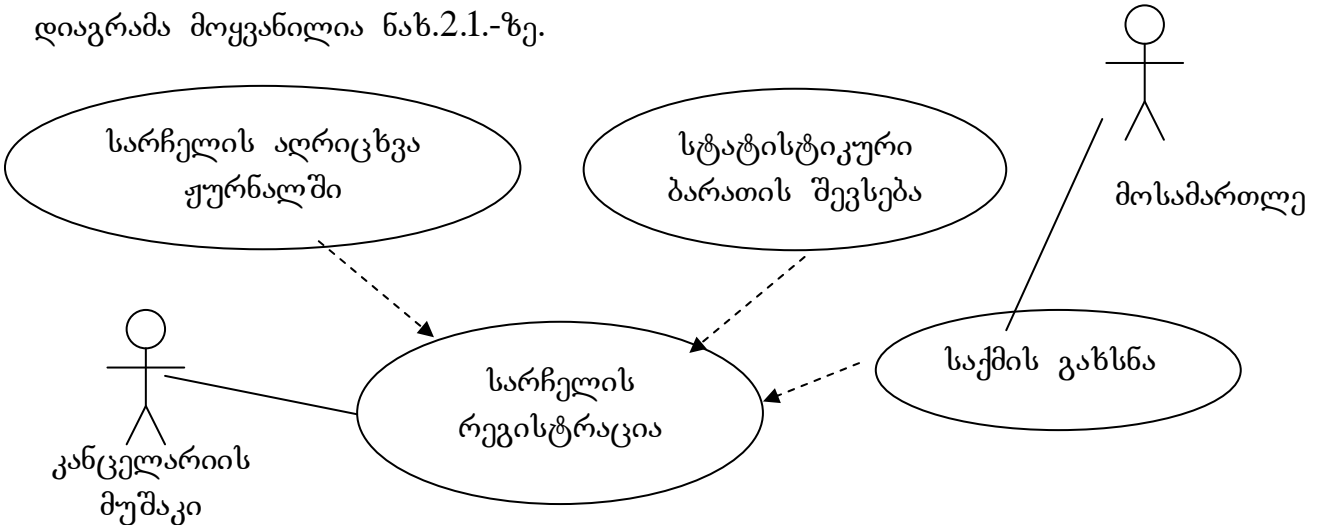
განზოგადება ნიშნავს, რომ პრეცედენტი შვილობილი მემკვიდრეობით იძენს თავისი მშობლის ქცევას და სემანტიკას, შეუძლია ჩაენაცვლოს მას ან შეავსოს მისი ქცევა.

ჩართვის მიმართება პრეცედენტებს შორის ნიშნავს, რომ ბაზური პრეცედენტის გარკვეულ წერტილში თავმოყრილია მეორე პრეცედენტის ქცევა. ჩართვადი პრეცედენტი არასდროს არ არსებობს ავტონომიურად, არამედ განიხილება როგორც მომცველი პრეცედენტის ნაწილი. შეიძლება ჩაითვალოს, რომ ბაზური პრეცედენტი ითავსებს ჩართულების თვისებებს.

გაფართოების მიმართება გულისხმობს, რომ ბაზური პრეცედენტი მოიცავს სხვა პრეცედენტის ქცევას. ბაზური პრეცედენტი შეიძლება იყოს ავტონომიურად, მაგრამ გარკვეულ პირობებში მისი ქცევა შეიძლება გაფართოვდეს მეორეს ხარჯზე.

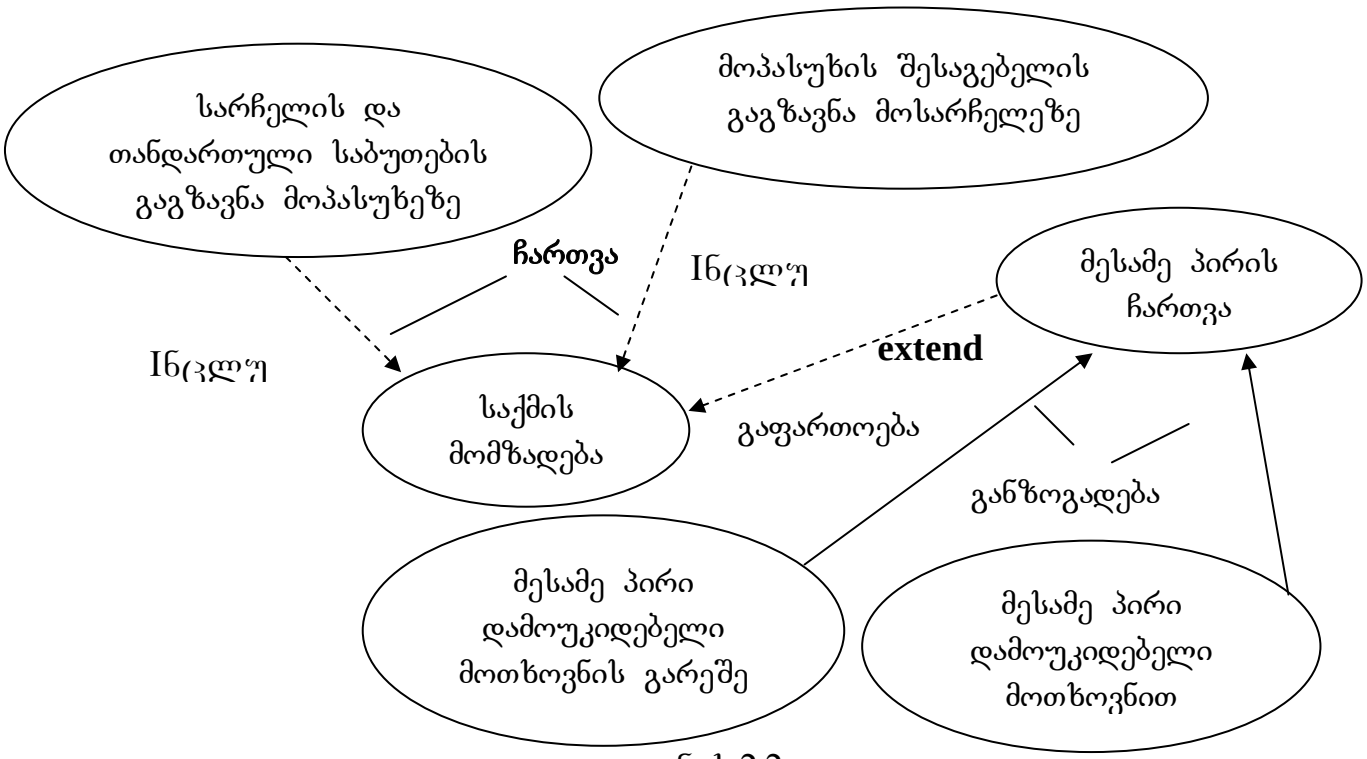
მაგალითად, სამოქალაქო სამართალწარმოებაში შესაძლებელია არსებობა პრეცედენტის ***სარჩელის რეგისტრაცია.*** მოსარჩელეს მიერ სარჩელის შეტანის შემდეგ, მოსამართლე განიხილავს და სარჩელის ხასიათის მიხედვით, ღებულობს

გადაწყვეტილებას საქმის გახსნის შესახებ, რომლის ამსახველი პრეცედენტის დიაგრამა მოყვანილია ნახ.2.1.-ზე.



ნახ.2.1.

ამავე სისტემაში მნიშვნელოვანია ***საქმის მომზადება*** მთავარ სხდომაზე განსახილველად. მოსამართლე ურთიერთქმედებს როგორც ***მოსარჩელესთან***, ისე ***მოპასუხესთან*** და გარკვეულ პირობებში შეიძლება გაფართოვდეს ***მესამე პირით***.

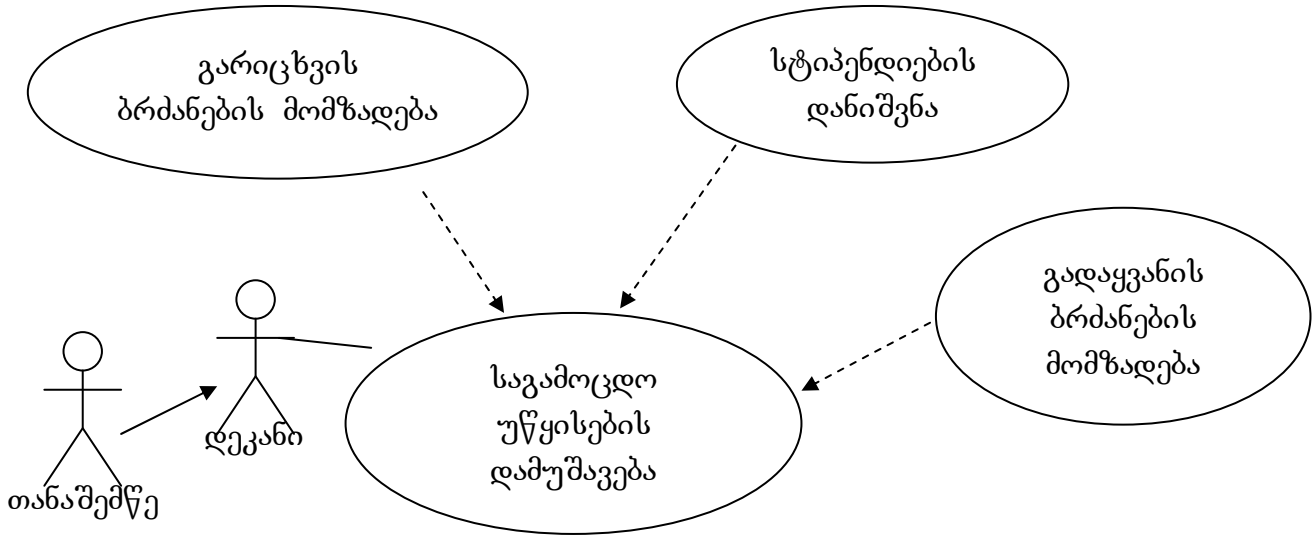


ნახ.2.2.

როგორც მოყვანილი მაგალითებიდან ჩანს განზოგადება პრეცედენტებს შორის გამოისახება ისრიანი ხაზით, ჩართვის მიმართება გამოისახება ისრიანი წყვეტილი

ხაზით სტერეოტიპით **include**, ხოლო გაფართოების მიმართება იგივე სახით სტერეოტიპით **extend**.

უმაღლეს სასწავლებლებში საგამოცდო სესიის ჩატარების შემდეგ ხდება საგამოცდო უწყისების დამუშავება სტუდენტების კურსიდან კურსზე გადაყვანისა და გასარიცხი სტუდენტების გამოვლენის მიზნით, აღნიშნულ ქცევას ასევე გამოვსახავთ პრეცედენტებით იხ. ნახ.2.3. -ზე.



ნახ.2.3.

მოყვანილ მაგალითებში შესაძლებელია გამოვყოთ საერთო ქცევა (***სარჩელის რეგისტრაცია, საქმის მომზადება, საგამოცდო უწყისების დამუშავება***) და ვარიაციები (***სარჩელის აღრიცხვა ჟურნალში, სტატისტიკური ბარათის შევსება, საქმის გახსნა, გადაყვანის ბრძანების მომზადება, გარიცხვის ბრძანების მომზადება, სტიპენდიის დანიშვნა***). თითოეული მათგანისათვის უნდა ჩაირთოს ქცევის სპეციფიკაცია თავიდან ტექსტის სახით, ხოლო ანალიზის შემდეგ ეტაპზე ავტომატით ან ურთიერთქმედებით.

§ 2.1. 2. სისტემის კონტექსტის და სისტემისადმი მოთხოვნილებების მოდელირება პრეცედენტების დიაგრამებით

პრეცენდენტების დიაგრამით შესაძლებელია სისტემის კონტექსტის მოდელირება. ამ მიზნით სისტემას შემოავლებენ წარმოსახვით ხაზს და გამოავლენენ აქტიორებს, რომლებიც იმყოფებიან ამ ხაზის იქით და ურთიერთქმედებენ სისტემასთან. პრეცენდენტების დიაგრამა ამ ეტაპზე საჭიროა აქტიორებისა და მათი როლების სემანტიკის იდენტიფიცირებისათვის, ამასთან ერთად სისტემის სასურველი ქცევის სპეციფიცირებისათვის. მნიშვნელოვანია სწორად განისაზღვროს აქტიორები, რამდენადაც ეს საშუალებას იძლევა აღვწეროთ არსების კლასები, რომლებიც ურთიერთქმედებენ სისტემასთან.

მაგალითისათვის განვიხილოთ უმაღლეს სასწავლებლებში სასწავლო პროცესის ორგანიზების პროცესი. სასწავლო პროცესის ორგანიზების მიზნით თითოეული სპეციალობის სტუდენტებისათვის დგება სასწავლო გეგმა. სასწავლო გეგმა მოიცავს მოცემულ სპეციალობაზე შესასწავლ საგნებს კურსებისა და სემესტრების მიხედვით, მათზე გამოყოფილი კრედიტების რაოდენობას, დადგენილი ნორმატივებით, სემესტრში შესასწავლ საგნებზე გამოყოფილი კრედიტების ჯამი არ უნდა აღემატებოდეს 30-ს, მთლიანად კურსზე კრედიტების ჯამი ტოლი უნდა იყოს60-ს(იხ.ნახ.2.4.ა).

გამოყოფილი კრედიტები გადაიყვანება საათებში(1 EKTS=25.2 სთ) და ნაწილდება სალექციო, პრაქტიკული, ლაბორატორიული, პრაქტიკის, სემინარის, საკურსო სამუშაოს(პროექტი) და სტუდენტთა დამოუკიდებელი სამუშაოებისათვის სემესტრის მითითებით(იხ.ნახ.2.4.ბ).

დადგენილი სასწავლო გეგმის საფუძველზე ხდება მოცემული სპეციალობის აკადემიური დატვირთვის საათების გაანგარიშება. რაც გულისხმობს სპეციალობის თითოეული კურსის საგნისათვის გეგმით გათვალისწინებული სალექციო, პრაქტიკული, ლაბორატორიული, პრაქტიკის(საწარმოო, სასწავლო), სემინარის, საკურსო სამუშაოს(პროექტი), ტესტის შედგენისა და ტესტირებაზე საათური განაკვეთების დადგენას.

საბაკალავრო პროგრამის სტრუქტურა (მოდულები, საგნები, შესაბამისი კრედიტები)

(აუცილებელი საგნები იბეჭდება თეთრ ფონზე, არჩევითი საგნები კი–მუქ ფონზე)

№	საგნის კოდი†	საგანი\ მოდული‡	I კურსი		II კურსი		III კურსი		IV კურსი		ECTS კრედიტი
			ECTS კრედიტი								
			I სემესტრი	II სემესტრი	I სემესტრი	II სემესტრი	I სემესტრი	II სემესტრი	I სემესტრი	II სემესტრი	
ECTS კრედიტები		სემესტრში									
		კურსზე									
საგნების\მოდულების რაოდენობა კურსზე											

ნახ.2.4.ა

საგანი\ მოდული / საათები	ECTS Cr\ საათი	დღეში, სთ	პრაქტიკული, სთ	ლაბორატო-რიული, სთ	პრაქტიკა, სთ	სემინარი, სთ	საკურსო პროექტი\ სამუშაო	დამოუკიდებელი სამუშაო, სთ
ჯამი								

ნახ.2.4.ბ

გაანგარიშების შემდეგ ეტაპზე კურსის თითოეული ჯგუფის კონტიგენტის და სასწავლო პროცესზე დადგენილი ნორმატივების გათვალისწინებით დგინდება ფაქტიური სალექციო, პრაქტიკული, ლაბორატორიული, პრაქტიკის, სემინარის,

საკურსო სამუშაოს(პროექტი), ტესტის შედგენისა და ტესტირებაზე საათური განაკვეთები.

ბოლოს ხდება ჯამური მნიშვნელობების ანგარიში თითოეული ჯგუფის, კურსის და მთლიანად სპეციალობისათვის ცალკეული რეკვიზიტების (ლექცია, პრაქტიკული, ლაბორატორიული, საკურსო სამუშაო-პროექტი, საწარმოო-სასწავლო პრაქტიკა, გამოცდა, ტესტის შედგენა) მიხედვით(იხ.ნახ.2.5.).

საქართველოს ტექნიკური უნივერსიტეტის 2008 /2009 სასწავლო წლის აკადემიური დატვირთვის საათების გაანგარიშება

კათედრის დასახელება და ნომერი 94

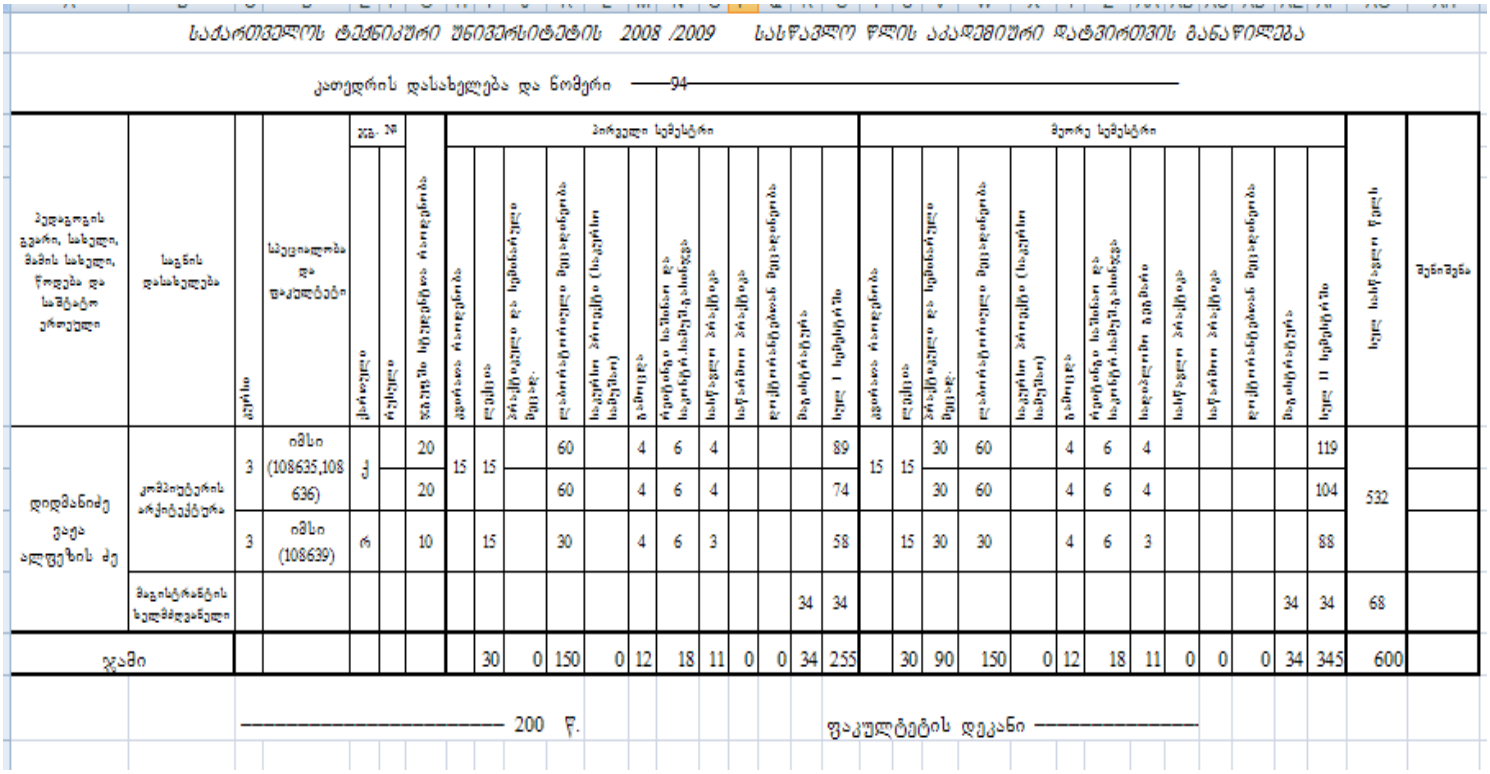
მიცადინეობის სახეობა / დისციპლინის დასახელება	კურსი	სპეციალობა, ფაკულტეტი	ჯგუფის №	სემესტრი	სტუდ. კონტინგენტი	სემესტრში კრედიტების რაოდენობა		ლექცია		პრაქტიკული ან სემინ. მეცად.		ლაბორატ. მეცად.		საკურსო პროექტი (საკურსო სამუშაო)	გამოცდა	ტესტირება	ტესტების შეფასება და შემოწმება	სასწავლო პრაქტიკა	საწარმოო პრაქტიკა	დიპლომისწინა მუშაობის	მაგისტრატურა		სულ საათების რაოდენობა			შენიშვნა
						I სემესტრი	II სემესტრი	საათები		საათები		საათები												სტუდიაზე	დაუნიშნავად სწავლებაზე	საერთო ჯამი	
								პირები	ჯგუფი	პირები	ჯგუფი	პირები	ჯგუფი													
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		17	18	19	20	21	22	23	24	25	26
საინფორმაციო ტექნოლოგიები	1	ი მ ს ფ	108835	ქ	20	5	5	15	15			45	90		4	6	13						128		482	
								15				45	90		4	6	13						113			
			108836	ქ	20			15				45	90		4	6	13						113			
								15	15			45	90		4	6	13						128			
მონაცემთა სტრუქტურები და ალგორითმები	1	ი მ ს ფ	108835	ქ	20	5	0	30	30	30	30				4	6	13						83		136	
																							0			
			108836	ქ	20	5	0	30		30	30				4	6	13						53			
																							0			
			108835	ქ	20																		0			

ნახ.2.5.

გაანგარიშებული დატვირთვა მტკიცდება ინსტიტუტის ხარისხის მართვისა და სასწავლო პროცესების დეპარტამენტში. დამტკიცების შემდეგ ხდება მისი განაწილება სპეციალობის პროფესორებზე.

განაწილებისათვის დატვირთვის უწყისში მოცემული პროფესორის კვალიფიკაციისა და სპეციალიზაციიდან გამომდინარე მოინიშნება საგანი(ლექცია, პრაქტიკული, ლაბორატორიული, პრაქტიკა, სემინარი, საკურსო სამუშაო(პროექტი), ტესტის შედგენა და ტესტირება), შესაბამისი ჯგუფებით. მონიშნული საგანი შესაბამისი საათური განაკვეთით გადაიტანება პროფესორის

ინდივიდუალური დატვირთვის დოკუმენტში. მოყვანილი პროცესი გრძელდება მანამდე, სანამ საათური განაკვეთის ჯამური რაოდენობა არ გახდება ინდივიდუალური დატვირთვის ნორმირებული რაოდენობის(600 სთ) ტოლი. დაკომპლექტებული ინდივიდუალური დატვირთვა არჩეული პროფესორისათვის თავსდება კართოთეკაში. ანლოგიური პროცესი გრძელდება სანამ სპეციალობის ყველა პროფესორისათვის არ იქნება შედგენილი ინდივიდუალური დატვირთვის უწყისი(იხ.ნახ.2.6.).



საქართველოს ტექნიკური უნივერსიტეტის 2008 /2009 სასწავლო წლის აკადემიური დატვირთვის განაწილება

კათედრის დასახელება და ნომერი —94—

პედაგოგის გვარი, სახელი, მამის სახელი, წოდება და სამტატო ერთეული	საგნის დასახელება	კურსი	სპეციალობა და ჯგუფები	ჯგ. N ქართული	ჯგ. N რუსული	ჯგუფში სტუდენტთა რაოდენობა	პირველი სემესტრი: ჯგუფთა რაოდენობა	ლექცია	პრაქტიკული და ლაბორატორიული მეცადინეობა	საკურსო პროექტი (სამუშაო)	გვიდეკ	რეფერატი საშინაო და საკურსო სამუშაოზე/ახსნა	საწარმო პრაქტიკა	დიპლომამდელი პრაქტიკა	მაგისტრატურა	სულ 1 სემესტრში	მეორე სემესტრი: ჯგუფთა რაოდენობა	ლექცია	პრაქტიკული და ლაბორატორიული მეცადინეობა	საკურსო პროექტი (სამუშაო)	გვიდეკ	რეფერატი საშინაო და საკურსო სამუშაოზე/ახსნა	სახელმძღვანელო ბიბლიო	საწარმო პრაქტიკა	დიპლომამდელი პრაქტიკა	მაგისტრატურა	სულ 11 სემესტრში	სულ საათების წელი	შენიშვნა
დიდმანიძე ვაჟა ალფეზის ძე	კომპიუტერის არქიტექტურა	3	იმსი (108635,108636)	ქ		20	15	15		60		4	6	4				89	15	15	30	60		4	6	4				119	532	
						20				60		4	6	4				74			30	60		4	6	4				104		
		3	იმსი (108639)	რ		10		15		30		4	6	3				58		15	30	30		4	6	3				88		
	მაგისტრანტის სელმძღვანელი															34	34										34	34	68	
ჯამი								30	0	150	0	12	18	11	0	0	34	255		30	90	150	0	12	18	11	0	0	0	34	345	600	

200 წ. ფაკულტეტის დეკანი

ნახ.2.6.

მიღებული ინდივიდუალური დატვირთვები წარმოადგენენ საფუძველს მონაცემთა დადგენისათვის ცხრილების შესადგენად. ამისათვის მოწმდება თითოეული პროფესორის ინდივიდუალური დატვირთვა მოცემული სემესტრისათვის (საგანი, ლექცია, პრაქტიკული, ლაბორატორიული, საკურსო სამუშაო-პროექტი). დადგენილი საათების რაოდენობა კვირაში გადაიტანება ცხრილის დადგენის ინდივიდუალურ მონაცემთა დოკუმენტში(იხ.ნახ2.7.).

კათედრა

ᲛᲝᲜᲐᲪᲔᲛᲔᲑᲘ

ცხრილის შედგენის შესახებ-----სემესტრი 20----

(გვარი, სახელი, მამის სახელი)

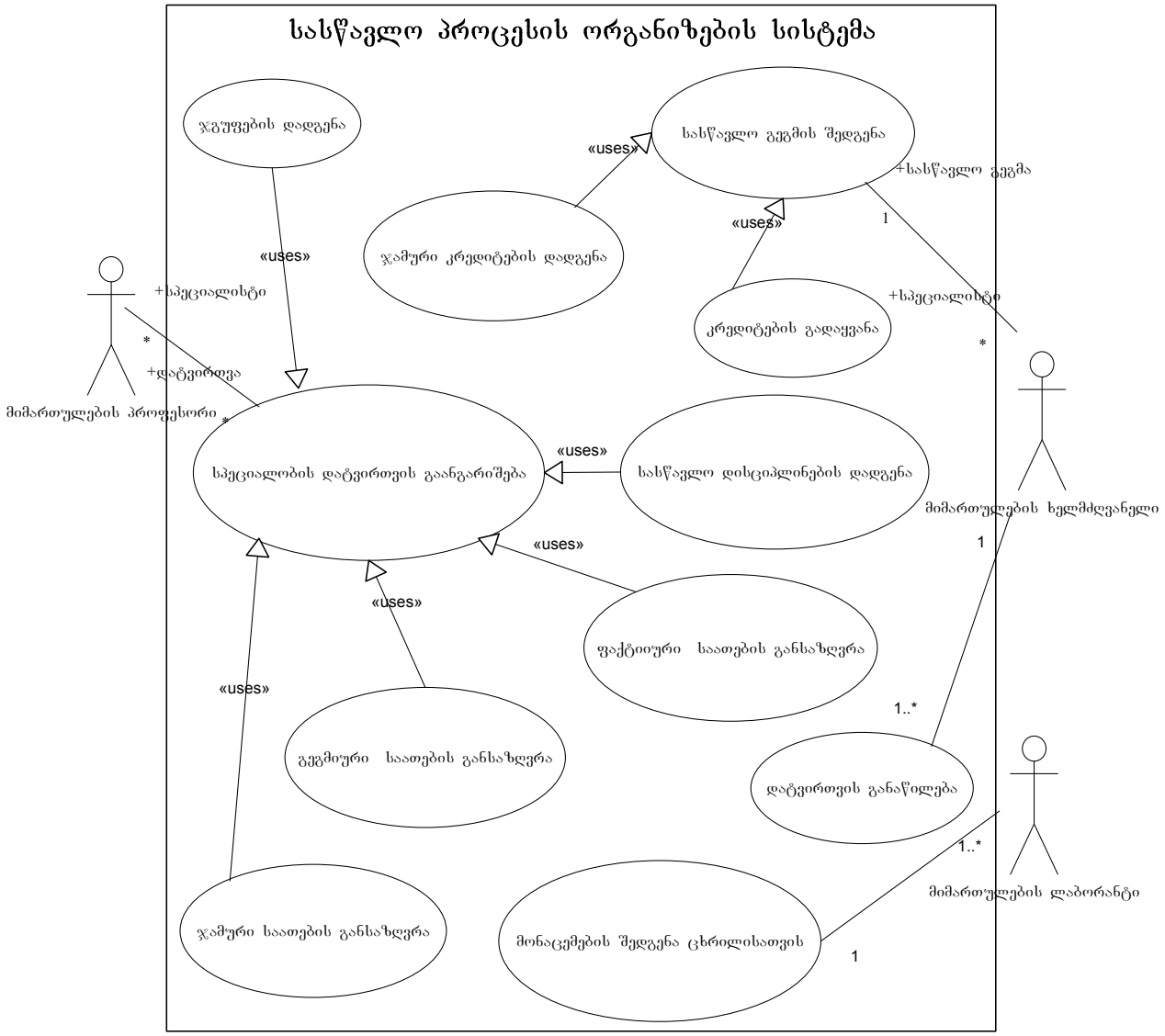
საგნის დასახელება	კურსი	აკადემიური ჯგუფის №	საათების რაოდენობა კვირაში					მიუთითეთ განსხვავებული მოთხოვნები ლექც. და ლაბ. მეცად. ჩატარების მიმართ	პედაგოგის დატვირთვა
			ლექცია	პრაქტიკული	ლაბორატორიული	საკურსო პროექტი	სახელოსნო		რომელ დღეებში და საათებში შეუძლია მეცადინეობის ჩატარება

ნახ.2.7.

მოყვანილი პროცესის ავტომატიზაციის მიზნით უნდა დადგინდეს კონტექსტი და მოთხოვნები სისტემისადმი. ამ მიზნით ვიყენებთ პრეცედენტების დიაგრამას, რომელიც მოყვანილია ნახ.2.8.-ზე.

სისტემის კონტექსტის მოდელირებისათვის თავიდან ახდენენ სისტემის გარემომცველი აქტიორების იდენტიფიცირებას. ამისათვის დგინდება ის ჯგუფები, რომლებისთვისაც სისტემის მონაწილეობაა საჭირო თავიანთი ამოცანების გადასაწყვეტად. რამდენადაც მოცემულ ამოცანაში განიხილება სასწავლო პროცესის ორგანიზაციული მხარე, რომელიც მოიცავს სასწავლო გეგმის შედგენას სპეციალობების მიხედვით, მათ საფუძველზე სპეციალობის დატვირთვის გაანგარიშებას და განაწილებას, სისტემის კონტექსტს შეადგენენ ამა თუ იმ სპეციალობის(მიმართულების) წარმომადგენლები(მიმირთულების ხელმძღვანელი, მიმართულების პროფესორი, ლაბორანტი), რომლებიც დიაგრამაზე გამოსახულია სისტემის შემომსაზღვრელი ხაზის გარეთ. შესაძლებელია მსგავსი აქტიორების ორგანიზება განზოგადებისა და სპეციალიზაციის მიმართებების გამოყენებით. გაგების გაადვილების მიზნით ყოველი აქტიორისათვის შესაძლებელია

სტერეოტიპის შემოტანა. ბოლოს, განსაზღვრავენ სისტემის პრეცედენტებთან კავშირის საშუალებებს (დიაგრამაზე აქტიორები ასოციაციური მიმართებით არიან დაკავშირებული შესაბამის პრეცედენტებთან).



ნახ.2.8.

სისტემისადმი მოთხოვნილებების მოდელირებისას პრეცედენტების დიაგრამას იყენებენ სისტემის სასურველი ქცევის სპეციფიცირებისათვის. მათი საშუალებით სისტემა განიხილება როგორც ‘‘შავი ყუთი‘‘. ჩანს ყველაფერი მის გარეთ, აკვირდებიან მის რეაქციას მოვლენაზე, მაგრამ არაფერი იციან მის შინაგან მოწყობაზე. დაადგენენ რა სისტემის კონტექსტს ანუ მოხდება გარემომცველი აქტიორების იდენტიფიცირება, ყოველი აქტიორისათვის განიხილება ქცევა,

რომელსაც ის ელოდება ან მოითხოვს სისტემისაგან. საერთო ვარიანტებს გამოყოფენ როგორც პრეცედენტებს. თუ რაიმე ქცევა გამოიყენება სხვების მიერ, ან დადგინდება ქცევა, რომელიც აფართოვებს მოვლენათა ძირითად ნაკადს, ქცევის ამ ვარიაციებს გამოყოფენ ახალ პრეცედენტში. ბოლოს, მოახდენენ პრეცედენტების დიაგრამაზე ამ პრეცედენტების, აქტიორებისა და მათ შორის მიმართებების მოდელირებას. შესაძლებელია დაემატოს პრეცედენტებს შენიშვნები, რომლებიც აღწერენ არაფუნქციონალურ მოთხოვნებს. მოცემული მაგალითისათვის მოთხოვნები სისტემისადმი გამოსახულია დიაგრამაზე პრეცედენტებით სისტემის შემომსაზღვრელი ხაზის შიგნით, რომლებსაც აქვთ შემდეგი შინაარსი:

- პრეცედენტი "სასწავლო გეგმის შედგენა" ითვალისწინებს გარკვეული სპეციალობის სტუდენტთათვის სასწავლო გეგმის ავტომატიზებულ (ადამიანი-კომპიუტერი) ფორმირებას დადგენილი ნორმატივების დაცვით(ფორმა «სასწავლო გეგმა» იხ.ნახ.2.4.ა,ბ). კერძოდ, უნდა გათვალისწინებული იქნას ზოგადსაგანმანათლებლო, არჩევით და სპეციალობის საგნებზე გათვალისწინებული შეზღუდვები კრედიტების რაოდენობაზე და სასწავლო სემესტრში(კურსზე) კრედიტების რაოდენობაზე დადგენილი ჯამური რაოდენობა.

მასში ჩართულია პრეცედენტი «ჯამური კრედიტების დადგენა», რომლითაც გამოითვლება თითოეულ არჩეულ საგანზე გათვალისწინებული კრედიტების რაოდენობა სემესტრებისა და კურსების მიხედვით, პრეცედენტი «კრედიტების გადაყვანა», რომლითაც დგინდება არჩეულ საგანზე(ლექცია, პრაქტიკული, ლაბორატორიული, პრაქტიკა, სემინარი, საკურსო სამუშაო(პროექტი) და სტუდენტთა დამოუკიდებელი სამუშაო) გათვალისწინებული საათები სემესტრში კვირეების მიხედვით(ერთი სემესტრი მოიცავს 15 სასწავლო კვირას).

- პრეცედენტი "სპეციალობის დატვირთვის გაანგარიშება" გულისხმობს სპეციალობის მოცემული სასწავლო გეგმის შესაბამისი აკადემიური დატვირთვის საათების გაანგარიშებას ჯამური მნიშვნელობებით კურსების

მიხედვით და მთლიანად სპეციალობისათვის (ფორმა «სპეციალობის დატვირთვა» იხ.ნახ.2.5.).

მასში ჩართულია პრეცედენტები «ჯგუფების დადგენა», «სასწავლო დისციპლინების დადგენა», «ფაქტიური საათების განსაზღვრა», «გეგმიური საათების განსაზღვრა», «ჯამური საათების განსაზღვრა».

- პრეცედენტი "დატვირთვის განაწილება" გულისხმობს ინდივიდუალური დატვირთვის უწყისების ფორმირებას და შენახვას ინფორმაციულ ბაზაში(ფორმა «ინდივიდუალური დატვირთვა» იხ.ნახ.2.6.).
- პრეცედენტი "მონაცემების შედგენა" გულისხმობს მონაცემების დადგენას თითოეული პროფესორისათვის მოცემულ სასწალო სემესტრში ცხრილის შესადგენად და მათ ბეჭდვას შესაბამისი ფორმის სახით(ფორმა «მონაცემები» იხ.ნახ.2.7.).

ამრიგად, პრეცედენტებით შესაძლებელია სისტემის ქცევის მოდელირება, რისთვისაც საჭიროა მოცემულ ელემენტთან ურთიერთქმედებაში მყოფი აქტიორების იდენტიფიცირება და ორგანიზება. დადგინდეს მათი საერთო და სპეციალიზებული როლები. ყოველი აქტიორისათვის განხილულ იქნას ელემენტებთან მისი ურთიერთქმედების ძირითადი და ალტერნატიული საშუალებები. გამოიყოს ისეთი ურთიერთქმედებები, რომლებიც ცვლიან ელემენტის მდგომარეობებს. გამოვლენილი ქცევის ორგანიზება მოხდეს პრეცედენტის სახით, ხოლო საერთო და განსაკუთრებული ქცევის გამოყოფისათვის გამოყენებულ იქნას ჩართვისა და გაფართოების მიმართებები.

§ 2.2. სამუშაო პროცესებისა და ოპერაციების მოდელირება

სისტემის კონტექსტის დადგენისა და აქტიორთა როლების განსაზღვრის შემდეგ უნდა დადგინდეს სისტემის სასურველი ქცევა მათი წარმოდგენით. აქ ძირითადი ყურადღება ფიქსირდება მოღვაწეობაზე აქტიორების თვალთახედვიდან, რომლებიც თანამშრომლობენ სისტემასთან. ამ პროცესების მოდელირებისათვის გამოიყენება მოღვაწეობის დიაგრამა.

§ 2.2.1. მოღვაწეობის დიაგრამა

მოღვაწეობის დიაგრამა გამოიყენება სისტემის ქცევის დინამიური ასპექტების მოდელირებისათვის. მისი მეშვეობით შესაძლებელია გამოთვლითი პროცესების მიმდევრობითი და პარალელური ბიჯების მოდელირება. მოღვაწეობის დიაგრამა ზოგადათ შესდგება:

- მოღვაწეობის და მოქმედების მდგომარეობებისაგან;
- გადასვლებისაგან;
- ობიექტებისაგან.

მოქმედების მდგომარეობა ისეთი მდგომარეობაა, რომლის შემდგომი დეკომპოზიცია შეუძლებელია. ეს ნიშნავს, რომ მათ შიგნით შეიძლება ხდებოდეს სხვადასხვა მოვლენები, მაგრამ მოქმედების მდგომარეობაში შესრულებადი სამუშაო არ შეიძლება შეწყვეტილ იქნას. ერთი მოქმედების მდგომარეობის ხანგრძლივობა იკავებს ძალიან მცირე დროს. მოქმედება შეიძლება მდგომარეობდეს სხვა ოპერაციის გამოძახებაში, რაიმე სიგნალის გაგზავნაში, ობიექტის შექმნაში ან მოსპობაში, გამოსახულების გამოთვლაში.

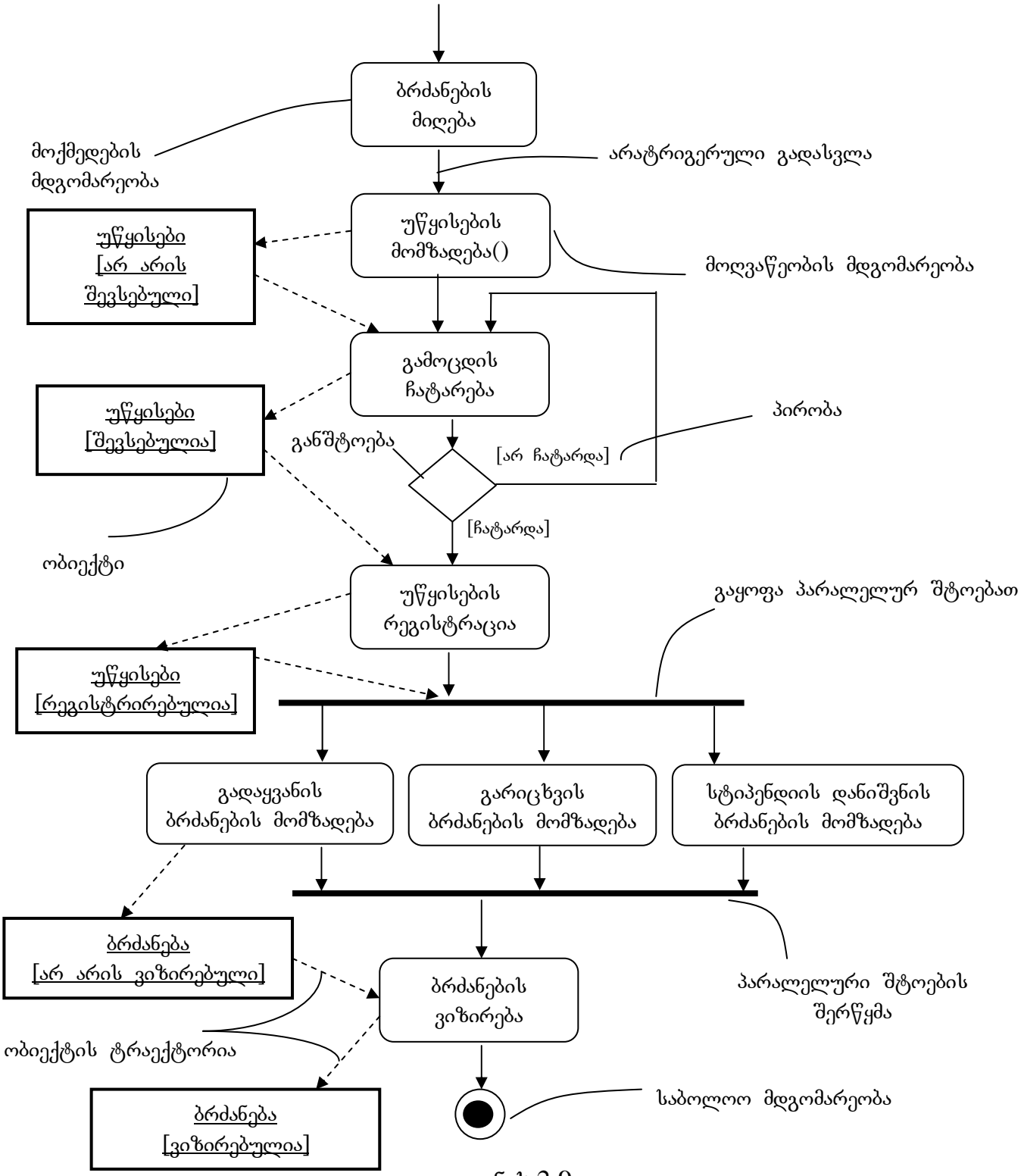
მისგან განსხვავებით შეიძლება ***მოღვაწეობის*** მდგომარეობების შემდგომი დეკომპოზიცია, ამის შედეგად შესრულებადი მოღვაწეობა შეიძლება წარმოვადგინოთ სხვა მოღვაწეობის დიაგრამების სახით. მოღვაწეობის მდგომარეობა არ არის ელემენტარული, ანუ შეიძლება შეწყვეტილ იქნას. ამასთან იგულისხმება, რომ მათი დამთავრებისათვის საჭიროა საკმაო დრო. შეიძლება ჩაითვალოს, რომ

მოქმედების მდგომარეობა – ეს მოღვაწეობის დიაგრამის კერძო სახეა, უფრო კონკრეტულად კი – ისეთი მდგომარეობა, რომლის შემდგომი დაშლა შეუძლებელია. რაც შეეხება მოღვაწეობის მდგომარეობას იგი წარმოდგენს შედგენილ მდგომარეობას, რომლის მართვის ნაკადი შეიცავს სხვა მოღვაწეობებისა და მოქმედებების მდგომარეობებს. გრაფიკულად მოღვაწეობისა და მოქმედების მდგომარეობები გამოისახებიან ერთნაირად, მხოლოდ იმის განსხვავებით, რომ პირველს შეიძლება გააჩნდეს დამატებითი მოქმედებები შესვლისას და გამოსვლისას ე.ი. მოქმედებები, რომლებიც სრულდებიან მოცემულ მდგომარეობაში შესვლისას და გამოსვლისას(იხ.ნახ.2.9.).

მდგომარეობაში შესვლის შედეგად სრულდება შესაბამისი მოქმედება ან მოღვაწეობა, ხოლო გამოსვლისას მართვა გადაეცემა შემდეგ მოქმედებას ან მოღვაწეობას. ასეთი გადაცემის აღწერისათვის გამოიყენება ***გადასვლები***, რომლებიც მიუთითებენ გზას ერთი მოქმედების მდგომარეობიდან მეორეში. გრაფიკულად იგი გამოისახება ისრიანი სწორი ხაზით. ასეთ გადასვლებს უწოდებენ გადასვლებს დასრულებისას ანუ არატრიგერულ გადასვლებს, რამდენადაც მართვა საწყის მდგომარეობაში მუშაობის დასრულებისას მყისიერად გადაეცემა შემდეგს.

ასეთი მარტივი მიმდევრობითი გადასვლები გვხვდება ყველაზე ხშირად, მაგრამ მხოლოდ ისინი არასაკმარისია მართვის ნაკადების მოდელირებისათვის. ისევე როგორც ბლოქ – სქემებში შესაძლებელია განშტოების გამოყენება, რომელიც აღწერს შესრულების სხვადასხვა გზებს გამომდინარე ბულის გამოსახულების რომელიღაც მნიშვნელობიდან. განშტოებას აქვს ერთი შესასვლელი და ორი ან მეტი გამოსასვლელი. დიაგრამაზე განშტოება აღინიშნება რომბით, ხოლო გამოსასვლელებზე მიეთითება პირობა. მოხერხებულობისათვის შეგვიძლია ვისარგებლოთ გასაღებური სიტყვით else, იმ გადასვლის აღსანიშნავათ, რომელიც უნდა აირჩეს იმ შემთხვევაში როდესაც პირობები ყველა დანარჩენი გადასვლებისათვის არ სრულდება.

საწყისი მდგომარეობა



ნახ.2.9.

განშტოების მეშვეობით შესაძლებელია იტერაციის რეალიზებაც. ამისათვის შემოაქვთ მოქმედების ორი მდგომარეობა. პირველში დგინდება მთვლელის საწყისი მნიშვნელობა, მეორეში იგი იზრდება, ხოლო განშტოებაში გამოთვლილი მნიშვნელობით დგინდება უნდა შეწყდეს იტერაცია თუ არა.

მარტივი და განშტოებადი გადასვლები მოღვაწეობის დიაგრამაზე გამოიყენება ყველაზე ხშირათ, მაგრამ გვხვდება აგრეთვე პარალელური ნაკადები. ასეთი პარალელური ნაკადების გაყოფისა და შერწყმის შესრულების აღნიშვნისათვის გამოიყენება სინქრონიზაციის ზღუდე, რომელიც იხაზება გამსხვილებული ვერტიკალური ან ჰორიზონტალური ხაზით. ყოველი პარალელურად შესრულებადი მართვის ნაკადი არსებობს დამოუკიდებელი აქტიური ობიექტის კონტექსტში, რომელიც მოდელირდება პროცესის სახით (იხ.§ 3.5.).

როგორც ნახაზიდან ჩანს გაყოფის წერტილი შეესაბამება ერთი მართვის ნაკადის გაყოფას სამ პარალელურზე. გაყოფის წერტილში შეიძლება არსებობდეს ერთი შემავალი და ორი ან მეტი გამომავალი მიმდევრობა. გამომავალი მიმდევრობა წარმოადგენს მართვის ერთ დამოუკიდებელ ნაკადს. გაყოფის წერტილის შემდეგ მოქმედებები სრულდებიან პარალელურად.

კონცეპტუალური თვალსაზრისით იგულისხმება ნამდვილი პარალელიზმი, მაშასადამე ერთდროული შესრულება, მაგრამ რეალურ სისტემებში ეს შეიძლება შესრულდეს (როდესაც სისტემა განლაგებულია რამოდენიმე კვანძზე) და შეიძლება არა (იმ შემთხვევაში როდესაც სისტემა განლაგებულია მხოლოდ ერთ კვანძზე). უკანასკნელ შემთხვევაში ადგილი აქვს მიმდევრობით შესრულებას ნაკადებს შორის გადართვით, რაც იძლევა პარალელიზმის მხოლოდ ილუზიას.

მოყვანილი ნახაზიდან აგრეთვე ჩანს, რომ შერწყმის წერტილი წარმოადგენს რამოდენიმე პარალელური ნაკადის სინქრონიზაციის მექანიზმს. ამ წერტილში შედიან ორი ან მეტი მიმდევრობა და გამოდის მხოლოდ ერთი. შერწყმის წერტილში პარალელური ნაკადები სინქრონიზირდებიან ანუ ყოველი მათგანი იცდის სანამ ყველა დანარჩენი მიაღწევს ამ წერტილს, რომლის შემდეგაც შესრულება გრძელდება ერთი ნაკადის ფარგლებში.

ბოლოს უნდა აღინიშნოს, რომ უნდა დაცული იქნას ბალანსი გაყოფის და შერწყმის წერტილებს შორის. ეს ნიშნავს, რომ ნაკადების რიცხვი, რომლებიც

გამოდიან გაყოფის წერტილიდან, ტოლი უნდა იყოს იმ ნაკადების რაოდენობის, რომლებიც შემოდის მის შესაბამის შერწყმის წერტილში.

მოღვაწეობის დიაგრამებზე შესაძლებელია ***ობიექტების*** გამოსახვაც. მართვის ნაკადის შინაარსიდან გამომდინარე მოღვაწეობის გარკვეული სახეობებით იქმნება მოცემული კლასის ობიექტ-ეგზემპლიარები(უწყისები, ბრძანებები), მაშინ როდესაც მოღვაწეობის სხვა სახეობები ახდენენ ამ ობიექტების მოდიფიცირებას. ობიექტები მოღვაწეობის დიაგრამაში ჩაირთვება დამოკიდებულების მიმართებით, რომლითაც ისინი უკავშირდებიან იმ მოღვაწეობას ან გადასვლას, სადაც იქმნებიან, მოდიფიცირდებიან ან ისპობიან. ობიექტისა და დამოკიდებულებების ასეთ მონაცვლეობას უწოდებენ ***ობიექტის ტრაექტორიას***, რამდენადაც აღწერს მის მონაწილეობას მართვის ნაკადში. ობიექტის ტრაექტორიის გარდა მოღვაწეობის დიაგრამაზე შეიძლება უჩვენოთ თუ როგორ იცვლება მისი როლი, მდგომარეობა და ატრიბუტების მნიშვნელობა. როგორც ნაჩვენებია ნახაზზე, ობიექტის მდგომარეობის გამოსახვისათვის მისი მნიშვნელობა თავსდება ფრჩხილებში და თავსდება ობიექტის სახელწოდებით.

ბიზნეს - პროცესების მოდელირებისას ხშირად მიმართავენ დიაგრამის დაყოფას ჯგუფებად, რომელთაგანაც თითოეული წარმოადგენს ამა თუ იმ განყოფილების ან თანამდებობის პირის დასახელებას, რომელიც პასუხს აგებს მოცემულ სამუშაოთა შესრულებაზე. ასეთ ჯგუფებს უწოდებენ ბილიკებს. ყოველ ბილიკს ენიჭება სახელი და გამოსახავს გარკვეულ პასუხიმგებლობას მასში მოთავსებულ სამუშაოებზე. აქტიურობის დიაგრამაზე, რომელიც დაყოფილია ბილიკებათ, ყოველი მოღვაწეობა მიეკუთვნება მხოლოდ ერთ ბილიკს, მაგრამ გადასვლებმა შეიძლება გადაკვეთონ ბილიკის საზღვრები.

§ 2.2.2. მოღვაწეობის დიაგრამების გამოყენება მართვის ნაკადების წარმოსადგენად

მოღვაწეობის დიაგრამა შეიძლება დაუკავშიროთ მოდელის ნებისმიერ ელემენტს მისი ქცევის ვიზუალირებისათვის, კერძოდ ისინი განიხილებიან მთლიანად სისტემის, პრეცედენტების, ოპერაციის ან კლასის კონტექსტში. სისტემის დინამიური ასპექტების მოდელირებისას მოღვაწეობის დიაგრამები გამოიყენებიან ძირითადათ ორი სახით: სამუშაო პროცესების მოდელირებისათვის, ოპერაციების მოდელირებისათვის.

როგორც აღნიშნული იყო პროგრამული სისტემები არ არსებობენ იზოლირებულად. ყოველთვის არის გარკვეული კონტექსტი, რომლის ფარგლებშიც ფუნქციონირებს სისტემა, ამასთან იგი ყოველთვის მოიცავს აქტიორებს, რომლებიც ურთიერთქმედებენ სისტემასთან. ნებისმიერი საწარმოს მასშტაბის ბიზნესისათვის განკუთვნილი პროგრამული უზრუნველყოფის განხილვიდან ჩანს, რომ ავტომატიზირებული სისტემა მუშაობს უფრო მაღალი დონის ბიზნეს-პროცესების კონტექსტში. ასეთი ბიზნეს-პროცესები წარმოადგენენ სამუშაო პროცესების მაგალითებს, რამდენადაც აღწერენ, როგორ ფუნქციონირებს საწარმო და რომელი ობიექტები არიან მათში ჩართული.

სამუშაო პროცესების მოდელირებისას ყურადღება ფოკუსირდება მოღვაწეობაზე აქტიორთა თვალთახედვიდან, რომლებიც თანამშრომლობენ სისტემასთან. ისინი გამოიყენებიან ბიზნეს-პროცესების ვიზუალირებისათვის, რომლებიც შეადგენენ დასამუშავებელი სისტემის არსს. მოღვაწეობის დიაგრამების ასეთი გამოყენებისას ობიექტთა ტრაექტორიის მოდელირებას აქვს განსაკუთრებით დიდი მნიშვნელობა.

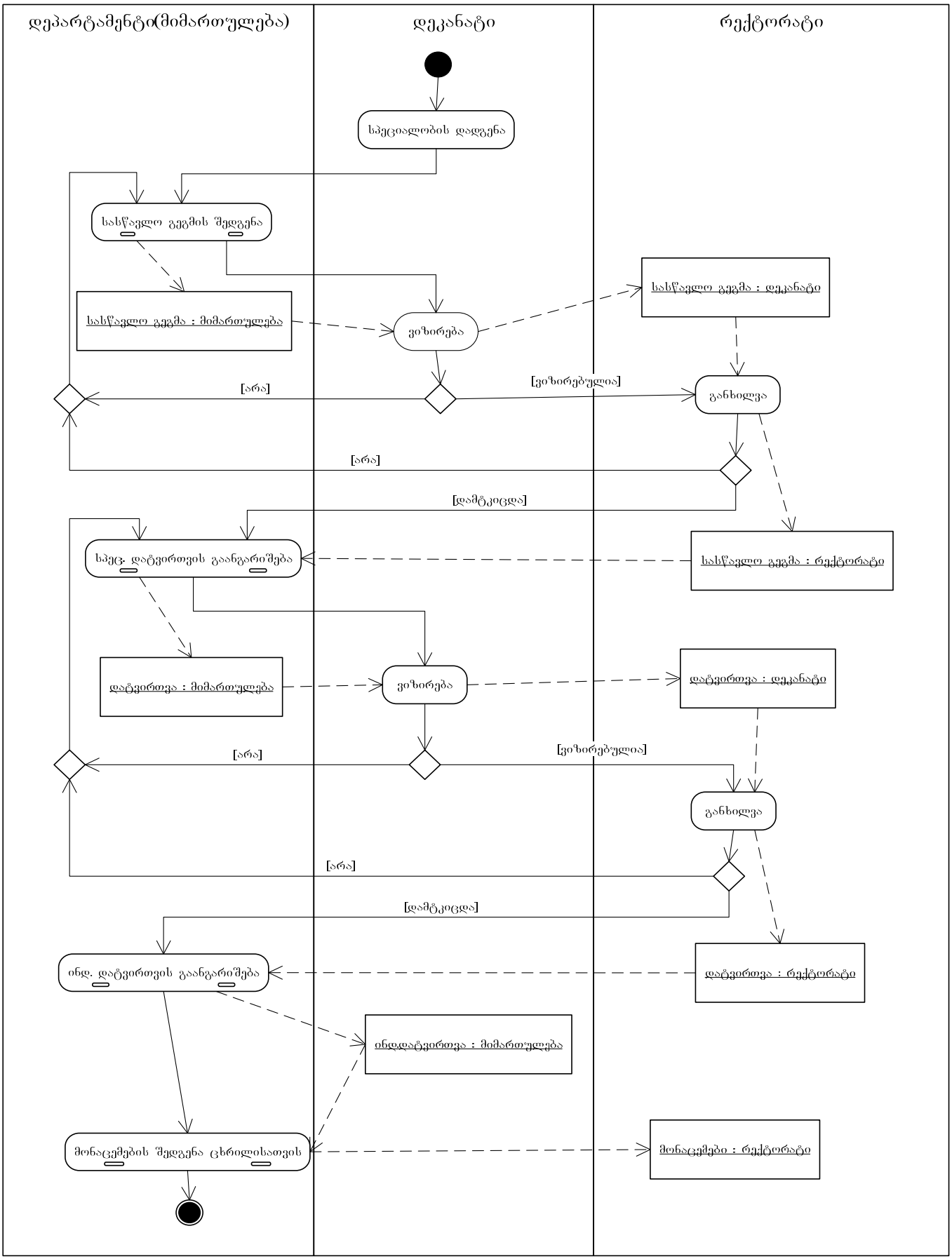
სამუშაო პროცესის მოდელის ასაგებად გამოიყოფა სამუშაო პროცესის რომელიმე ნაწილი. რთული სისტემების პროექტირებისას, შეუძლებელია გამოისახოს ყველა საინტერესო თანმიმდევრობა ერთ დიაგრამაზე. აირჩევა ობიექტები, რომლებზედაც დაკისრებულია პასუხისმგებლობა სამუშაო პროცესის ამა თუ იმ ნაწილზე. ყოველი ასეთი ობიექტისათვის შეიქმნება ბილიკები. პროცესის საზღვრების მოდელირებისათვის ახდენენ სამუშაო პროცესების საწყისი და საბოლოო მდგომარეობების პირობების იდენტიფიცირებას. საწყისი მდგომარეობიდან დაწყებული აღიწერება მოღვაწეობა და მოქმედება, რომლებიც სრულდება დროის

სხვადასხვა მომენტში და გამოისახება ისინი დიაგრამაზე მოღვაწეობის ან მოქმედების მდგომარეობის სახით.

რთული მოქმედებები გამოიყოფა მოღვაწეობის მდგომარეობაში და ყოველი ასეთი მდგომარეობისათვის უნდა შედგეს მოღვაწეობის ცალკე დიაგრამა. მიუთითებენ გადასვლებს, რომლებიც აკავშირებენ ამ მოღვაწეობებისა და მოქმედებების მდგომარეობებს. თავიდან ყურადღება უნდა გამახვილდეს მიმდევრობით ნაკადებზე, ხოლო შემდეგ გადადიან განშტოებებზე და ბოლოს განიხილება გაყოფა და შერწყმა.

მაგალითისათვის ნახ.2.10.-ზე მოყვანილია მოღვაწეობის დიაგრამა, რომელიც ასახავს წინა პარაგრაფში აღწერილ სამუშაო პროცესს - სასწავლო პროცესის ორგანიზებას უმაღლეს სასწავლებელში. მოყვანილი ტექსტიდან დგინდება, რომ უმაღლეს სასწავლებელში სასწავლო პროცესის ორგანიზებაზე პასუხისმგებელია რექტორატი, დეკანატი და დეპარტამენტი (მიმართულება). დიაგრამაზე თითოეულს შეესაბამება ბილიკი, რომელშიც ასახულია მათ მიერ წარმოებული მოქმედებები.
ეს მოქმედებები დიაგრამაზე გამოსახულია როგორც მოქმედების, ისე მოღვაწეობის (სასწავლო გეგმის შედგენა, სპეციალობის დატვირთვის გაანგარიშება, ინდივიდუალური დატვირთვის შედგენა, მონაცემების შედგენა ცხრილისათვის) მდგომარეობებით. დიაგრამაზე ნაჩვენებია აგრეთვე ობიექტები(სასწავლო გეგმა, დატვირთვა, ინდივიდუალური დატვირთვა, მონაცემები), რომლებიც ფორმირდება ამა თუ იმ მოქმედების შედეგად და მათი ტრაექტორიები. ერთი მოქმედებიდან მეორეზე გადასვლა ხორციელდება როგორც პირდაპირი(არატრიგერული), ასევე პირობითი გადასვლებით.

მოღვაწეობის მდგომარეობები თავის მხრივ წარმოიდგინებიან მათი მოქმედების აღმწერი მოღვაწეობის დიაგრამებით, რომლებიც მოყვანილია ქვევით. მათ შორის არის ოპერაციებიც, რომლითაც გამოისახება ამა თუ იმ მაჩვენებლის გამოთვლის პროცესი.


დეპარტამენტი(მიმართულება)
დეკანატი
რექტორატი
სპეციალობის დადგენა
სასწავლო გეგმის შედგენა
სასწავლო გეგმა : მიმართულება
ვიზირება
სასწავლო გეგმა : დეკანატი
[არა]
[ვიზირებულია]
განხილვა
[არა]
[დამტკიცდა]
სპეც. დატვირთვის გაანგარიშება
სასწავლო გეგმა : რექტორატი
დატვირთვა : მიმართულება
ვიზირება
დატვირთვა : დეკანატი
[არა]
[ვიზირებულია]
განხილვა
[არა]
[დამტკიცდა]
ინდ. დატვირთვის გაანგარიშება
დატვირთვა : რექტორატი
ინდდატვირთვა : მიმართულება
მონაცემების შედგენა ცხრილისათვის
მონაცემები : რექტორატი

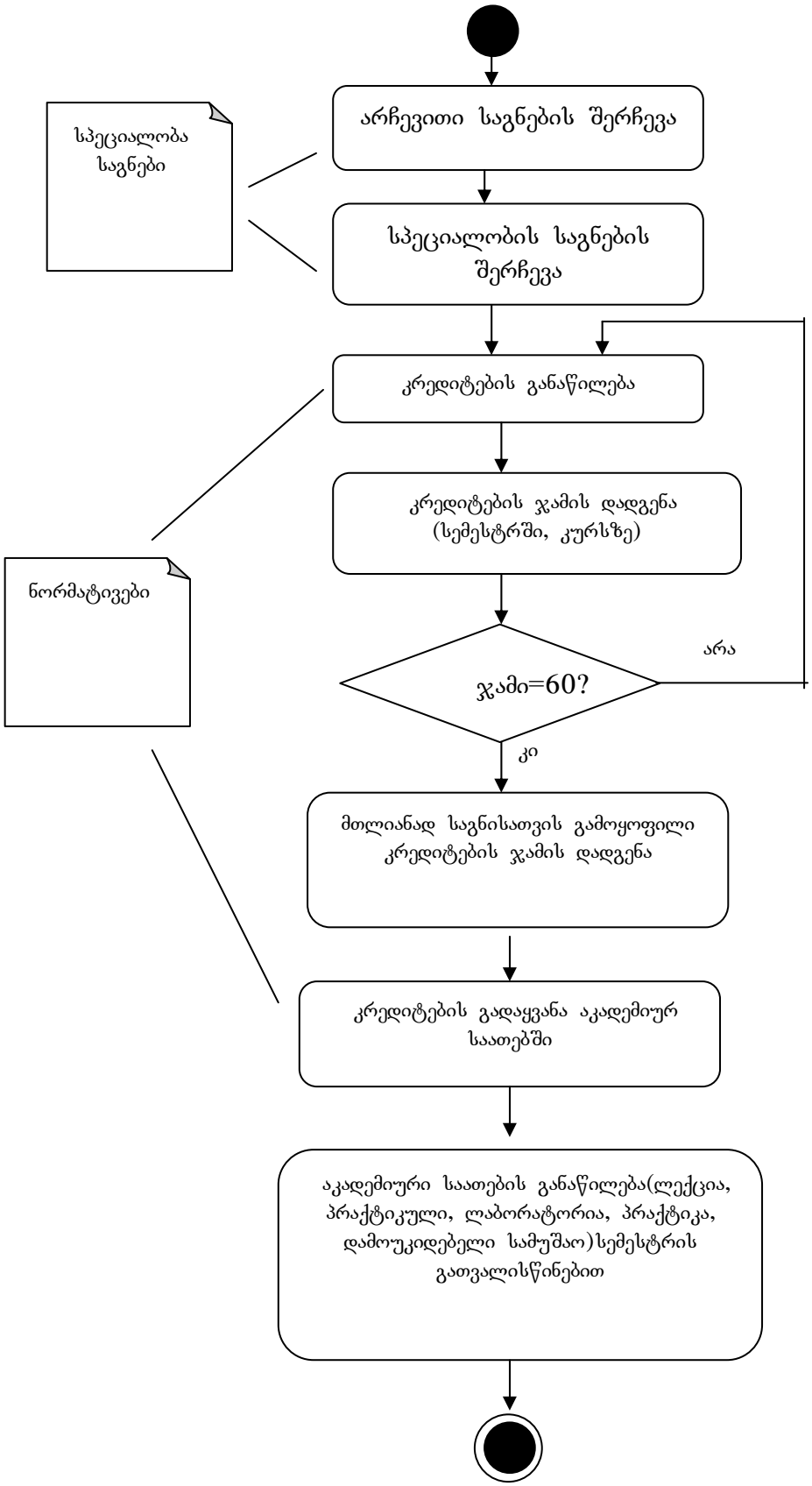
ნახ.2.10.

ყველაზე ხშირად მოღვაწეობის დიაგრამებს უკავშირებენ ოპერაციებს(იხ. §3.1). ამ შემთხვევაში მოღვაწეობის დიაგრამები გამოიყენებიან როგორც ბლოქ-სქემები გამოთვლის დეტალების მოდელირებისათვის.

ასეთი გამოყენებისას მოღვაწეობის დიაგრამა ხდება შესასრულებელი მოქმედებების ბლოქ – სქემა. ოპერაციების მოდელირებისათვის გამოვლინდება ის აბსტრაქციები, რომლებიც განეკუთვნებიან ოპერაციას, მას განეკუთვნება ოპერაციის პარამეტრებიც. მოახდენენ ოპერაციის საწყისი და საბოლოო მდგომარეობის პირობების იდენტიფიცირებას. დაწყებული ოპერაციის საწყისი მდგომარეობიდან აღიწერება მოღვაწეობა და მოქმედება, რომლებიც სრულდება დროის სხვადასხვა მომენტში, ამის შემდეგ გამოისახება ისინი დიაგრამაზე მოღვაწეობის ან მოქმედების მდგომარეობის სახით. აუცილებლობის შემთხვევაში გამოიყენება განშტოების წერტილები პირობითი გადასვლების და იტერაციის აღწერისათვის.

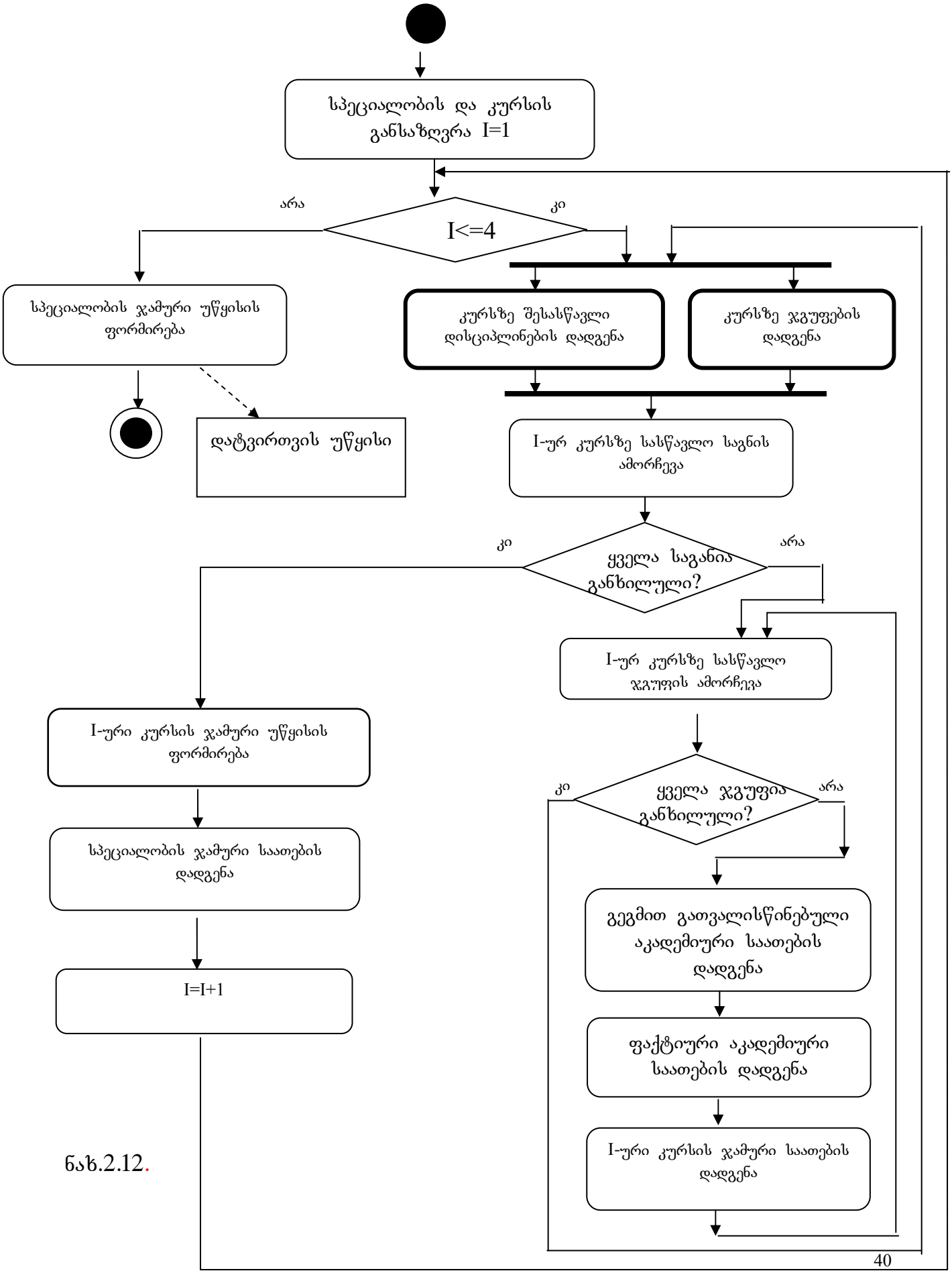
მხოლოდ იმ შემთხვევაში თუ ოპერაციის მფლობელი აქტიური კლასია(იხ. §3.7) და ამის აუცილებლობა წარმოიშობა, გამოიყენება გაყოფისა და შერწყმის ოპერაციები შესრულების პარალელური ნაკადების შესრულებისათვის.

სასწავლო გეგმის შედგენა



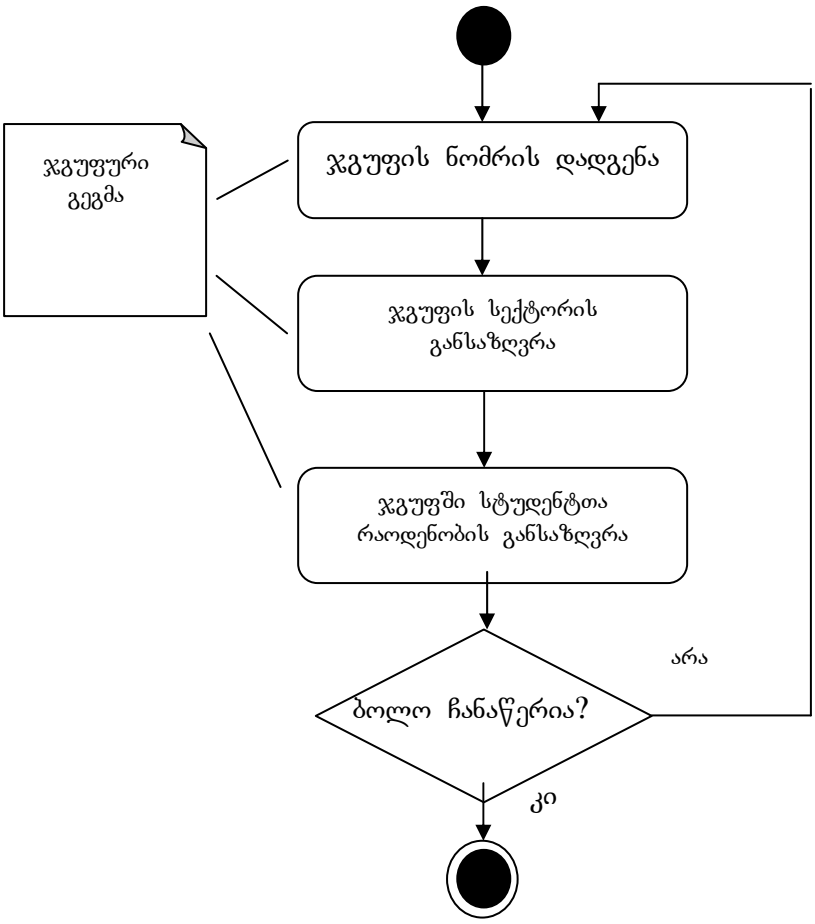
ნახ.2.11.

სპეციალობის დატვირთვის გაანგარიშება



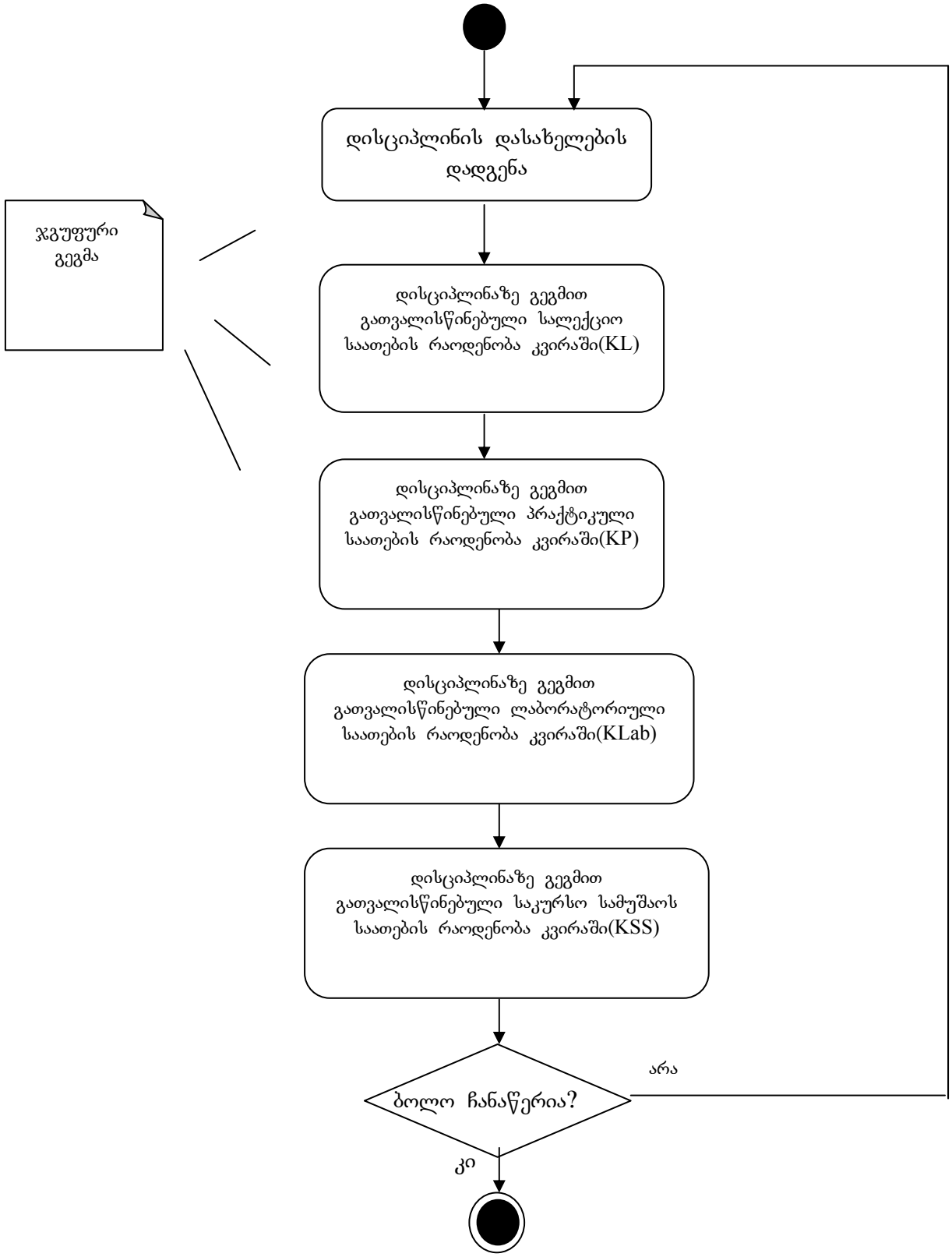
ნახ.2.12.

კურსზე ჯგუფების დადგენა



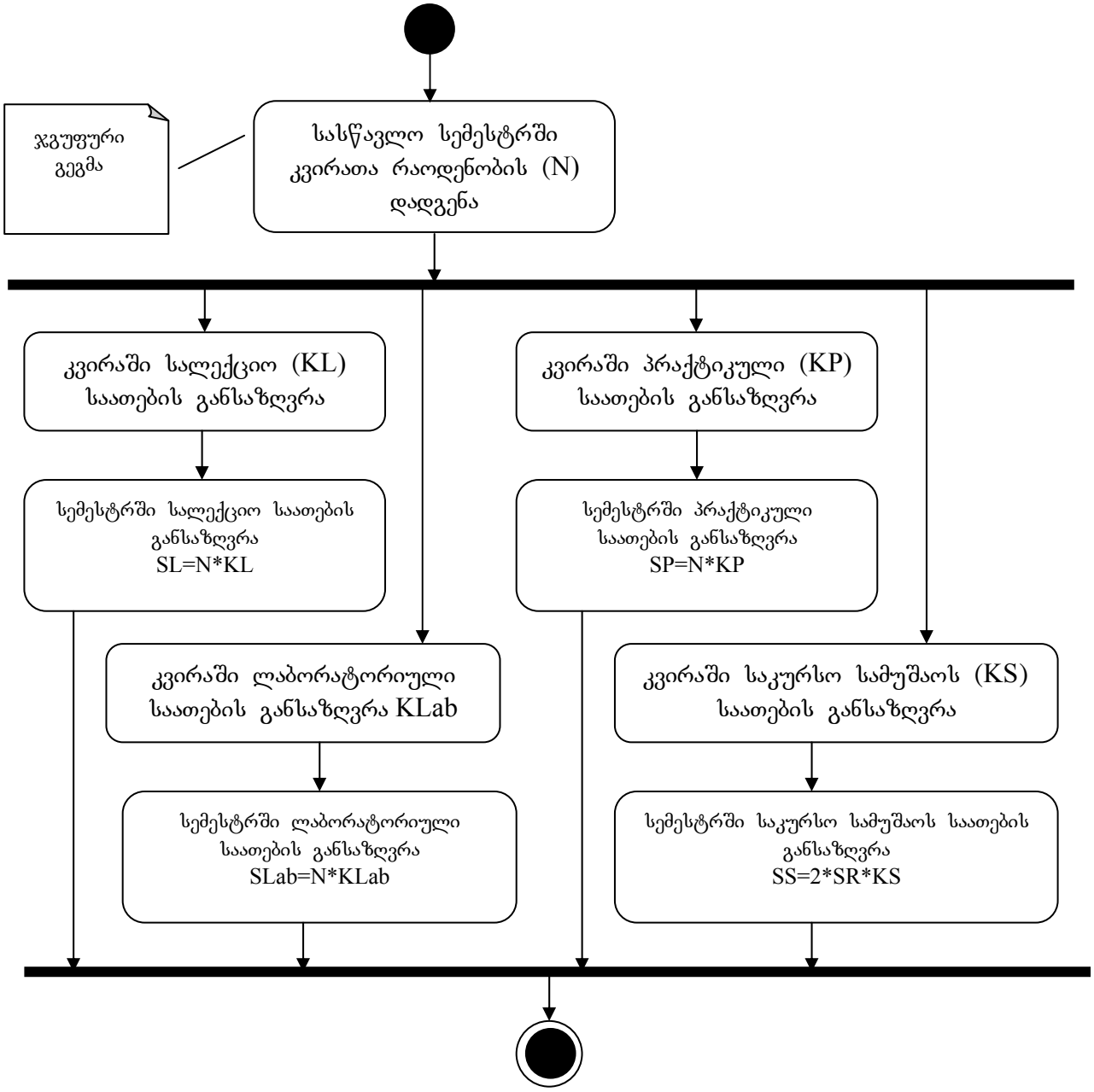
ნახ.2.13.

კურსზე შესასწავლი დისციპლინების დადგენა



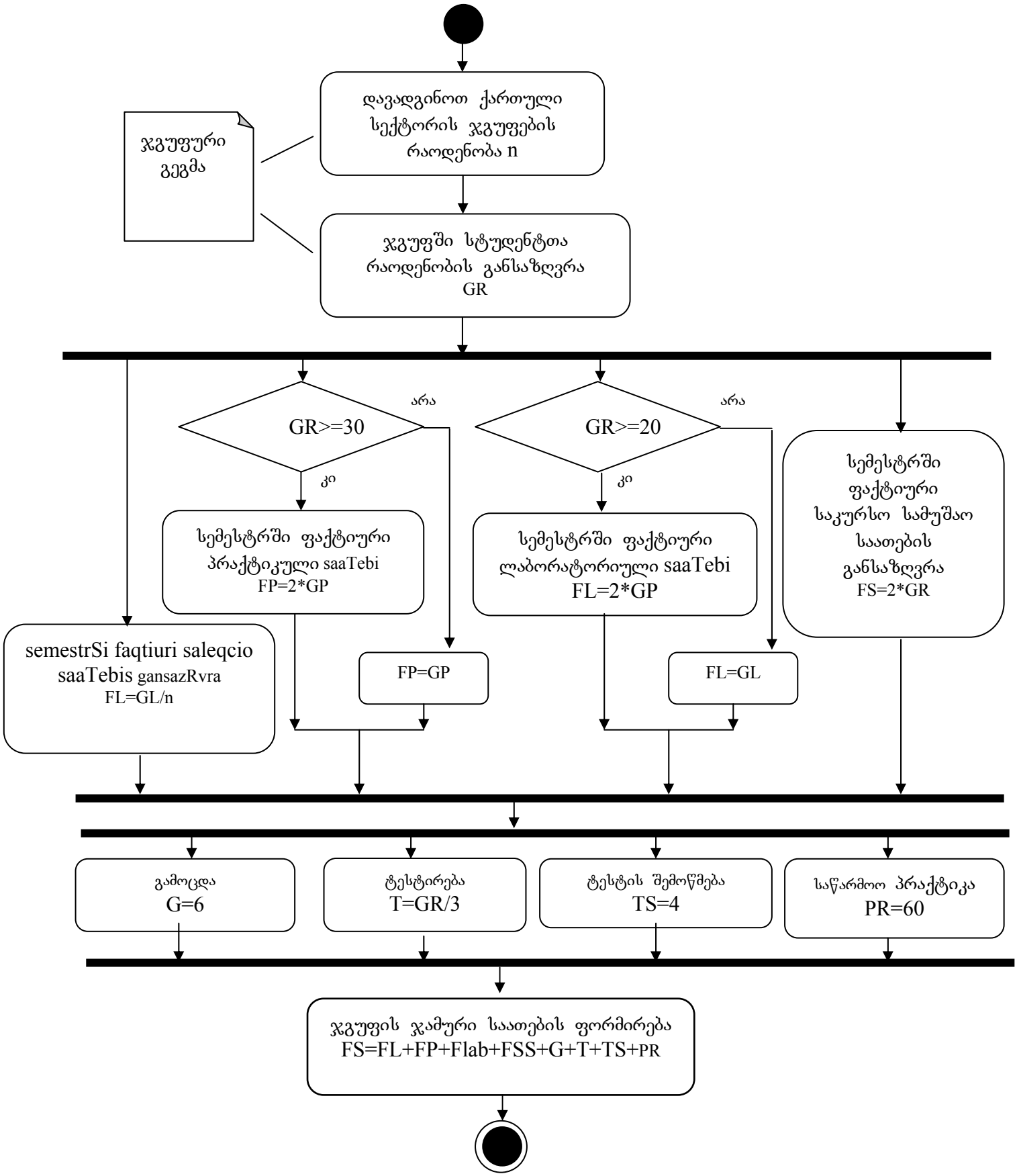
ნახ.2.14.

გეგმით გათვალისწინებული აკადემიური საათების დადგენა

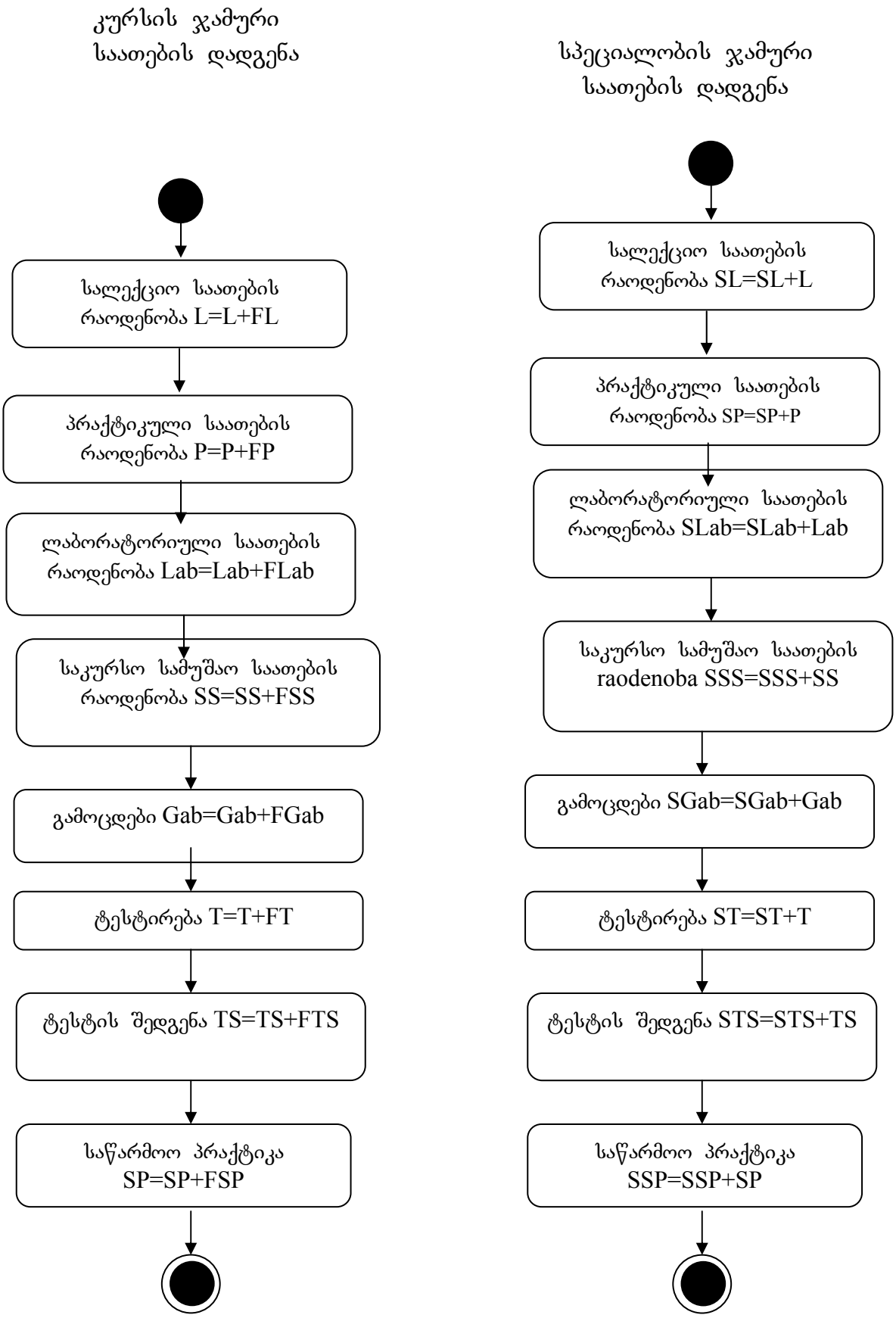


ნახ.2.15.

ფაქტიური აკადემიური საათების დადგენა

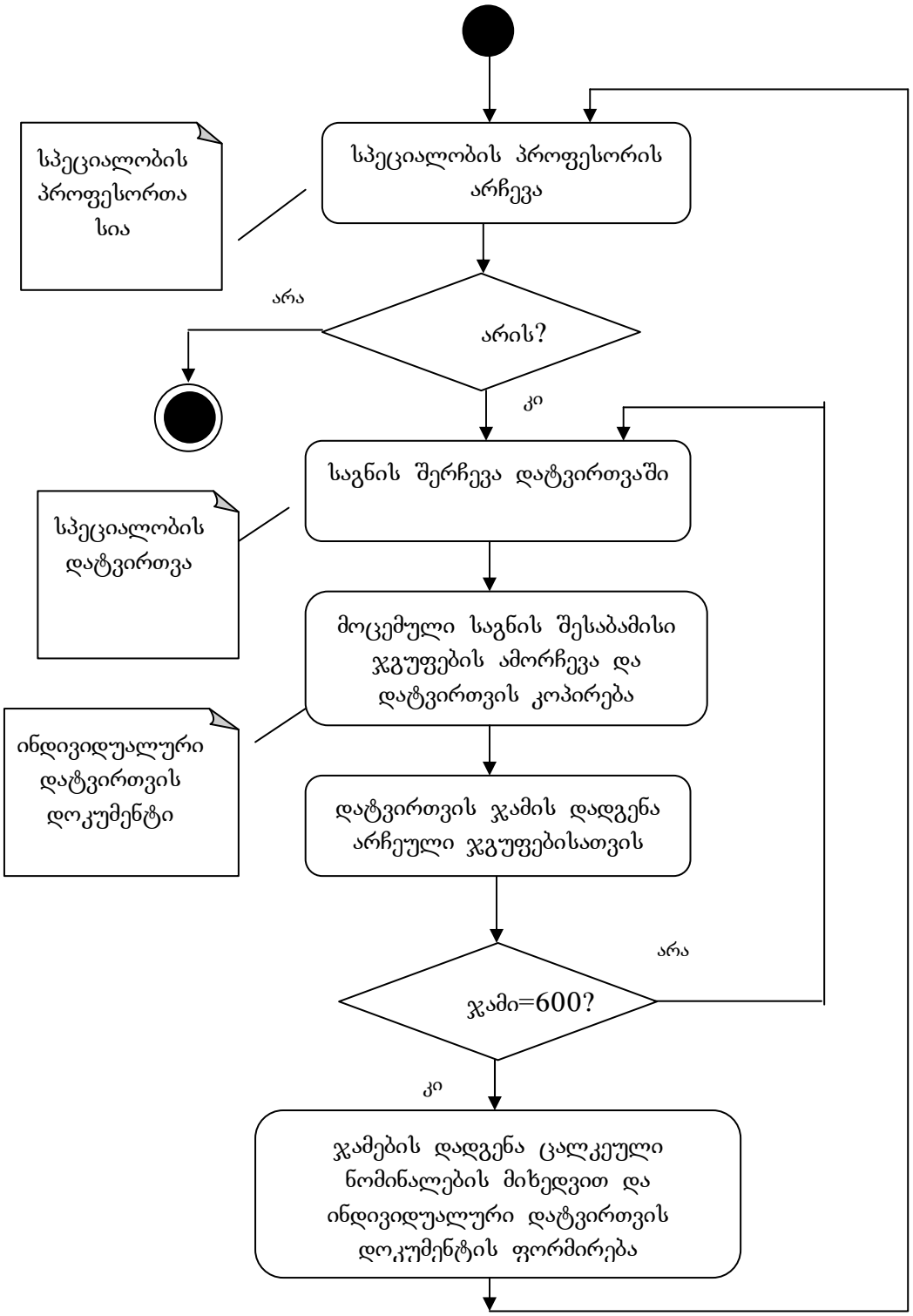


ნახ.2.16.


კურსის ჯამური
საათების დადგენა
სალექციო საათების
რაოდენობა L=L+FL
პრაქტიკული საათების
რაოდენობა P=P+FP
ლაბორატორიული საათების
რაოდენობა Lab=Lab+FLab
საკურსო სამუშაო საათების
რაოდენობა SS=SS+FSS
გამოცდები Gab=Gab+FGab
ტესტირება T=T+FT
ტესტის შედგენა TS=TS+FTS
საწარმოო პრაქტიკა
SP=SP+FSP
სპეციალობის ჯამური
საათების დადგენა
სალექციო საათების
რაოდენობა SL=SL+L
პრაქტიკული საათების
რაოდენობა SP=SP+P
ლაბორატორიული საათების
რაოდენობა SLab=SLab+Lab
საკურსო სამუშაო საათების
raodenoba SSS=SSS+SS
გამოცდები SGab=SGab+Gab
ტესტირება ST=ST+T
ტესტის შედგენა STS=STS+TS
საწარმოო პრაქტიკა
SSP=SSP+SP

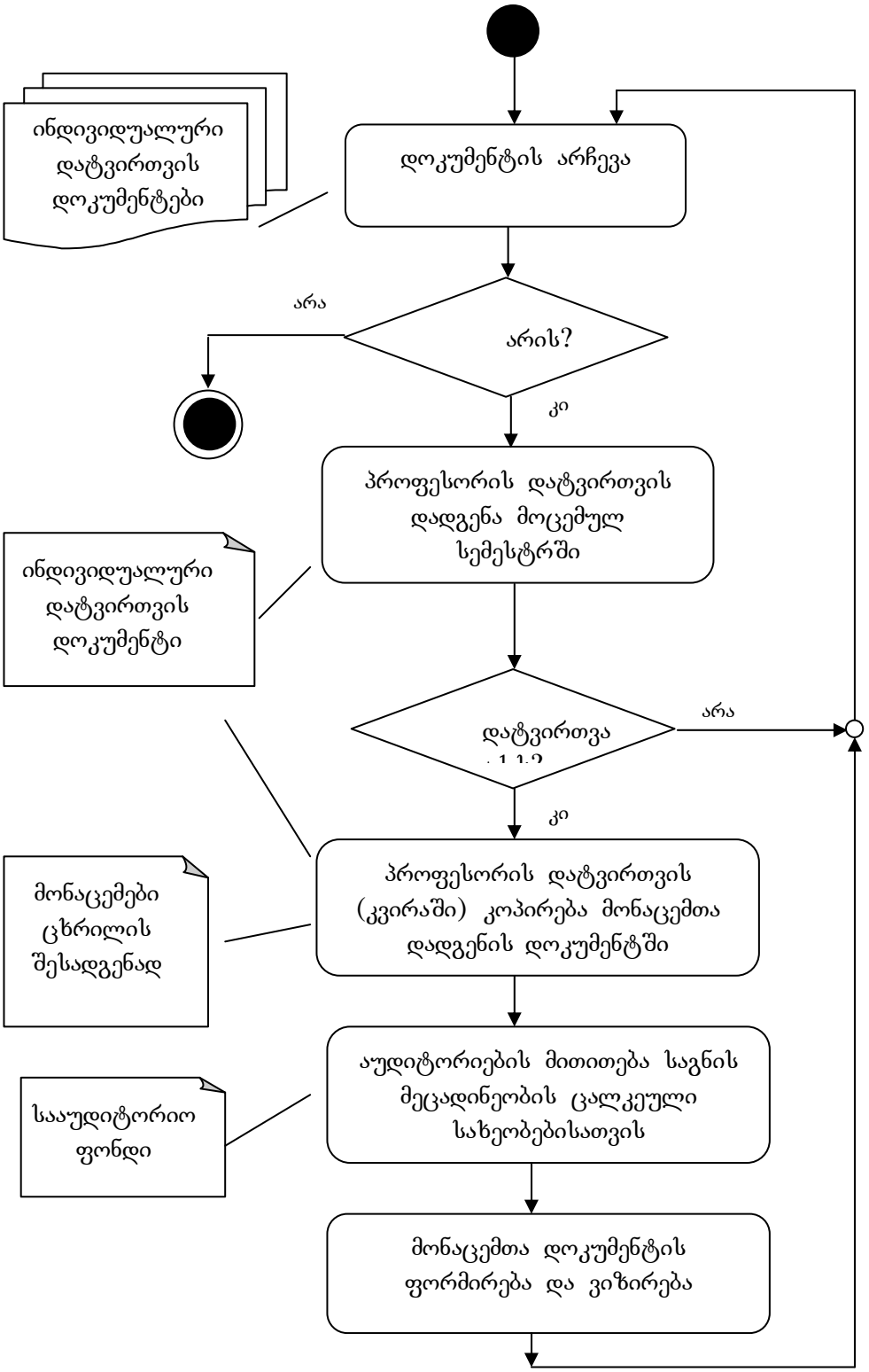
ნახ.2.17.

დატვირთვის განაწილება



ნახ.2.18.

მონაცემები ცხრილის შესადგენად



ნახ.2.19.

§ 2.3. მართვის ნაკადების მოდელირება ობიექტებს შორის ურთიერთქმედებით

ანალიზის წინა ეტაპზე მოხდა სისტემის სასურველი ქცევის წარმოდგენა აქტიორების თვალთახედვიდან, რომლებიც თანამშრომლობენ სისტემასთან. ამ სამუშაო პროცესებისა და ოპერაციების მოდელირებისათვის გამოყენებული იქნა მოღვაწეობის დიაგრამები. ამ უკანასკნელით მართვის ნაკადი წარმოიდგინება მოქმედებიდან მოქმედებამდე. მაგრამ, როგორც აღნიშნული იყო ობიექტ-ორიენტირებულ სისტემებში ძირითად სამშენებლო ბლოკს წარმოადგენს ობიექტი(კლასი), ამიტომ მნიშვნელოვანია იგივე მართვის ნაკადის წარმოდგენა ობიექტებს შორის ურთიერთქმედებით.

ურთიერთქმედებას უწოდებენ ქცევას, რომელიც გამოიხატება მოცემულ კონტექსტში მოცემული ერთობლიობის ობიექტებს შორის შეტყობინებების გაცვლაში, რის შედეგადაც მიიღწევა განსაზღვრული მიზანი. ურთიერთქმედებას ადგილი აქვს ყოველთვის, როდესაც ობიექტები დაკავშირებულია ერთმანეთთან.

ურთიერთქმედებაში მონაწილე ობიექტები შეიძლება იყვნენ კონკრეტული არსები ან პროტოტიპები. კონკრეტული არსების სახით ობიექტი წარმოადგენს რაიმეს, რეალურ სამყაროში არსებულს. მაგალითად p, კლასი ***ადამიანის*** ეგზემპლიარი, შესაძლებელია აღნიშნავდეს კონკრეტულ ადამიანს. პირიქით, პროტოტიპმა p შესაძლებელია წარმოადგინოს კლასი ***ადამიანის*** ნებისმიერი ეგზემპლიარი.

ობიექტებს შორის კავშირი წარმოადგენს მათ სემანტიკურ შეერთებას. ზოგადათ კავშირი წარმოადგენს ასოციაციის ეგზემპლიარს(იხ.§3.2.). თუ ორი ობიექტი(კლასი) შედიან ასოციაციაში, მაშინ მათ ეგზემპლიარებს შორის შესაძლებელია კავშირის არსებობა. ხოლო, თუ ორ ობიექტს შორის არსებობს კავშირი, მაშინ ერთ ერთს შეუძლია გაუგზავნოს შეტყობინება მეორეს. კავშირი

განსაზღვრავს გზას, რომლითაც ერთი ობიექტი გადასცემს შეტყობინებას მეორეს(ან თავის თავს).

შეტყობინება – ობიექტებს შორის მონაცემთა გაცვლაა, რომლის დროსაც გადაიცემა გარკვეული ინფორმაცია იმის გათვალისწინებით, რომ პასუხად განხორციელდება გარკვეული მოქმედება. ყველაზე ხშირად შეტყობინება დაიყვანება ოპერაციის გამოძახებამდე ან სიგნალის გაგზავნამდე, ამავე დროს მას შეუძლია შექმნას ან მოსპოს სხვა ობიექტები.

მოქმედება, რომელიც წარმოადგენს შეტყობინების მიღების შედეგს შესრულებადი წინადადებაა, რომელიც ქმნის გამოთვლითი პროცედურის აბსტრაქციას. მოქმედებას შეუძლია გამოიწვიოს მდგომარეობის შეცვლა.

ურთიერთქმედების აგებისას შესაძლებელია რამდენიმე სახის მოქმედების მოდელირება:

call(გამოძახება) – იძახებს ოპერაციას, რომელიც გამოიყენება ობიექტზე. ობიექტს შეუძლია გაუგზავნოს შეტყობინება თავისთავს, რაც გულისხმობს ოპერაციის ლოკალურ გამოძახებას.

return(დაბრუნება) – აბრუნებს მნიშვნელობას გამომძახებელ ობიექტთან.

send(გაგზავა) – აგზავნის სიგნალს ობიექტთან.

create(შექმნა) – ქმნის ახალ ობიექტს.

destroy(განადგურება) – ანადგურებს ობიექტს. ობიექტს შეუძლია გაანადგუროს თავისი თავი.

როდესაც ობიექტი იძახებს ოპერაციას ან უგზავნის სიგნალს მეორე ობიექტს, შეტყობინებასთან ერთად შეიძლება გადავცეთ მისი ფაქტიური პარამეტრები. ასევე, როდესაც ობიექტი უბრუნებს მართვას მეორე ობიექტს, შესაძლებელია მივუთითოთ დასაბრუნებელი მნიშვნელობა.

როდესაც ობიექტი უგზავნის შეტყობინებას მეორე ობიექტს, მიმღებს შეუძლია თავის მხრივ გაუგზავნოს შეტყობინება მესამე ობიექტს, ამ უკანასკნელმა მეოთხეს და ა.შ. შეტყობინებათა ასეთი ნაკადი გვაძლევს მიმდევრობას.

ურთიერთქმედებაში მონაწილე ობიექტები არსებობს მთელი ურთიერთქმედების მანძილზე. მაგრამ ზოგჯერ ობიექტები საჭიროა შეიქმნას (**create**) და განადგურდეს (**destroy**). ეს ეხება კავშირებსაც: მიმართებები ობიექტებს შორის შეიძლება აღიძვრას ან გაქრეს. იმისათვის, რომ აღინიშნოს ობიექტების ან კავშირების ურთიერთქმედების პროცესში წარმოშობისა და გაქრობის ფაქტი, ელემენტს უერთებენ ერთერთ შემდეგ შეზღუდვას:

- **new**(ახალი) – გვიჩვენებს, რომ ეგზემპლარი ან კავშირი წარმოიქმნება მოცემული ურთიერთქმედების შესრულების დროს;
- **destroyed**(განადგურებული) – ეგზემპლიარი ან კავშირი ნადგურდება მოცემული ურთიერთქმედების შესრულების დასრულებამდე;
- **transient**(დროებითი) – ეგზემპლიარი ან კავშირი იქმნება მოცემული ურთიერთქმედების შესრულებისას და ნადგურდება მის დამთავრებამდე.

ურთიერთქმედების საშუალებით შესაძლებელია მართვის ნაკადების მოდელირება პრეცედენტების ან მთლიანად სისტემის შიგნით. პრეცედენტების კონტექსტში ურთიერთქმედებით აღიწერება სცენარი, რომელიც თავის მხრივ წარმოადგენს პრეცედენტის მოქმედების ერთ-ერთ ნაკადს. ანალიზის მოცემულ ეტაპზე უნდა მოხდეს მოთხოვნათა რეალიზების მექანიზმების მოდელირება კონკრეტული სტრუქტურებისა და მათ შორის ურთიერთქმედების სახით, რაც გულისხმობს პრეცედენტით გათვალისწინებული მართვის ნაკადების მოდელირებას.

ურთიერთქმედების საშუალებით ფაქტობრივად აღიწერება მოქმედებათა თანამიმდევრობა, რომელსაც ასრულებენ ობიექტები. ურთიერთქმედება გრაფიკულად წარმოიდგინება ურთიერთქმედების დიაგრამის სახით.

ურთიერთქმედების დიაგრამით მართვის ნაკადის მოდელირებისას პირველ რიგში განისაზღვრება ურთიერთქმედების კონტექსტი და აღიწერება სცენარი, რომელშიც განხორციელდება ურთიერთქმედება. ამისათვის ხდება ობიექტთა იდენტიფიცირება, რომლებიც გარკვეულ როლს ასრულებენ და დადგინდება მათი

საწყისი თვისებები, მათ შორის ატრიბუტების მნიშვნელობა, მდგომარეობა და როლი.

თუ მოდელში ყურადღება ენიჭება ობიექტების სტრუქტურულ ორგანიზაციას, ხდება მათი კავშირების იდენტიფიცირება, რომლებსაც აქვთ მიმართება მონაცემთა გაცვლასთან ურთიერთქმედების დროს. თუ ძირითადი ყურადღება ენიჭება დროით მოწესრიგებას, ხდება შეტყობინებათა ფორმირება, რომლებიც გადაიცემა ობიექტებს შორის.

როგორც წესი, ურთიერთქმედების დიაგრამები შეიცავენ: -ობიექტებს; -კავშირებს; -შეტყობინებებს.

თუ მოღვაწეობის დიაგრამა აღწერს გადასვლებს ერთი მოქმედებიდან მეორეზე, ურთიერთქმედების დიაგრამაზე აქცენტირება ხდება მართვის ნაკადების გადასვლებზე ობიექტიდან ობიექტზე. მათი მეშვეობით შესაძლებელია მართვის ნაკადების მოწესრიგება დროში და აგრეთვე მართვის ნაკადების სტრუქტურული ორგანიზაცია.

§ 2.3.1. მართვის ნაკადების მოწესრიგება დროში

მართვის ნაკადების დროის მიხედვით მოწესრიგებისათვის გამოიყენება ურთიერთქმედების დიაგრამის ერთ-ერთ სახე მიმდევრობითობის დიაგრამა, რომელიც ყურადღებას ამახვილებს შეტყობინებათა დროის მიხედვით მოწესრიგებაზე.

მიმდევრობითობის დიაგრამის ასაგებად საჭიროა განვალაგოთ ობიექტები, რომლებიც მონაწილეობს ურთიერთქმედებაში X ღერძის ზედა ნაწილის გასწვრივ. ჩვეულებრივ ურთიერთქმედების ინიციატორი ობიექტი თავსდება მარცხნივ, ხოლო დანარჩენები მარჯვნივ (რაც უფრო შორს არის, მით უფრო დამოკიდებული ობიექტია). შემდეგ Y ღერძის გასწვრივ განალაგებენ შეტყობინებებს, რომლებსაც ობიექტები აგზავნიან და ღებულობენ. ამასთან რაც უფრო გვიანია მით უფრო

ქვევითაა. გარდა ამისა დიაგრამაზე უჩვენებენ ობიექტის სიცოცხლის ხაზს. ეს არის ვერტიკალური წყვეტილი ხაზი, რომელიც ასახავს ობიექტის არსებობას დროში. ობიექტების უმრავლესობა, რომლებიც წარმოდგენილია ურთიერთქმედების დიგრამაზე, არსებობს მათი ურთიერთქმედების განმავლობაში, ამიტომ მათ გამოხატავენ დიაგრამის ზედა ნაწილში, ხოლო მისი სიცოცხლის ციკლი იხაზება ზევიდან ქვევით. ობიექტები შეიძლება იქმნებოდეს ურთიერთქმედების პერიოდშიც. მათი სიცოცხლის ციკლი იწყება create შეტყობინების მიღებით. ობიექტები ასევე შეიძლება დაიხუროს (განადგურდნენ) ურთიერთქმედების პროცესში, ასეთ შემთხვევაში მათი სიცოცხლის ციკლი მთავრდება destroy შეტყობინების მიღებით.

აღნიშნულ დიაგრამაზე უჩვენებენ აგრეთვე მართვის ფოკუსს. იგი გამოისახება სწორკუთხედით, რომელიც მიუთითებს დროის ინტერვალს, რომლის განმავლობაშიც ობიექტი ასრულებს გარკვეულ მოქმედებას უშუალოდ ან დამოკიდებული პროცედურით. სწორკუთხედის ზედა ხაზი გაუტოლდება დროის ღერძზე მოქმედების დაწყებას, ხოლო ქვედა - მის დამთავრებას.

შესაძლებელია მართვის ფოკუსის ჩართვა, გამოწვეული რეკურსიით (საკუთარ ოპერაციასთან მიმართვა) ან უკუგამოძახებით მეორე ობიექტის მხრიდან. მას უჩვენებენ მართვის მეორე ფოკუსის აგებით, ოდნავ მარცხნივ თავისი მშობლისაგან (დასაშვებია ჩართვა ნებისმიერი სიღრმით). თუ მართვის ფოკუსის განლაგება საჭიროა მიეთითოს მაქსიმალური სიზუსტით, შეიძლება სწორკუთხედი დაიშტრიხოს, იმ დროის შესაბამისად, რომლის განმავლობაშიც მეთოდი ნამდვილად მუშაობს და არ გადასცემს მართვას მეორე ობიექტს.

მართვის ნაკადების მოწესრიგებისათვის დროის მიხედვით თავიდან ადგენენ ურთიერთქმედების კონტექსტს, ეს იქნება სისტემა, ქვესისტემა, ოპერაცია ან პრეცედენტის ერთ-ერთი სცენარი. განსაზღვრავენ ურთიერთმოქმედ ობიექტებს და განალაგებენ მარცხნიდან მარჯვნივ ისე, რომ შედარებით მნიშვნელოვანი ობიექტები განლაგდეს უფრო მარჯვნივ. ადგენენ ყოველი ობიექტისათვის სიცოცხლის ხაზს. უფრო ხშირად ობიექტები არსებობენ მთელი

ურთიერთმოქმედების განმავლობაში. ხოლო იმ ობიექტებისათვის, რომლებიც ურთიერთმოქმედების პროცესში იხურება(ნადგურდება), სიცოცხლის ხაზზე ნათლად უნდა უჩვენონ დაბადებისა და სიკვდილის წერტილები შესაბამისი სტერეოტიპით. დაწყებული შეტყობინებიდან, რომელიც ურთიერთმოქმედების ინიცირებას ახდენს, განალაგებენ ყველა შემდეგ შეტყობინებას ზევიდან ქვევით ობიექტების სასიცოცხლო ხაზებს შორის, უნდა ჩანდეს ყოველი შეტყობინების თვისება, მაგალითად, მისი პარამეტრები. თუ საჭიროა შეტყობინებათა ჩართვა ან გამოთვლის ზუსტი ინტერვალის მითითება, შეავსებენ ობიექტების სიცოცხლის ციკლს მართვის ფოკუსით. თუ საჭიროა დროითი ან სივრცობრივი შეზღუდვების მითითება, შეავსებენ შეტყობინებას დროითი აღნიშვნებით და დაუკავშირებენ შესაბამის შეზღუდვებს. მართვის ნაკადების უფრო ფორმალური აღწერისათვის დაუკავშირებენ ყოველ შეტყობინებას წინა და შემდგომ პირობებს.

მიმდევრობითობის ყოველ დიაგრამაზე შეიძლება მხოლოდ ერთი მართვის ნაკადის ჩვენება, ამიტომ, როგორც წესი, ქმნიან ურთიერთქმედების რამდენიმე დიაგრამას, რომელთაგან ერთი ითვლება ძირითადად, ხოლო დანარჩენები აღწერენ ალტერნატიულ გზებს და განსაკუთრებულ პირობებს. მიმდევრობითობის დიაგრამების ასეთი ერთობლიობა, შეიძლება გაერთიანდეს პაკეტში და მიეთითოს თვითეულს შესაბამისი სახელი.

მაგალითისათვის განვიხილოთ პრეცედენტი "სპეციალობის სასწავლო გეგმის დადგენა" და აღვწეროთ სცენარი, რომელზედაც განხორციელდება ურთიერთქმედება (იხ. §2.1.2.). როგორც აღვნიშნეთ, ყოველ პრეცედენტში პროცესი შესაძლებელია სხვადასხვა სცენარით განვითარდეს, რომლებიც მოვლენათა ნაკადებით არის განსაზღვრული. შესაბამისად, ერთ გამოყენებით შემთხვევას შეიძლება ჰქონდეს რამოდენიმე ურთიერთმოქმედების დიაგრამა, ვინაიდან მისი განხორციელების სცენარი შეიძლება შედგებოდეს ალტერნატიულ მოვლენათა ნაკადებისაგან. ჩვენი მაგალითის შემთხვევაში პრეცედენტის მოქმედება იწყება მიმართვით სასწავლო გეგმის მომზადებაზე, რომელსაც თან ახლავს მოვლენათა ნაკადები, სასწავლო

საგნები (სპეციალობის, არჩევითი) დადგენილია ან არ არის დადგენილი, საგანზე განკუთვნილი კრედიტები დადგინდა ან არ დადგინდა. თითოეული ამ მოვლენის ფარგლებში პროცესი შესაძლებელია სხვადასხვა სცენარით განხორციელდეს, რომლებიც განაპირობებს დოკუმენტის ***სასწავლო გეგმა*** ფორმირების სხვადასვა ვარიანტს.

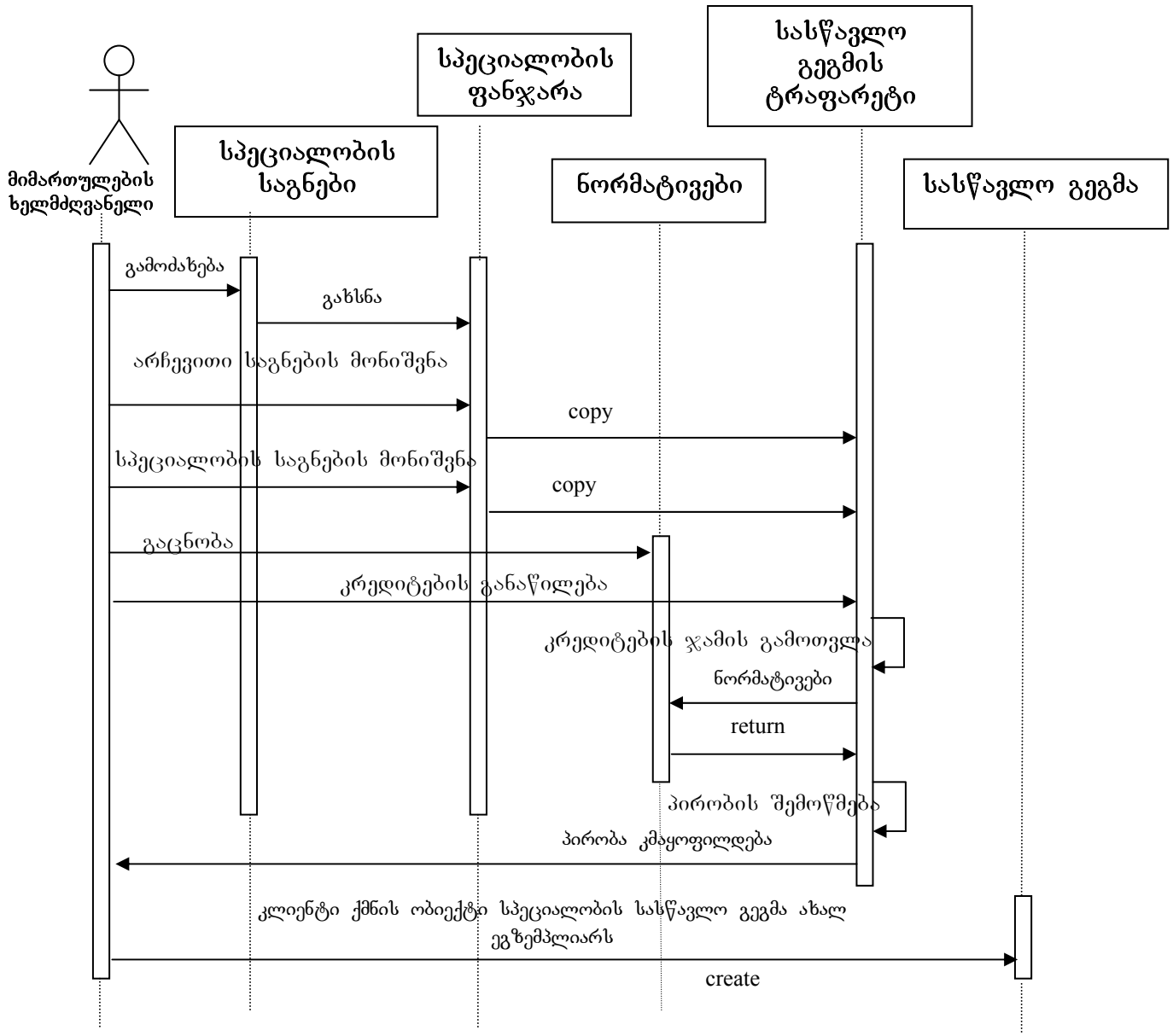
თითოეული ეს სცენარი აღიწერება დამოუკიდებელი ურთიერთმოქმედების დიაგრამით. ყველა შემთხვევაში თავდაპირველად ხდება მოცემულ პრეცედენტში მონაწილე ობიექტების იდენტიფიცირება, რომლებიც გარკვეულ როლს ასრულებენ და დადგინდება მათი საწყისი თვისებები, მათ შორის ატრიბუტების მნიშვნელობა, მდგომარეობა და როლი. ამის შემდეგ ხდება მათ შორის კავშირების იდენტიფიცირება.

2.3.1. ნახაზზე მოყვანილია მიმდევრობის დიაგრამა მართვის ძირითადი ნაკადისათვის – სპეციალობის სასწავლო გეგმის დადგენა(იხ. § 2.1.2.). ჩვეულებრივ სცენარის ინიციალიზაციას ახდენს კლიენტი (ჩვენ შემთხვევაში მიმართულების ხელმძღვანელი), რომელიც ირჩევს მთავარი ფანჯრიდან ახალი დოკუმენტის შექმნის ოპციას. განვიხილოთ მოქმედების მიმდევრობა:

1. მომხმარებელი ირჩევს ახალი დოკუმენტის(სასწავლო გეგმა) შექმნას;
2. ამის შემდეგ მთავარ ფანჯარას გამოაქვს ეკრანზე სპეციალობის სასწავლო გეგმისათვის შესარჩევი საგნების ნუსხა;
3. მომხმარებელი ირჩევს არჩევით საგნებს და მონიშნავს შესაბამის ველებს;
4. მონიშნული არჩევითი საგნები კოპირდება სასწავლო გეგმის ფორმაში;
5. მომხმარებელი ირჩევს სპეციალობის საგნებს და მონიშნავს შესაბამის ველებს;
4. მონიშნული სპეციალობის საგნები კოპირდება სასწავლო გეგმის ფორმაში;
5. მომხმარებელი ეცნობა საგნებზე კრედიტების დაწესების წესებს;
6. მომხმარებელი წესების გათვალისწინებით ანაწილებს კრედიტებს არჩეულ საგნებზე კურსებისა და სემესტრების მიხედვით;
7. გამოითვლება კრედიტების ჯამი სემესტრებისა და კურსების მიხედვით;

8. დგინდება ნორმატივები კრედიტების რაოდენობაზე სემესტრსა და კურსზე;

9. ნორმატივი უბრუნებს მოთხოვნილ მნიშვნელობებს;

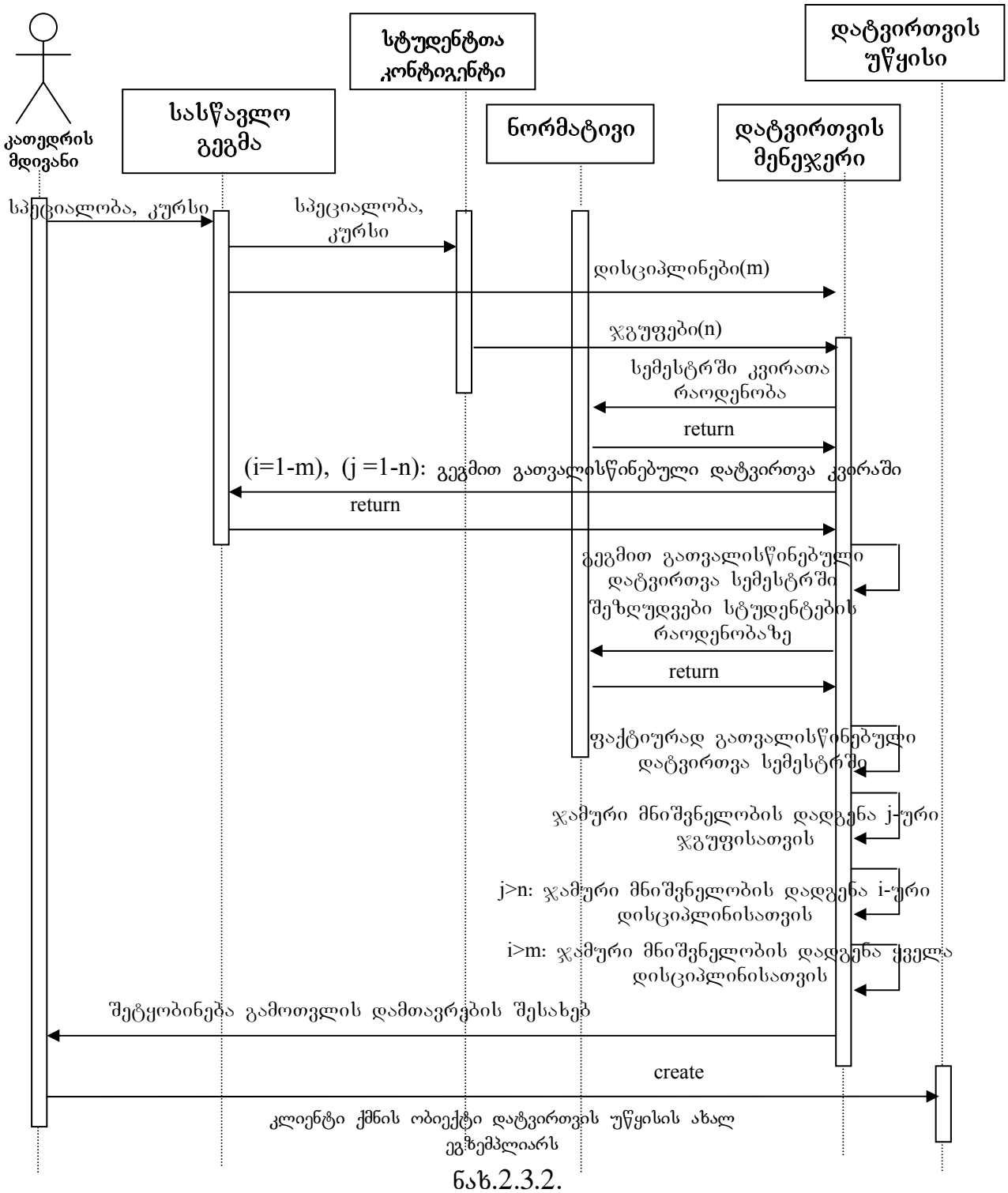


ნახ.2.3.1

10. მოწმდება პირობა - კრედიტების ჯამი სემესტრსა და კურსზე თუ აკმაყოფილებს ნორმატივებს;

11. მომხმარებელი ღებულობს შეტყობინებას პირობა კმაყოფილდება;

12. მომხმარებელი ქმნის კლასის "სპეციალობის სასწავლო გეგმა" ახალ ეგზემპლიარს და ავსებს მას მომხმარებლის მიერ შეყვანილი ინფორმაციით.



ნახ.2.3.2.

მიღებული სასწავლო გეგმა გამოიყენება დოკუმენტის "სპეციალობის დატვირთვა" შესაქმნელად. ნახ.2.3.2.-ზე ნაჩვენებია დოკუმენტის (სპეციალობის დატვირთვა) შექმნის სცენარი, რომელიც ასახავს იმ შემთხვევას, როდესაც მომხმარებელს აქვს დოკუმენტის შექმნის უფლება. მასზე ჩანს სხვადასხვა

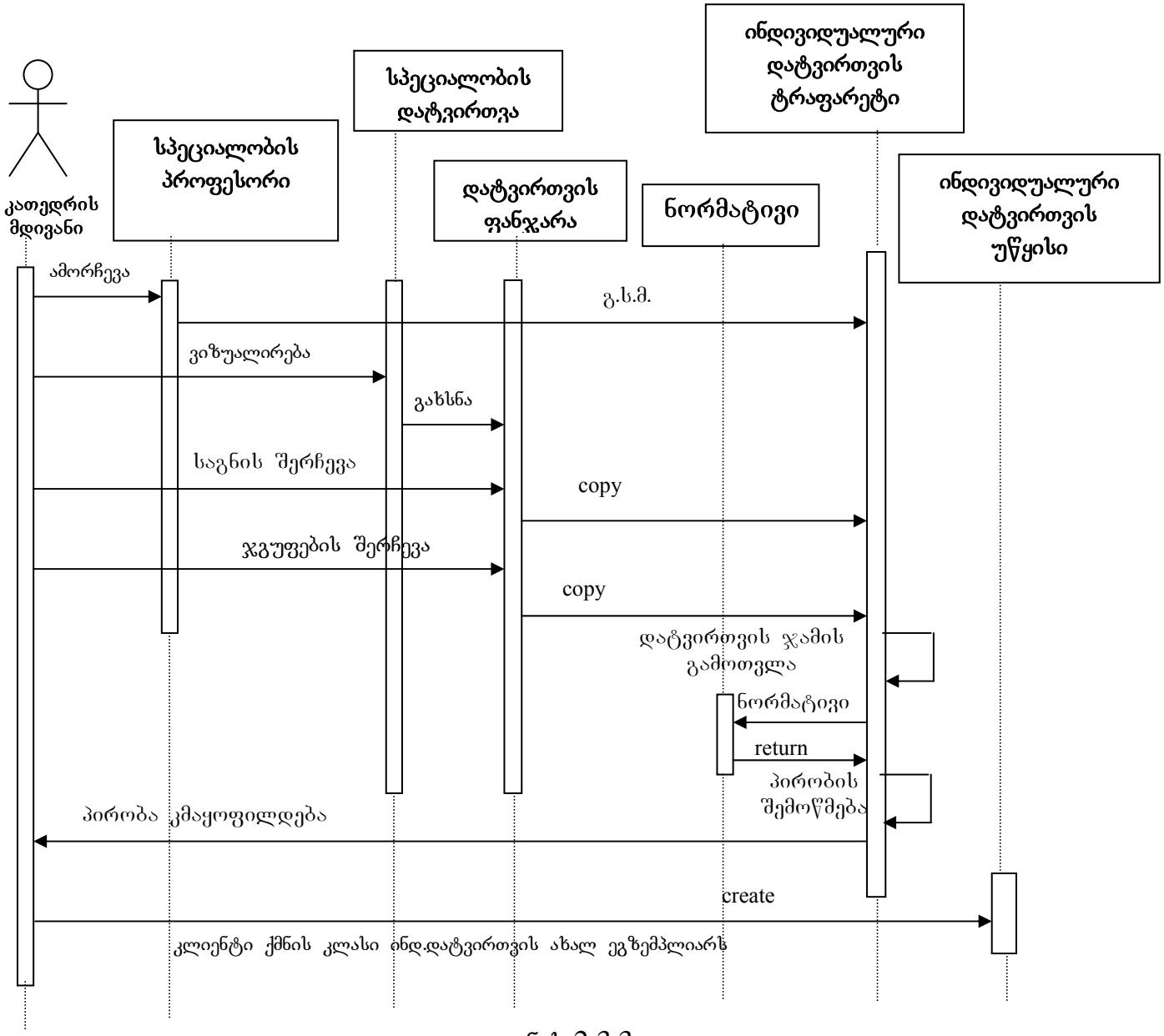
ობიექტების ურთიერთქმედება ამ შემთხვევისათვის. მოცემულ შემთხვევაში ობიექტიდან "სასწავლო გეგმა" სპეციალობისა და კურსის მიხედვით დგინდება ის დისციპლინები, რომლებიც ისწავლება და ობიექტიდან "სტუდენტთა კონტიგენტი" ჯგუფები, რომლებიც ირიცხებიან მოცემული სპეციალობის მითითებულ კურსზე (ჯგუფის ნომერი და ჯგუფში სტუდენტების რაოდენობა). "ნორმატივებიდან" დგინდება სემესტრში კვირათა რაოდენობა, ხოლო "სასწავლო გეგმიდან" გეგმით გათვალისწინებული დატვირთვა კვირაში(ლექცია, პრაქტიკული, ლაბორატორია, საკურსო სამუშაო). მიღებული მონაცემების საფუძველზე დატვირთვის მენეჯერი ანგარიშობს მოცემულ დისციპლინაზე გეგმით გათვალისწინებულ დატვრთვას სემესტრში თითოეული ჯგუფისათვის.

ჯგუფში სტუდენტთა კონტიგენტის გათვალისწინებით გამოითვლება გამოცდასა (სამ სტუდენტზე ერთი საათი), რეიტინგზე საჭირო საათები და იანგარიშება ფაქტიური დატვირთვა სემესტრში თითოეული ჯგუფისა და მთლიანად დისციპლინისათვის.

აღნიშნულ პროცესს დოკუმენტების მენეჯერი იმეორებს დისციპლინების რაოდენობის(m-ჯერ), ჯგუფების რაოდენობის (n-ჯერ) და კურსების(4) მიხედვით. ბოლოს იანგარიშება ჯამები ცალკეული ნომინალების (ლექცია, პრაქტიკული, ლაბორატორია, საკურსო სამუშაო, პრაქტიკა, მაგისტრატურა, გამოცდა, რეიტინგი, მთლიანი ჯამი) მიხედვით. გამოთვლის დამთავრების შესახებ შეტყობინების მიღების საფუძველზე მომხმარებელი ქმნის ობიექტი "დატვირთვის უწყისის" ახალ ეგზემპლიარს, რომელიც გამოიყენება "ინდივიდუალური დატვირთვის" დოკუმენტის ფორმირებისათვის.

ნახ.2.3.3.-ზე ნაჩვენებია დოკუმენტის (ინდივიდუალური დატვირთვა) შექმნის სცენარი, რომელიც ასევე ასახავს იმ შემთხვევას როდესაც მომხმარებელს აქვს დოკუმენტის შექმნის უფლება. მოცემულ შემთხვევაში ხდება სპეციალობის პროფესორის ამორჩევა, რომელიც გადაიტანება ინდივიდუალური დატვირთვის ტრაფარეტში. შემდეგ ხდება სპეციალობის დატვირთვის ვიზუალირება ეკრანზე, საიდანაც ამოირჩევა დისციპლინა, ჯგუფის ნომერი და შესაბამისი დატვირთვა

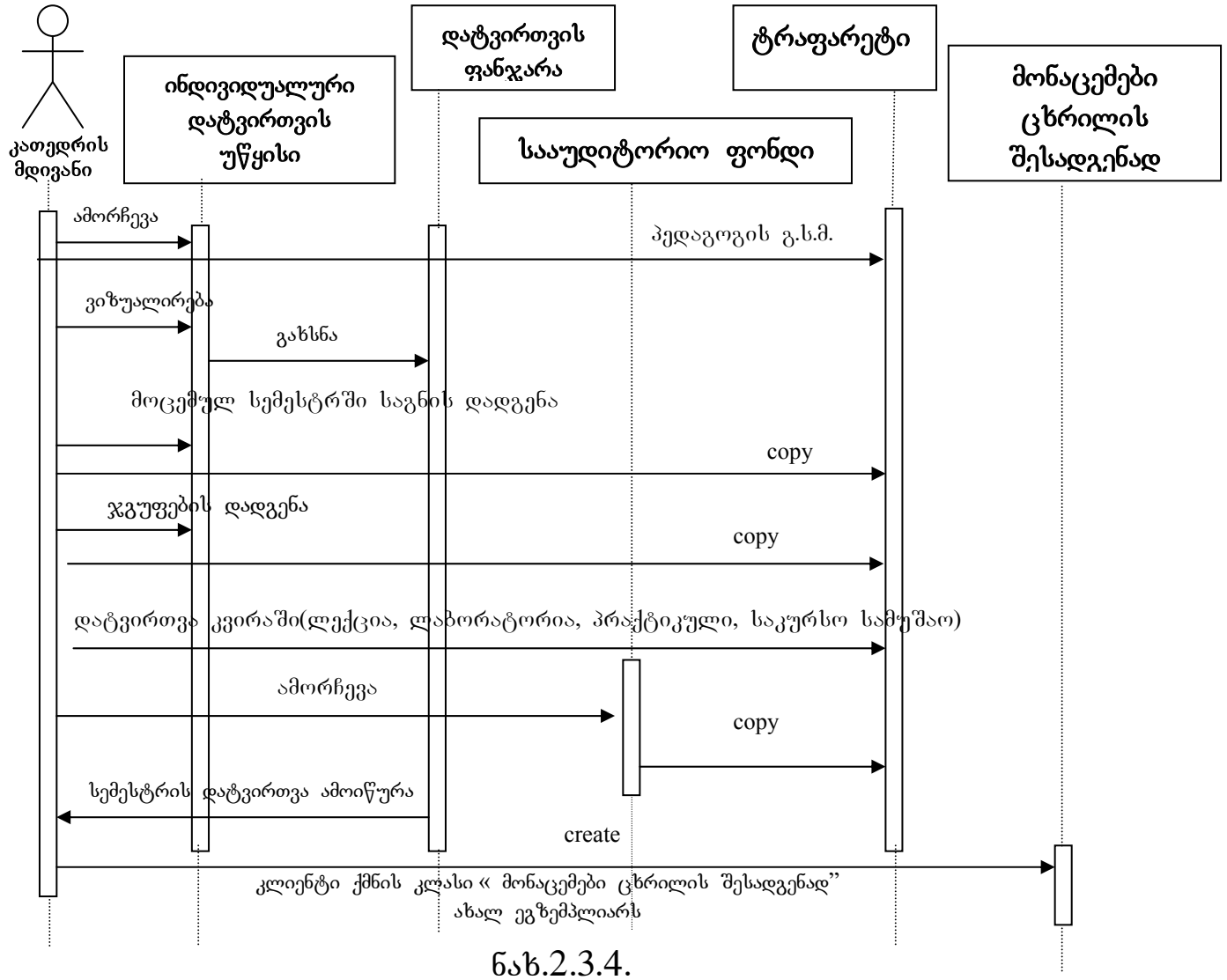
(ლექცია, პრაქტიკული, ლაბორატორია, საკურსო სამუშაო, გამოცდა, რეიტინგი), რომელიც ასევე გადაიტანება ‘‘ინდივიდუალური დატვირთვის ტრაფარეტში’’.



ნახ.2.3.3.

გამოითვლება ამორჩეული დატვირთვის ჯამი და პროცესი მეორდება სანამ ეს ჯამი არ გახდება ინდივიდუალური დატვირთვისათვის ნორმატივით გათვალისწინებულის (600 სთ) ტოლი(ან მეტი). პირობის დაკმაყოფლების შესახებ შეტყობინების მიღების საფუძველზე მომხმარებელი ქმნის ობიექტი ‘‘ინდივიდუალური დატვირთვის უწყისის’’ ახალ ეგზემპლიარს, რომელიც გამოიყენება დოკუმენტის ფორმირებისათვის ’’მონაცემები ცხრილის შესადგენად’’.

ნახ.2.3.4.-ზე ნაჩვენებია დოკუმენტის (მონაცემები ცხრილის შესადგენად) შექმნის სცენარი.



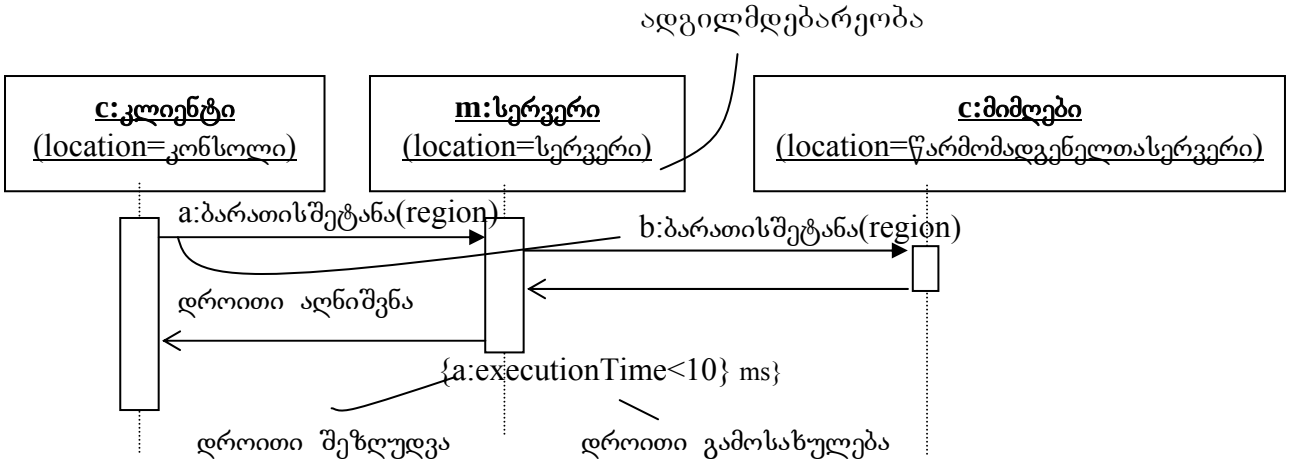
ნახ.2.3.4.

ისევე როგორც წინა შემთხვევაში ურთიერთქმედების ინიცალზაციას ახდენს მომხმარებელი (კათედრის მდივანი), რომელიც იძახებს ცალკეული პროფესორების დატვირთვას ეკრანზე სემესტრის გათვალისწინებით, მონიშნავს და გადააქვს დისციპლინები, ჯგუფები და მათზე გათვალისწინებული დატვირთვები(ლექცია, პრაქტიკული, ლაბორატორია, საკურსო სამუშაო) კვირაში დოკუმენტის ტრაფარეტში. თუ დისციპლინა მოითხოვს სპეციალურ სალექციო(ლაბორატორიული) აიდიტორიას იგი დგინდება ობიექტიდან

საუდიტორიო ფონდი და ასევე გადაიტანება დოკუმენტში. მას შემდეგ რაც მომხმარებელი მიიღებს შეტყობინებას დატვირთვის ფანჯრიდან სემესტრში დატვირთვის დამთავრების შესახებ იგი ქმნის დოკუმენტის "მონაცემები ცხრილის შესადგენად" ახალ ეგზემპლიარს.

რეალური დროის სისტემებში ერთ ერთი მნიშვნელოვანი ელემენტია დროს მოდელირება.

რეალური დროის სისტემებს უწოდებენ იმიტომ, რომ იგი თავის ფუნქციებს უნდა ასრულებდეს მკაცრად განსაზღვრულ აბსოლუტურ ან შეფარდებით დროის მომენტებში და ამაზე ხარჯავდეს წინასწარ განსაზღვრულ ან ხშირად შეზღუდულ დროს. არსებულ სისტემებს შორის არსებობს ისეთები, რომლებისთვისაც რეაქციის საჭირო დრო აღირიცხება ნანო ან მილიწამებით.



ნახ.2.3.5. დროითი შეზღუდვა

მაგრამ გვხვდება თითქმის რეალური დროის სისტემები, რომლებისთვისაც რეაქციისათვის დასაშვები დრო – წამები ან უფრო მეტიც არის. რეალური დროის სისტემების მოდელირების მოთხოვნილების დასაკმაყოფილებლად შემოტანილია გრაფიკული წარმოდგენა დროითი ნიშნულის, დროითი გამოსახულების და დროითი შეზღუდვის, როგორც ეს ნაჩვენებია ნახ.2.3.5.-ზე.

დროითი ნიშნული გამოიყენება დროის მომენტის აღნიშვნისათვის, რომელშიც მოხდა მოვლენა. იგი გამოისახება გამოსახულებით, დამოკიდებული სახელისაგან, რომელიც ენიჭება შეტყობინებას.

დროითი გამოსახულება ეს გამოსახულებაა, რომლის მნიშვნელობას წარმოადგენს აბსოლუტური ან შეფარდებითი დრო. *დროითი შეზღუდვა* ეს სემანტიკური მტკიცებაა შეფარდებითი ან აბსოლუტური დროის შესახებ.

რეალური დროის სისტემები, როგორც ეს დასახელებიდან გამომდინარეობს, მკაცრნი არიან დროის მიმართ. მოვლენები მათში შეიძლება წარმოებდეს რეგულარულად ან სპონტანურად, მაგრამ ნებისმიერ შემთხვევაში რეაქციის დრო მოვლენაზე უნდა იყოს განსაზღვრული ან აბსოლუტური ხანგრძლივობით, ან მოვლენის აღძვრის მომენტის მიმართ.

შეტყობინებების გადაცემა – ეს სისტემის ერთ-ერთი დინამიური ასპექტია, ამიტომ სისტემის დროითი ფაქტორების მოდელირებისას შეიძლება ყოველ შეტყობინებას, რომლებიც მონაწილეობას ღებულობენ ურთიერთქმედებაში მივცეთ სახელი, რომელიც გამოიყენება როგორც დროითი ნიშნული.

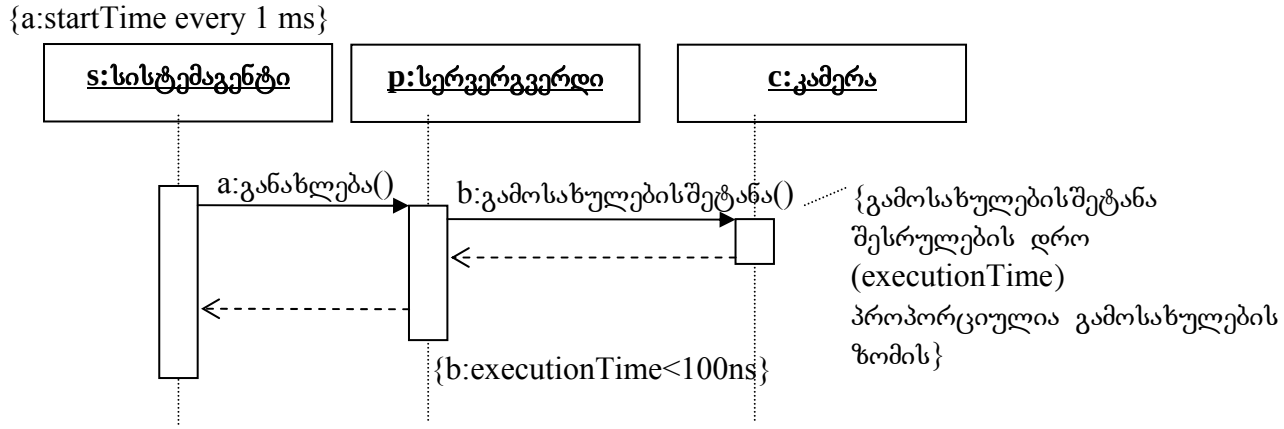
დროითი გამოსახულებების აგებისათვის გამოიყენება ფუნქციები – **startTime** (დაწყების დრო), **stopTime**(დამთავრების დრო), **executionTime**(შესრულების დრო), მასში გვხვდება შეტყობინების სახელიც, რომელიც მონაწილეობას ღებულობს ურთიერთქმედებაში. შესაბამისად, თუ მოცემულია შეტყობინების სახელი, ეს სამი ფუნქცია შეიძლება გამოვიყენოთ ნებისმიერი სირთულის დროითი გამოსახულებების აგებისათვის.

მოვლენის აბსოლუტური დრო, ინტერვალი მოვლენებს შორის და დრო, რომელიც საჭიროა მოქმედების შესრულებისათვის – აი სამი ძირითადი დროითი ასპექტი რეალური დროის სისტემებისათვის, რომელთა მოდელირებისას გამოიყენებიან დროითი შეზღუდვები.

დროითი შეზღუდვების მოდელირებისას ყოველი მოვლენისათვის ურთიერთქმედებაში უნდა დადგინდეს, იწყება თუ არა იგი დროის განსაზღვრულ

აბსოლუტურ მომენტში. ეს თვისება აღინიშნება მოვლენაზე დროითი შეზღუდვის საშუალებით. შეტყობინებების ყოველი ინტერესმქონე თანმიმდევრობისათვის ურთიერთქმედებაში ასევე განისაზღვრება, შეზღუდულია თუ არა მისი შესრულების დრო. ეს თვისებაც აღინიშნება თანმიმდვრობაზე დროითი შეზღუდვის საშუალებით. ყოველი დროში კრიტიკული ოპერაციისათვის განიხილება მისი დროითი სირთულე. ეს სემანტიკა დამოდელირდება ოპერაციაზე დროითი შეზღუდვის საშუალებით.

მაგალითად, 2.3.6. ნახაზზე მარცხენა შეზღუდვა ადგენს საწყის დროს განმეორებადი მოვლენის ***განახლება*** გამოძახებისათვის. დროითი შეზღუდვა ნახაზის შუაში, ადგენს ***გამოსახულებისშეტანა*** გამოძახების მაქსიმალურ ხანგრძლივობას. ბოლოს, მარჯვენა შეზღუდვა ადგენს ***გამოსახულებისშეტანა*** გამოძახების მოვლენის დროით სირთულეს.



ნახ.2.3.6. დროითი შეზღუდვების მოდელირება

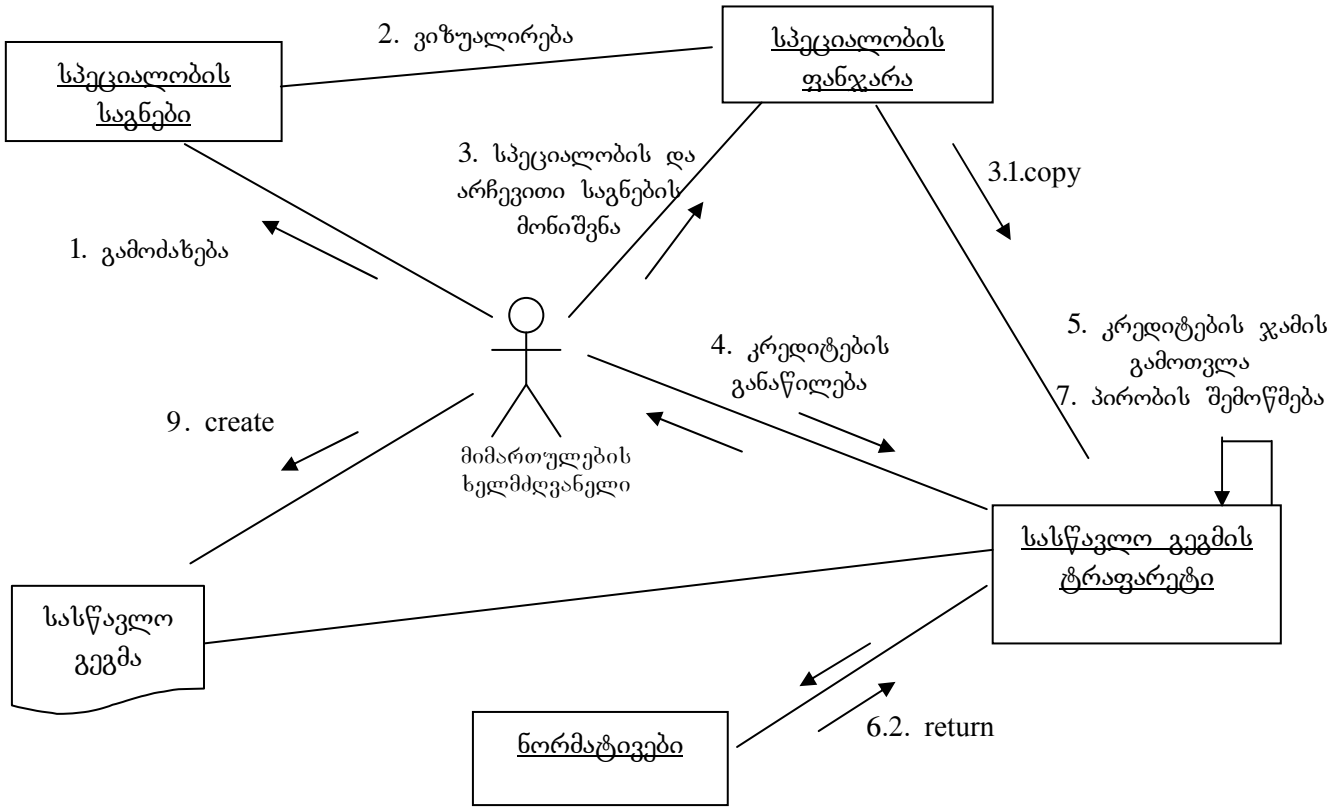
§ 2.3.2. მართვის ნაკადების სტრუქტურული ორგანიზაცია

მართვის ნაკადების სტრუქტურული ორგანიზაციის მოდელირებისათვის გამოიყენება ურთიერთქმედების დიაგრამის მეორე სახე - კოოპერაციის (თანამოქმედების) დიაგრამას, რომელიც ყურადღებას ამახვილებს შეტყობინების გამგზავნი და მიმღები ობიექტების სტრუქტურულ ორგანიზაციაზე.

კოოპერაციის დიაგრამის შექმნისათვის ურთიერთქმედებაში მონაწილე ობიექტები უნდა განლაგდეს გრაფის მწვერვალების სახით. შემდეგ კავშირები, რომლებიც ამ ობიექტებს აკავშირებს, გამოისახება ამ გრაფის წიბოების სახით. კავშირებზე უთითებენ შეტყობინებებს, რომლებსაც ობიექტები ღებულობენ ან აგზავნიან. ეს საშუალებას აძლევს მომხმარებელს მიიღოს ნათელი წარმოდგენა მართვის ნაკადზე.

2.3.6. ნახაზზე მოყვანილია ზემოთ განხილული მიმდევრობითობის დიაგრამის (სასწავლო გეგმის შედგენა) შესაბამისი კოოპერაციის დიაგრამა. დიაგრამაზე წარმოდგენილია ხუთი ობიექტი: ***მიმართულების ხელმძღვანელი, სპეციალობის საგნები, სპეციალობის ფანჯარა, ნორმატივები, სასწავლო გეგმის ტრაფარეტი, სასწავლო გეგმა***. მოვლენათა ნაკადი დანომრილია თანამიმდევრულად.

მოქმედება იწყება იმით, რომ კათედრის მდივანი ახდენს ***სპეციალობის საგნების*** ვიზუალირებას ***სპეციალობის ფანჯარაში***. მიმართულების ხელმძღვანელი მონიშნავს რა სპეციალობის და არჩევით საგნებს, გადაიტანება სასწავლო გეგმის ტრაფარეტში. ***ნორმატივებიდან*** დგინდება საგნებზე დასაშვები კრედიტების ოდენობა და სემესტრსა(კურსზე) მათი ზღვრული მნიშვნელობა. ***სასწავლო გეგმის ტრაფარეტი*** ახდენს კრედიტების ჯამის გამოთვლას სემესტრში(კურსზე) და ამოწმებს ნორმით გათვალისწინებულ პირობას. მას შემდეგ რაც კლიენტი(სპეციალობის ხელმძღვანელი) ღებულობს შეტყობინებას, რომ პირობა დაცულია იგი ქმნის დოკუმენტს ***სპეციალობის სასწავლო გეგმა*** ახალ ეგზემპლიარს.



ნახ. 2.3.6.

მიმდევრობითი ნაკადების მოდელირების გარდა შესაძლებელია უფრო რთული ნაკადების მოდელირება, როგორიც არის იტერაცია და განშტოება. იტერაცია წარმოადგენს შეტყობინებათა განმეორებად თანამიმდევრობას. მისი მოდელირებისათვის შეტყობინების ნომრის წინ თანამიმდევრობაში ისმება იტერაციის გამოსახულება მაგ., **i=1-n**. იტერაცია უჩვენებს, რომ შეტყობინება განმეორდება მოცემული გამოსახულების შესაბამისად. ანალოგიურად, პირობა წარმოადგენს შეტყობინებას, რომლის შესრულება დამოკიდებულია ბულის გამოსახულების გამოთვლის შედეგზე. პირობის მოდელირებისათვის შეტყობინების რიგითი ნომრის წინ ისმება გამოსახულება მაგ., **x>0**. ყველა ალტერნატიულ შტოს ექნება ერთი და იმავე ნომერი, მაგრამ პირობა ყოველი შტოსათვის უნდა დაენიშნოს ისე, რომ ორი მათგანი მათ შორის არ სრულდებოდეს ერთდროულად.

ურთიერთქმედებისას ობიექტების ატრიბუტთა მნიშვნელობები, მათი მდგომარეობა ან როლი, როგორც წესი იცვლება. იგი შეიძლება გამოიხატოს ობიექტის ასლის შექმნით ატრიბუტთა სხვა მნიშვნელობით, მდგომარეობით და

როლით. ურთიერთქმედების დიაგრამაზე მათ აკავშირებენ შეტყობინების სტერეოტიპით **become**.

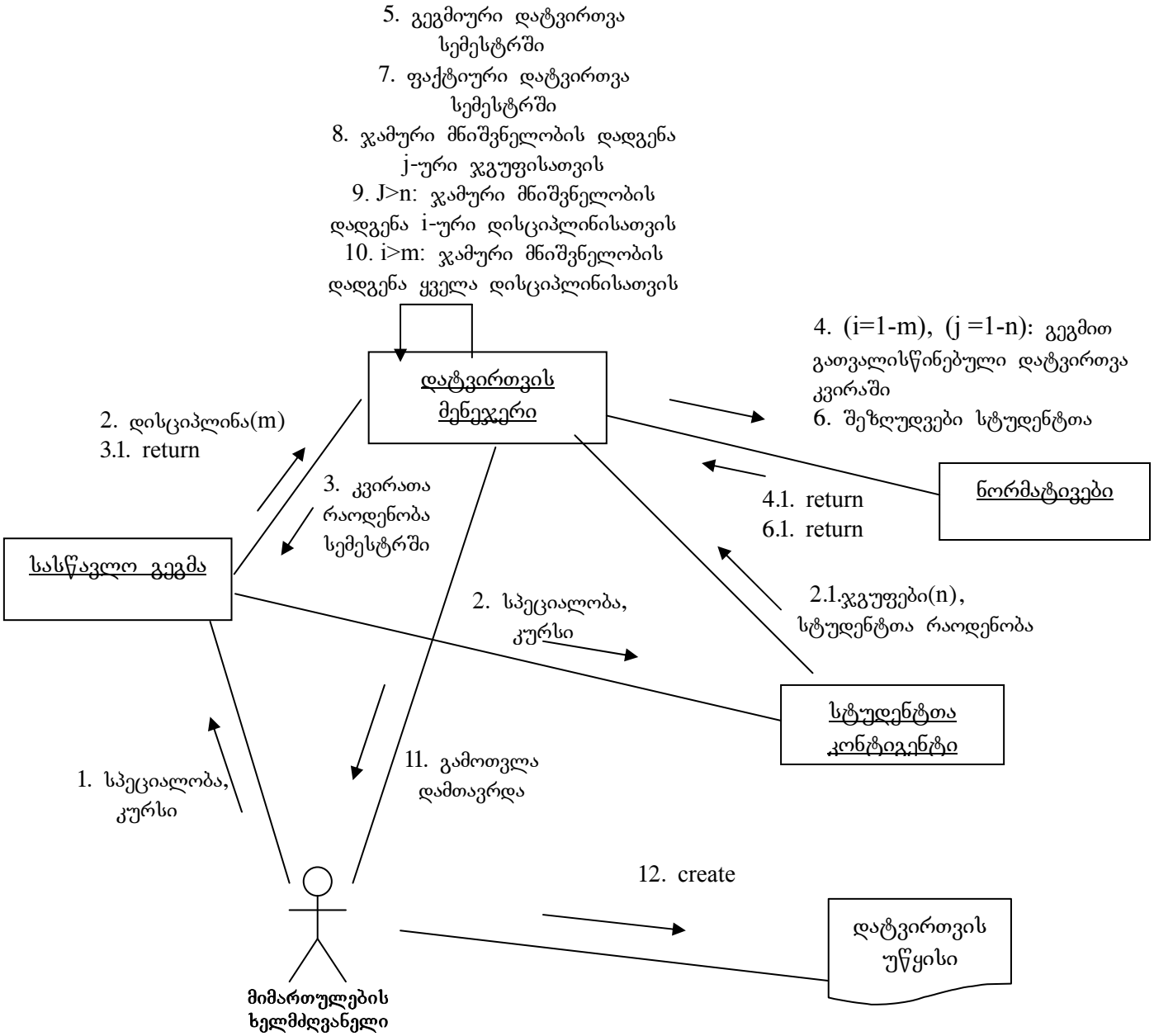
ისევე როგორც მიმდევრობითობის დიაგრამაზე, ერთ კოოპერაციის დიაგრამაზე შესაძლებელია უჩვენონ მხოლოდ ერთი მართვის ნაკადი, ამიტომ როგორც წესი, ქმნიან ურთიერთქმედების რამოდენიმე დიაგრამას, რომელთაგანაც ერთი ითვლება ძირითადათ, ხოლო დანარჩენები აღწერენ ალტერნატიულ გზებს.

2.3.7.-ზე მოყვანილია კოოპერაციის დიაგრამა, რომელიც შეესაბამება პრეცედენტს "დატვირთვის შედგენა", რომელზეც ასახულია ორი იტერაცია. ერთი ასახავს განმეორებად შეტყობინებებს კურსზე შესასწავლი ყოველი ახალი დისციპლინისათვის(i=1-m), ხოლო მეორე(ჩართულია პირველში) ასახავს განმეორებად შეტყობინებებს კურსზე არსებული ყოველი ჯგუფისათვის(j =1-n).

ამრიგად, მართვის ნაკადების სტრუქტურული ორგანიზაციის მოდელირებისათვის უნდა დადგინდეს ურთიერთქმედების კონტექსტი, ეს იქნება სისტემა, ქვესისტემა, ოპერაცია ან პრეცედენტის ერთ-ერთი სცენარი.

განვსაზღვროთ ურთიერთქმედების სცენარი, დადგინდება რა, თუ რომელი ობიექტები მონაწილეობენ მასში. განალაგებენ მათ კოოპერაციის დიაგრამაზე გრაფის მწვერვალებად, ისე რომ მნიშვნელოვანი ობიექტები აღმოჩნდეს დიაგრამის ცენტრში, ხოლო მათი მეზობლები ნაპირზე. განსაზღვრავენ თითოეული ამ ობიექტის საწყის თვისებებს. თუ ატრიბუტის მნიშვნელობა, ობიექტების როლი და მდგომარეობა ურთიერთქმედებისას იცვლება, დიაგრამაზე მოათავსებენ დუბლიკატებს ახალი მნიშვნელობებით და დააკავშირებენ მათ შეტყობინების სტერეოტიპით **become** და **copy**, შესაბამისი რიგითი ნომრების დართვით. დეტალურად აღიწერება კავშირები ობიექტებს შორის, რომელთა გასწვრივაც გადაიცემა შეტყობინებები. თავიდან მიეთითება კავშირი ასოციაცია. ისინი ყველაზე მნიშვნელოვანია, რამდენადაც წარმოადგენს სტრუქტურულ შეერთებებს. დაწყებული შეტყობინებიდან, რომელიც ურთიერთქმედების ინიცირებას ახდენს,

დაუკავშირებენ ყველა დანარჩენ კვანძებს შესაბამის შეტყობინებებს, მოინიშნება რიგითი ნომრები.



ნახ.2.3.7.

თ ა ვ ი 3

სისტემის დაპროექტება

3.1. კლასები

სისტემის მოდელირება ობიექტ-ორიენტირებული მიდგომით გულისხმობს არსთა იდენტიფიცირებას, რომლებიც მნიშვნელოვანია ამა თუ იმ თვალსაზრისით. თუ რომელი არსები აირჩევა და როგორი კავშირები დამყარდება მათ შორის, ცხადია დამოკიდებულია იმაზე, თუ როგორ იქნება გამოყენებული სისტემა.

შესაბამისად დაპროექტების პირველ ეტაპზე უნდა განისაზღვროს რომელ ელემენტებს-აბსტრაქციებს იყენებენ მომხმარებლები და დამმუშავებლები თითოეულ პრეცედენტში გათვალისწინებული მოქმედებების აღწერისა და მისი გადაწყვეტისათვის.

ობიექტ-ორიენტირებულ სისტემებში ყველა არსი მოდელირდება როგორც კლასები. კლასი ეს არსთა აბსტრაქციაა, იგი წარმოადგენს არა ინდივიდუალურ ობიექტს, არამედ არსთა ერთობლიობას. ტერმინებს "სასწავლო გეგმა" და "ჩემი სასწავლო გეგმა" არსებობს ფუნდამენტალური განსხვავება. პირველი – ეს მხოლოდ აბსტრაქციაა, რომელიც აღწერს სასწავლო გეგმის გარკვეულ ტიპს სხვადასხვა თვისებებით, მაშინ როდესაც მეორე წარმოადგენს კონკრეტულ ეგზემპლიარს ამ აბსტრაქციის, რომელიც არსებობს რეალურ სამყაროში და მის ყოველ თვისებას აქვს რეალური მნიშვნელობა.

აბსტრაქცია აღწერს საგნის იდეალურ არსს, ეგზემპლიარი – მის კონკრეტულ მატერიალიზაციას. ერთ აბსტრაქციას შეიძლება გააჩნდეს რამოდენიმე ეგზემპლიარი. მოცემული ეგზემპლიარისათვის ყოველთვის არსებობს აბსტრაქცია, რომელიც განსაზღვრავს საერთო მახასიათებლებს ყველა მსგავსი ეგზემპლიარებისათვის.

ეგზემპლიარს(**Instance**) უწოდებენ აბსტრაქციის კონკრეტულ მატერიალიზაციას, რომლის მიმართ შეიძლება გამოყენებულ იქნას ოპერაციები და რომელსაც შეუძლია შეინახოს მათი შედეგები. ცნებები "ეგზემპლიარი" და "ობიექტი" პრაქტიკულად სინონიმებია. ეგზემპლიარს გამოხატავენ ხაზგასმული სახელით.

ობიექტები – ეს კლასების ეგზემპლიარებია. ობიექტი არა მარტო იკავებს ადგილს რეალურ სამყაროში, მათზე შესაძლებელია ასევე მანიპულირება. ოპერაციები, რომლებიც სრულდება ობიექტებზე, გამოცხადდება მის აბსტრაქციაში. ეს მიუთითებს იმაზე, რომ ყოველი ობიექტი ხასიათდება მდგომარეობით.

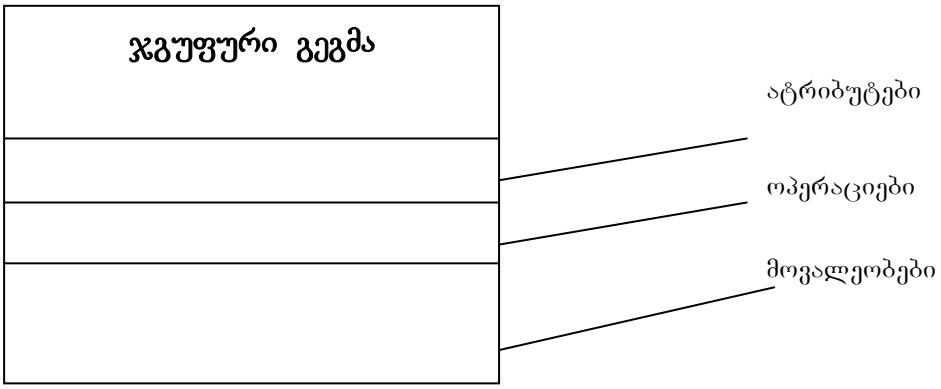
ობიექტის მდგომარეობას უწოდებენ მისი ყველა თვისებების ერთობლიობას და მათ მიმდინარე მნიშვნელობებს. თვისებათა რიცხვში შედის ობიექტთა ატრიბუტები. მაშასადამე, ობიექტის მდგომარეობა დინამიურია და მისი ვიზუალიზაციისას ფაქტიურად აღიწერება მისი მდგომარეობა დროის მოცემულ მომენტში და სივრცის მოცემულ წერტილში. ვასრულებთ რა ობიექტზე ოპერაციას, ფაქტიურად ვცვლით მის მდგომარეობას. მაგრამ, აქვე უნდა აღინიშნოს, რომ ობიექტის გამოკითხვისას მდგომარეობა არ იცვლება.

მაშასადამე, ***კლასს*** უწოდებენ ობიექტთა ერთობლიობის აღწერას საერთო ატრიბუტებით, ოპერაციებით, მიმართებებით და სემანტიკით.

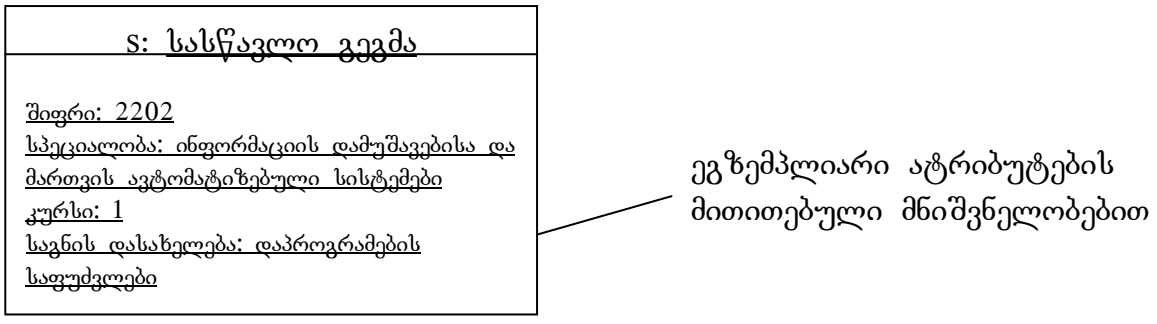
აქედან გამომდინარე შეიძლება ჩაითვალოს, რომ "სასწავლო გეგმა" არის ობიექტების კლასი, საერთო თვისებებით, როგორიც არის სპეციალობა, კურსი, ჯგუფის ნომერი, დისციპლინის დასახელება და ა. შ. ამასთან ცალკეული სპეციალობის ჯგუფური გეგმები განიხილება როგორც კლასი "სასწავლო ჯგუფური გეგმა"-ს ეგზემპლიარები ანუ ობიექტები.

ყოველ კლასს უნდა გააჩნდეს **სახელი**, რომელიც განასხვავებს მას სხვა კლასებისაგან. კლასის სახელი - ეს ტექსტური სტრიქონია. სახელის გარდა კლასის დახასიათებისათვის გამოიყენება ატრიბუტები, ოპერაციები და მოვალეობები.

კლასის გრაფიკული გამოსახვისას მიუთითებენ როგორც სახელს, ისე ატრიბუტებსა და ოპერაციებს. მოვალეობებს გამოსახავენ კლასის ქვედა ნაწილის სპეციალურ განყოფილებაში. ისინი შეიძლება მიუთითოთ აგრეთვე შენიშვნებში.



ატრიბუტი - კლასის დასახელებული თვისებაა, რომელიც შეიცავს მნიშვნელობათა სიმრავლეს, რომელსაც ღებულობენ ამ თვისების ეგზემპლიარები. კლასს შეიძლება ქონდეს ატრიბუტების ნებისმიერი რაოდენობა ან საერთოდ არ ქონდეს. ატრიბუტი წარმოადგენს სამოდელო არსის გარკვეულ თვისებას, რომელიც საერთოა მოცემული კლასის ყველა ობიექტისათვის. მაგალითად, კლიენტის მოდელირებისას შეიძლება მიუთითოთ გვარი, სახელი, მისამართი, ტელეფონი და დაბადების თარიღი. მაშასადამე, ატრიბუტი წარმოადგენს ობიექტის მონაცემების ან მისი მდგომარეობის აბსტრაქციას. დროის ყოველ მომენტში ობიექტის ნებისმიერ ატრიბუტს გააჩნია სავსებით გარკვეული მნიშვნელობა. ატრიბუტის სახელი, ისევე როგორც კლასის, შეიძლება იყოს ნებისმიერი ტექსტური სტრიქონი.



ოპერაციას უწოდებენ მომსახურების რეალიზაციას, რომელიც შეიძლება მოთხოვნილ იქნას მოცემული კლასის ნებისმიერი ობიექტისაგან ქცევაზე ზემოქმედებისათვის. სხვა სიტყვებით რომ ითქვას, ოპერაცია – ეს აბსტრაქციაა

იმისა, თუ რისი გაკეთება შეიძლება ობიექტზე. კლასს შეიძლება ქონდეს ოპერაციების ნებისმიერი რაოდენობა ან საერთოდ არ ქონდეს. ოპერაციის სახელი, ისევე როგორც კლასის, შეიძლება იყოს ნებისმიერი ტექსტური სტრიქონი. ოპერაციის აღწერისას შეიძლება მიუთითოთ პარამეტრები, მათი ტიპი და მნიშვნელობები, ფუნქციების შემთხვევაში – დასაბრუნებელი მნიშვნელობის ტიპი.

კლასის **მოვალეობები** - ეს თავისებურად კონტრაქტია, რომელსაც იგი უნდა დაემორჩილოს. განსაზღვრავენ რა კლასს შესაბამისად მიეთითება, რომ ყველა მის ობიექტებს აქვთ ერთგვაროვანი მდგომარეობა და ქცევა. შესაბამისი ატრიბუტები და ოპერაციები წარმოადგენენ სწორედ იმ თვისებებს, რომლის მეშვეობითაც სრულდება კლასის მოვალეობები. მაგალითად, კლასი "სასწავლო გეგმა" პასუხს აგებს ინფორმაციაზე კურსზე არსებულ სასწავლო ჯგუფებზე და შესასწავლ დისციპლინებზე, მათზე გამოყოფილ კრედიტებზე და ა.შ.

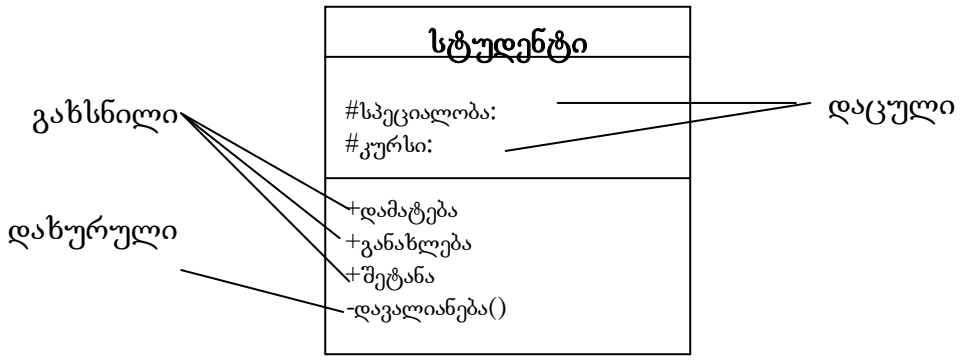
კლასები, ხასიათდებიან ასევე ისეთი დამატებითი თვისებებით(გარდა იმ მარტივებისა, რომელიც განხილული იყო), როგორიც არის ხედვა, მოქმედების არე, ჯერადობა და სხვა.

ხედვა. ერთ-ერთი დეტალი, რომელიც საკმაოდ მნიშვნელოვანია ატრიბუტებისა და ოპერაციებისათვის ეს არის მათი ხედვა. თვისების ხედვა მიუთითებს შეიძლება თუ არა იგი გამოყენებულ იქნას სხვა კლასის მიერ. ბუნებრივია, ეს გულისხმობს თვით კლასის ხედვას. ერთ კლასს შეუძლია "შეხედოს" მეორეს, თუ იგი იმყოფება პირველის მოქმედების სფეროში და მათ შორის არსებობს პირდაპირი ან უშუალო მიმართება. ობიექტ-ორიენტირებულ სისტემებში არსებობს ხედვის სამი დონე:

- **public(**გახსნილი) – ნებისმიერი გარე კლასი, რომელიც "ხედავს" მოცემულს შეუძლია ისარგებლოს მისი გახსნილი თვისებებით. აღინიშნება ნიშნით +(შეკრების ნიშანი) ატრიბუტის ან ოპერაციის წინ.
- **proteqted**(დაცული) – ნებისმიერ შთამომავალს მოცემული კლასისა შეუძლია ისარგებლოს მისი დაცული თვისებებით. აღინიშნება #(დიეზ) ნიშნით.

- **private**(დახურული) – მხოლოდ მოცემულ კლასს შეუძლია ისარგებლოს დახურული თვისებებით. აღინიშნება სიმბოლოთი –(გამოკლების ნიშანი).

კლასის თვისებების ხედვას განსაზღვრავენ იმისათვის, რომ დამალონ მისი რეალიზაციის დეტალები და უჩვენონ მხოლოდ ის თავისებურებანი, რომლებიც აუცილებელია მოვალეობების განსახორციელებლად. ეს საშუალებას გვაძლევს შეიქმნას საიმედო და მოქნილი სისტემა. თუ ხედვის სიმბოლო არ არის მითითებული, ჩვეულებრივ იგულისხმება, რომ თვისება არის გახსნილი.

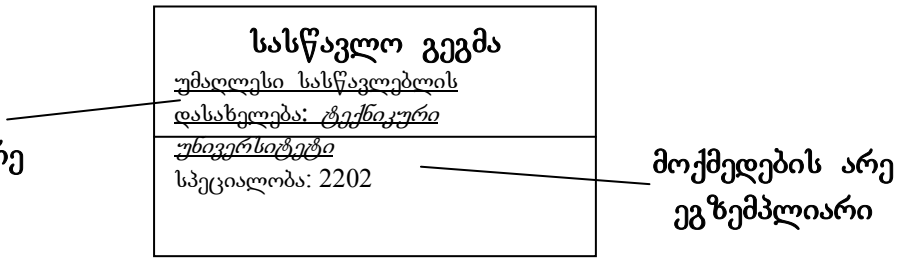


მოქმედების არე. კლასის ატრიბუტების და ოპერაციების კიდევ ერთ მნიშვნელოვან მახასიათებელს წარმოადგენს მოქმედების არე. რომელიმე თვისებისათვის მოქმედების არის მითითებით აღნიშნავენ, გამოავლენს თუ არა იგი თავის თავს სხვა სახით კლასის ყოველ ეგზემპლიარში, თუ თვისების ერთი და იმავე მნიშვნელობა ერთობლივად გამოიყენება ყველა ეგზემპლიარის მიერ. ობიექტ-ორიენტირებულ სისტემებში განსაზღვრულია მოქმედებათა არის ორი სახე:

- **instance** (ეგზემპლიარი) – კლასის ყოველ ეგზემპლიარს აქვს მოცემული თვისების საკუთარი მნიშვნელობა.
- **classifier** (კლასიფიკატორი) – კლასის ყველა ეგზემპლიარები ერთობლივად იყენებენ მოცემული თვისების საერთო მნიშვნელობას.

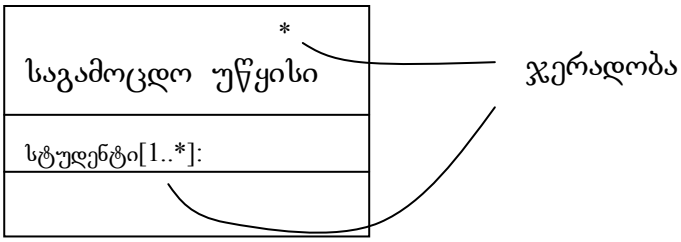
როგორც წესი, სამოდელო კლასების თვისებებს აქვთ მოქმედების არე ეგზემპლიარი. თვისებები მოქმედების არით კლასიფიკატორი ყველაზე ხშირად გამოიყენება დახურული ატრიბუტების აღწერისათვის.

ქვემოთ მოყვანილ ნახაზზე თვისება, რომელსაც აქვს მოქმედების არე **classifier** ხაზგასმულია, ხოლო თუ ხაზგასმული არ არის იგულისხმება მოქმედების არე **instance**.



ჯერადობა. კლასებთან მუშაობისას იგულისხმება, რომ შეიძლება არსებობდეს მისი ნებისმიერი რაოდენობის ეგზემპლიარები. თუ რასაკვირველია ის არ არის აბსტრაქტული კლასი, რომელსაც საერთოდ არ გააჩნია ეგზემპლიარები, თუმცა მის შვილობილებს შეიძლება გააჩნდეთ ნებისმიერი რაოდენობით.

კლასის ეგზემპლიარების რაოდენობას უწოდებენ მის ჯერადობას. კლასის ჯერადობა მიეთითება გამოსახულებით ზედა მარჯვენა კუთხეში. ჯერადობა გამოიყენება ატრიბუტების მიმართაც. ატრიბუტის ჯერადობა იწერება გამოსახულების სახით კვადრატულ ფრჩხილებში და განთავსდება ატრიბუტის დასახელების შემდეგ.



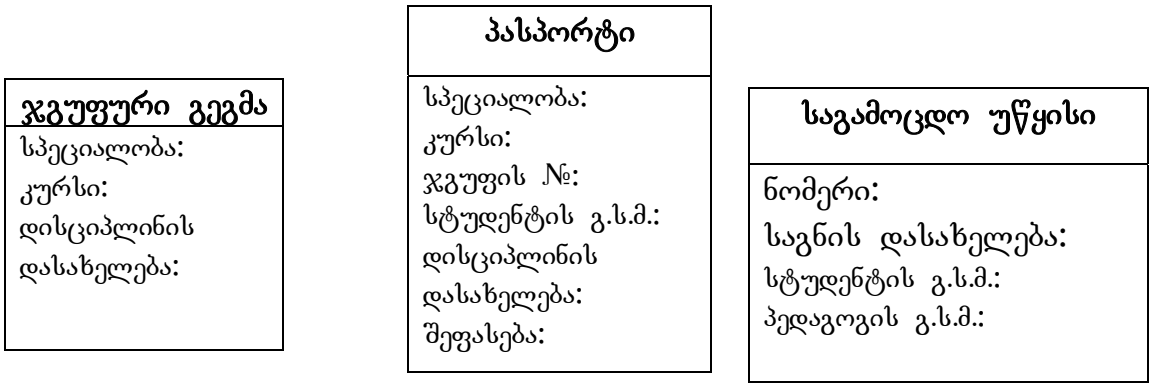
§ 3.1.1. კლასების დადგენა და მოვალეობების განაწილება

კლასების მეშვეობით ჩვეულებრივ გამოხატავენ აბსტრაქციებს, რომლებიც გამოიყოფა გადასაწყვეტი ამოცანიდან და გამოიყენება მის გადასაწყვეტად. ასეთი აბსტრაქციები წარმოადგენენ სისტემის ლექსიკონს ე.ი. წარმოადგენენ არსებს, რომლებიც მნიშვნელოვანია მომხმარებლებისა და დამმუშავებლისათვის. იმისათვის, რომ მომხმარებელმა სრულად გამოყოს ეს აბსტრაქციები მიმართავენ პრეცედენტების ანალიზს მათი რეალიზებიდან გამომდინარე.

პრეცედენტების რეალიზების მოდელირებისათვის ობიექტ-ორიენტირებლი მიდგომიდან გამომდინარე პირველ რიგში ახდენენ იმ სტრუქტურული ელემენტების იდენტიფიცირებას, რომლებიც შეადგენენ პრეცედენტის სემანტიკას. ამის შემდეგ განიხილავენ ცალკეულ სცენარებს, რომლებიც წარმოადგენენ მოცემულ პრეცედენტს. ამ სცენარების დინამიკას გამოსახავენ ამ სტრუქტურულ ელემენტებს შორის ურთიერთქმედებით, რისთვისაც სარგებლობენ მიმდევრობითობის და კოოპერაციის დიაგრამებით.

ასეთი სტრუქტურლი ელემენტების იდენტიფიცირების შემდეგ, სცენარებში მათი როლიდან გამომდიარე, ახდენენ მიღებულ ლექსიკონში შემავალ არსთა მოვალეობების განსაზღვრას. პრინციპში კლასის მოვალეობების რაოდენობა შეიძლება იყოს ნებისმიერი, მაგრამ კარგათ სტრუქტურირებულ კლასს აქვს სულ მცირე ერთი მოვალეობა. მოდელის დაზუსტებისას კლასის მოვალეობები გარდაიქმნებიან ატრიბუტებისა და ოპერაციების ერთობლიობაში, რომლებმაც უნდა უზრუნველყონ მათი შესრულება.

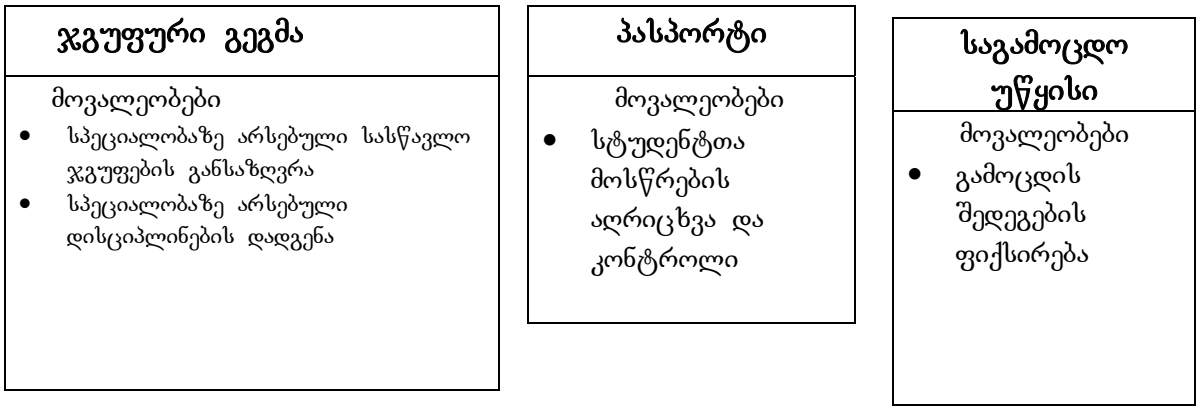
მაგალითად, უმაღლეს სასწავლებლებში სასწავლო პროცესის ორგანიზების სისტემისათვის გვექნება შემდეგი კლასები:



მოვალეობების განაწილების მოდელირებისათვის მოახდენენ რა სისტემაში იმ კლასების იდენტიფიცირებას, რომლებიც ერთობლივად აგებენ პასუხს გარკვეულ მოქმედებაზე, განსაზღვრავენ ყოველი კლასის მოვალეობას. მიღებულ კლასებს განიხილავენ როგორც ერთ მთლიანს, გამოყოფენ მათგან კლასებს, რომლებსაც

ძალიან ბევრი მოვალეობები აქვთ და დაყოფენ შედარებით მცირე კლასებათ – პირიქით, პატარა კლასებს ელემენტარული მოვალეობებით, გააერთიანებენ უფრო დიდში. ამრიგად, გადაანაწილებენ მოვალეობებს ისე, რომ ყოველი აბსტრაქცია გახდეს ავტონომიური. ბოლოს, განიხილავენ თუ როგორ კოოპერირებენ კლასები ერთმანეთთან და გადაანაწილებენ მოვალეობებს ისეთი ანგარიშით, რომ არც ერთი კლასი კოოპერაციის ჩარჩოში არ აკეთებდეს ძალიან ბევრს ან ძალიან ცოტას.

მაგალითისათვის მოვიყვანოთ მოვალეობების განაწილება ზემოთ მოყვანილი კლასებისათვის.



კლასების მოდელირებისას ხელმძღვანელობენ იმ მოსაზრებით, რომ ყოველ კლასს უნდა შეესაბამებოდეს გარკვეული არსი და კონცეპტუალური აბსტრაქცია იმ სფეროდან, რომელთანაც საქმე აქვს მომხმარებელს ან დამმუშავებელს. შეიცავდეს გარკვეული მოვალეობების ნაკრებს და ასრულებდეს ყოველ მათგანს. გასაგები და მარტივი იყოს, მაგრამ ამავე დროს შესაძლებელი იყოს მისი გაფართოება და ადაპტაცია.

3.2. მიმართებები

აბსტრაქციების აგებისას შესამჩნევია, რომ კლასები იშვიათად არსებობენ ავტონომიურად. როგორც წესი ისინი სხვადასხვა საშუალებით ურთიერთქმედებენ ერთმანეთთან. ეს იმას ნიშნავს, რომ სისტემის მოდელირებისას საჭიროა არა

მარტო არსთა იდენტიფიცირება, არამედ უნდა აღიწეროს თუ როგორ დამოკიდებულებაში არიან ერთმანეთთან. არსებობს მიმართებების სამი სახე, რომლებიც განსაკუთრებით მნიშვნელოვანია ობიექტ - ორიეტირებული მოდელირებისათვის:

- ***დამოკიდებულება***, რომლითაც აღიწერება კლასებს შორის არსებული გამოყენების მიმართებები.
- ***განზოგადება,*** რომლებიც აკავშირებენ განზოგადებულ კლასებს სპეციალიზირებულთან.
- ***ასოციაცია,*** რომლებიც აღწერენ ობიექტებს შორის სტრუქტურულ კავშირებს.

ყოველი მათგანი აბსტრაქციების სხვადასხვა სახით კომბინირების საშუალებას იძლევა.

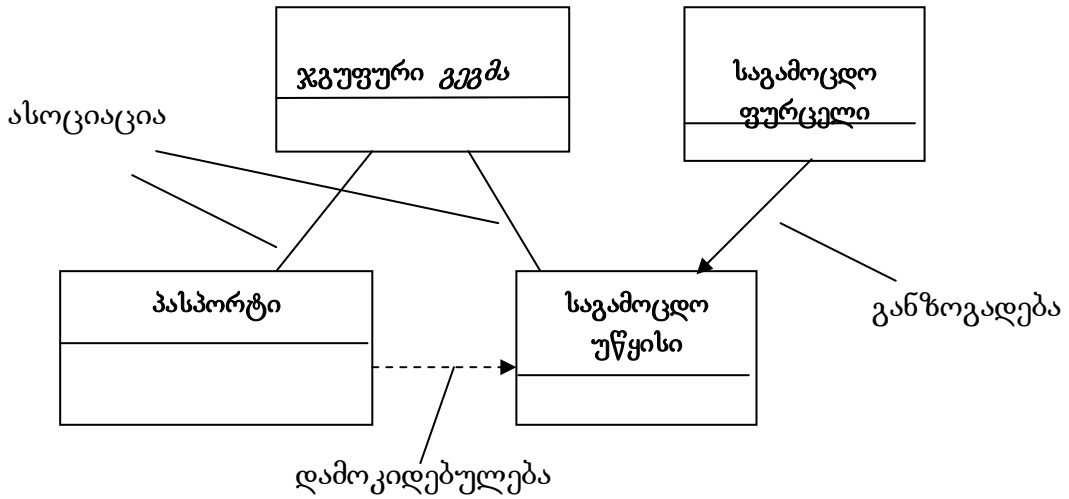
სასწავლო პროცესის ორგანიზაციისას ისეთი არსები როგორიც არის სასწავლო გეგმა, უწყისი, პასპორტი გახდება დასამუშავებელი სისტემის ლექსიკონის ნაწილი. მათგან არც ერთი არ არსებობს თავისთავად. უწყისები და პასპორტი ფორმირდება სასწავლო გეგმის საფუძველზე, რომლითაც უზრუნველყოფენ მოსწრების აღრიცხვას სტუდენტებისათვის. ვაერთიანებთ რა ამ ობიექტებს ვქმნით უფრო მაღალი დონის არსს – საგამოცდო სესია..

არსებს შორის შეიძლება აღმოვაჩინოთ არა მარტო სტრუქტურული მიმართებები. საგამოცდო სესიის ჩატარებისას აუცილებლად გვაქვს უწყისი, მაგრამ ის შეიძლება იყოს სხვადასხვა სახის. მაგ. ჯგუფური ან ინდივიდუალური, გარდა ამისა ზოგი განკუთვნილია მიმდინარე გამოცდისათვის, ზოგიერთი დავალიანების ლიკვიდაციისათვის. მიუხედავად ჩამოთვლილი განსხვავებებისა მათთვის დამახასიათებელია საერთო თვისება: ყოველი უწყისი განკუთვნილია სტუდენტის მოსწრების დასაფიქსირებლად.

საშუალებები, რომლებითაც ელემენტები უკავშირდებიან ერთმანეთს, მოდელირდებიან მიმართებების სახით. ეს სამი მიმართება მოიცავს ელემენტების

ურთიერთქმედების საშუალებების უდიდეს ნაწილს, რაც კარგად გამოისახება ობიექტებს შორის კავშირების საშუალებებით და მიღებულია დაპროგრამების ობიექტ- ორიენტირებულ ენებში.

თითოეული დასახელებული მიმართებისათვის არსებობს გრაფიკული გამოსახვა, რომელიც ნაჩვენებია ქვემოთ მოყვანილ მაგალითზე. ეს ნოტაცია საშუალებას გვაძლევს მოვახდინოთ სამოდელირო მიმართებების ვიზუალირება გამოყენებელი დაპროგრამების ენისაგან დამოუკიდებლად, ისე რომ ხაზი გაესვას მათ ყველაზე მნიშვნელოვან შემადგენლებს: სახელს, დამაკავშირებელ არსებს და თვისებებს.



დამოკიდებულება ეს არის გამოყენების მიმართება, რომლის მიხედვითაც ერთი ელემენტის სპეციფიკაციების ცვლილებას (მაგ. **საგამოცდო უწყისი**), შეიძლება გავლენა იქონიოს მეორე ელემენტზე, რომელიც მას იყენებს (მოცემულ შემთხვევაში კლასი **პასპორტი**), ამასთან პირიქით არ არის აუცილებელი.

ყველაზე ხშირად დამოკიდებულებას იყენებენ კლასებთან მუშაობისას, იმისათვის რომ გამოვსახოთ ოპერაციათა სიგნატურაში ის ფაქტი, რომ ერთი კლასი იყენებს მეორეს არგუმენტის სახით ე.ი. ერთ კლასში მომხდარი ცვლილება გამოისახება მეორეს მუშაობაზე. მაგალითად, პასპორტის მონაცემები დამოკიდებულია უწყისებისაგან, რომელიც აფიქსირებენ სტუდენტების შეფასებას ამა თუ იმ საგანში.

განზოგადება ეს მიმართებაა საერთო არსსა და მის კონკრეტულ გამოვლინებას შორის. მას კიდევ უწოდებენ მიმართებას "წარმოადგენს", იმის გამო რომ ერთი არსი უფრო ზოგადი წარმოადგენს მეორეს კერძო შემთხვევას. განზოგადება ნიშნავს იმას, რომ შვილობილი ობიექტი შეიძლება გამოყენებულ იქნას ყველგან, სადაც გვხვდება მშობელი კლასის ობიექტები, მაგრამ არა პირიქით. სხვა სიტყვებით რომ ვთქვათ შვილობილი შეიძლება დავაყენოთ მშობელის მაგიერ. ამასთან ის მემკვიდრეობით ღებულობს მშობელის თვისებებს, კერძოთ მის ატრიბუტებსა და ოპერაციებს. ხშირათ შვილობილს გააჩნია თავისი საკუთარი ატრიბუტები და ოპერაციები, გარდა იმისა რაც გააჩნია მშობელს. შვილობილი ღებულობს ოპერაციებს იმავე შემადგენლობით მშობლისაგან.

კლასს, რომელსაც არ გააჩნია მშობელი, მაგრამ აქვს შვილობილი უწოდებენ ბაზურს ან ფუძისეულს (ფესვს), ხოლო კლასს რომელსაც არ აქვს შვილობილი-ფოთლოვანს (leaf).

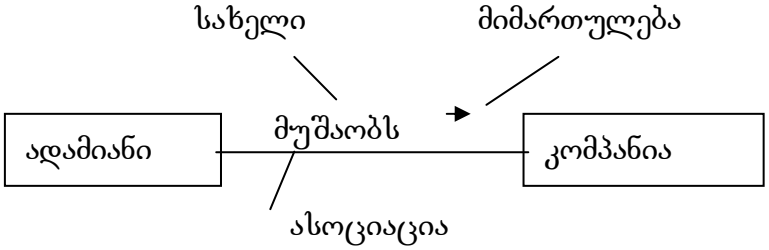
განზოგადების მიმართების მოდელირებისას თავიდან განსაზღვრავენ ატრიბუტებს, ოპერაციებს და მოვლენებს, რომლებიც საერთოა ორი ან მეტი კლასისათვის. ეს ელემენტები გამოიტანება საერთო კლასში, ახალი კლასის შესაქმნელად. ამასთან ყურადღება მისაქცევია იმაზე, რომ დონეები არ აღმოჩნდეს ძლიან ბევრი. იმისათვის, რომ მიუთითონ მოდელში შედარებით სპეციალიზებული კლასების მიერ მემკვიდრეობით აღებული თვისებები უფრო ზოგადისაგან, გამოიყენება განზოგადების მიმართება მიმართული შვილობილიდან მის მშობელზე.

ასოციაციას უწოდებენ სტრუქტურულ მიმართებას, რომელიც გვიჩვენებს, რომ ერთი ტიპის ობიექტები გარკვეულად დაკავშირებული არიან მეორე ტიპის ობიექტებთან. დასაშვებია შემთხვევა, როდესაც ასოციაციის ორივე ბოლო ერთი და იმავე კლასს ეკუთვნის. ეს ნიშნავს, რომ კლასის რომელიმე ობიექტზე დასაშვებია დაუკავშირდეს სხვა ობიექტები იმავე კლასიდან. ასოციაციას, რომელიც აკავშირებს ორ კლასს უწოდებენ ბინარულს. შესაძლებელია, თუმცა იშვიათია, შეიქმნას

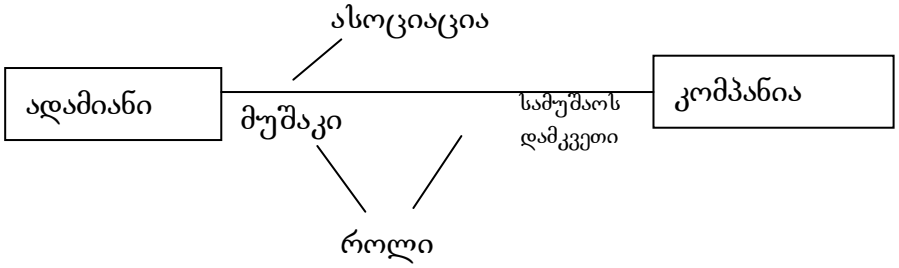
ასოციაცია, რომელიც აკავშირებს ერთდროულად რამოდენიმე კლასს, მათ უწოდებენ n- არულს.

არსებობს ოთხი დამატება, რომელიც გამოიყენება ასოციაციასთან მიმართებაში.

დასახელება. ასოციაციას შეიძლება მიენიჭოს სახელი და მისი კითხვის მიმართულება.

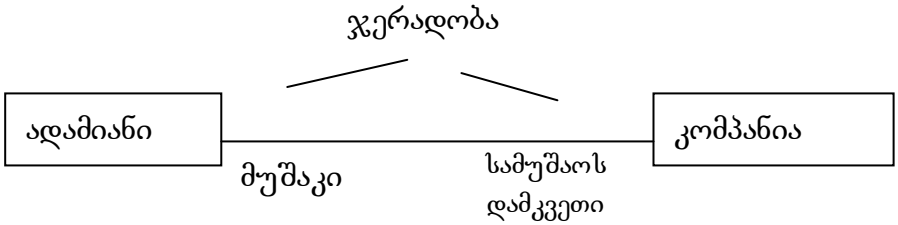


როლი. კლასი, რომელიც მონაწილეობს ასოციაციაში, აკისრია მასში გარკვეული როლი. შეგვიძლია მიუთითოთ ეს როლი.

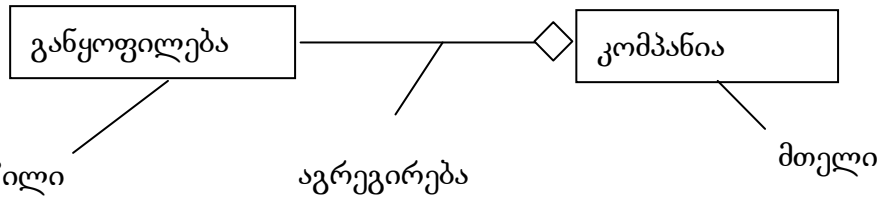


როლი – ეს გარკვეული არსების ქცევაა კონკრეტულ კონტექსტში ან სხვა სიტყვებით – პირი, რომლითაც აბსტრაქცია მიმართულია სამყაროს მიმართ.

ჯერადობა. ხშირათ მოდელირებისას საჭიროა მიეთითოს, რამდენი ობიექტი შეიძლება დაუკავშირდეს ასოციაციის ერთი ეგზემპლიარით. ამ რიცხვს უწოდებენ ასოციაციის როლის ჯერადობას. მას მიუთითებენ მნიშვნელობათა დიაპაზონის სახით. მაგ. ერთი (1), ერთი ან ნოლი (0..1), "ბევრი" (0..*), ერთი ან მეტი (1..*). შესაძლებელია მიეთითოს გარკვეული რიცხვი- (3). ქვემოთ მოყვანილი მაგალითიდან გამომდინარეობს, რომ ერთი ან მეტი მუშაკი მუშაობს ნებისმიერ (*) კომპანიაში.



აგრეგირება. უბრალო ასოციაცია ორ კლასს შორის გამოსახავს სტრუქტურულ მიმართებას თანაბარ არსებს შორის, როდესაც ორივე კლასი იმყოფება ერთ კონცეპტუალურ დონეზე და არც ერთი არ არის უფრო მნიშვნელოვანი მეორესთან შედარებით. მაგრამ ამას გარდა გვხვდება შემთხვევები, როდესაც საჭიროა მოდელირება მიმართებისა "ნაწილი/მთელი". ერთერთ კლასს აქვს მეტი რანგი(მთელი) და შედგება რამოდენიმე უფრო მცირე რანგის ნაწილებისაგან. ასეთი ტიპის მიმართებას უწოდებენ აგრეგირებას. გრაფიკულად მას გამოხატავენ რომბით.



გარდა ზემოთ მოყვანილი მიმართებებისა, ჩვენ გვხვდება კიდევ ორი სახის მიმართება:

- კავშირები – მიმართებები რომლის მეშვეობითაც შესაძლებელია შეტყობინებების გადაცემა(განხილული იყო ურთიერთქმედების განხილვისას);
- გადასვლები – რომლითაც აკავშირებენ მდგომარეობებს ავტომატში(განხილული იქნება შესაბამისი სტრუქტურული ელემენტის განხილვისას).

სტრუქტურული მიმართებების მოდელირებისას თავიდან განისაზღვრება ასოციაცია კლასების ყოველი წყვილისათვის. თუ ერთი კლასის ობიექტები ურთიერთქმედებდნენ სხვა კლასის ობიექტებთან განსხვავებულად, ვიდრე ეს ოპერაციის პარამეტრებით არის განსაზღვრული, მიზანშეწონილია განისაზღვროს ამ კლასებს შორის ასოციაცია. ყოველი განსაზღვრული ასოციაციისათვის მიუთითებენ ჯერადობას და როლების დასახელებას. თუ ასოციაციის ერთ-ერთი კლასი სტრუქტურულად ან ორგანიზაციულად წარმოადგენს მთელს ასოციაციის მეორე ბოლოში მყოფი კლასის მიმართ, ასეთი ასოციაცია მოინიშნება როგორც აგრეგირება.

§ 3.3. კლასების დიაგრამა

კლასების დიაგრამა ობიექტ - ორიენტირებული სისტემების მოდელირებისას გვხვდება ყველაზე ხშირად. ასეთ დიაგრამებზე უჩვენებენ კლასებს, ინტერფეისებს და მიმართებებს(დამოკიდებულების, განზოგადების, ასოციაციის) მათ შორის.

კლასების დიაგრამა ჩვეულებრივ გამოიყენება სისტემის ლექსიკონის და მონაცემთა ბაზის ლოგიკური სქემის მოდელირებისათვის.

სისტემის ლექსიკონის მოდელირება გულისხმობს გადაწყვეტილების მიღებას იმის შესახებ, თუ რომელი აბსტრაქციები არიან სისტემის ნაწილი, ხოლო რომელი არა. კლასების დიაგრამით საშუალება გვეძლევა განვსაზღვროთ ეს აბსტრაქციები და მათი მოვალეობები.

ლოგიკური სქემა შეიძლება წარმოვადგინოთ როგორც მონაცემთა ბაზის კონცეპტუალური პროექტის ნახაზი და მისი მოდელირებისათვის გამოიყენება კლასების დიაგრამა.

სამოდელო სისტემების დიდი ნაწილი შეიცავენ მდგრად ობიექტებს, ესე იგი ისეთებს, რომლებიც შესაძლებელია შევინახოთ მონაცემთა ბაზაში და შემდეგ საჭიროებისამებრ გამოვიძახოთ. ამისათვის ყველაზე ხშირად იყენებენ რელაციურ, ობიექტ-ორიენტირებულ ან ჰიბრიდულ ობიექტ-რელაციურ მონაცემთა ბაზებს. მონაცემთა ბაზების ლოგიკური პროექტირებისათვის ხშირად იყენებენ დიაგრამებს "არსი-კავშირი" (**E-R** დიაგრამები). მაგრამ თუ კლასიკურ **E-R** დიაგრამებზე ძირითადი ყურადღება გამახვილებულია მხოლოდ მონაცემებზე, კლასების დიგრამით შესაძლებელია ქცევის მოდელირებაც.

ლოგიკური სქემის მოდელირებისათვის თავიდან მოახდენენ მოდელში შემავალი კლასების იდენტიფიცირებას, რომელთა მდგომარეობაც უნდა შენახულ იქნას. გახსნიან კლასების სტრუქტურულ თავისებურებებს. ეს ნიშნავს, რომ დეტალურად მიეთითება ატრიბუტები და განსაკუთრებულ ყურადღებას აქცევენ ასოციაციას. ყოველი განსაზღვრული ასოციაციისათვის მიუთითებენ ჯერადობას და როლების დასახელებას. საჭიროების შემთხვევაში შეიქმნება შუალედური

აბსტრაქციები ლოგიკური სქემის გამარტივებისათვის. ამისათვის დადგინდება ატრიბუტები და ოპერაციები, რომლებიც საერთოა ორი ან მეტი კლასისათვის, გამოიტანება ეს ელემენტები საერთო კლასში, რათა შეიქმნას ახალი კლასი. განიხილება ამ კლასების ქცევები, ამისათვის დადგინდება ოპერაციები, რომლებიც მნიშვნელოვანია მონაცემებთან მიმართვისათვის და მათი მთლიანობის დასაცავად. ცდილობენ გამოიყენონ ინსტრუმენტალური საშუალებები, რომლებიც საშუალებას მისცემენ გარდაქმნან ლოგიკური პროექტი ფიზიკურში.

ნახ.3.1. მოყვანილია კლასების დიაგრამა, რომელიც შეესაბამება უმაღლეს სასწავლებლებში ამა თუ იმ სპეციალობის სასწავლო პროცესის ორგანიზებას. ყველა კლასი დიაგრამაზე, როგორც წინა შემთხვევაში, მონიშნულია როგორც მდგრადი(**persistent**), ესე იგი მათი ეგზემპლიარები უნდა იყვნენ მონაცემთა ბაზაში. მოყვანილია აგრეთვე კლასის ატრიბუტები. ოპერაციები, რომლებიც უზრუნველყოფენ მათზე მანიპულირებას. მონაცემთა მთლიანობის დაცვას და ვიზუალირებას, დიაგრამაზე არ არის ნაჩვენები.

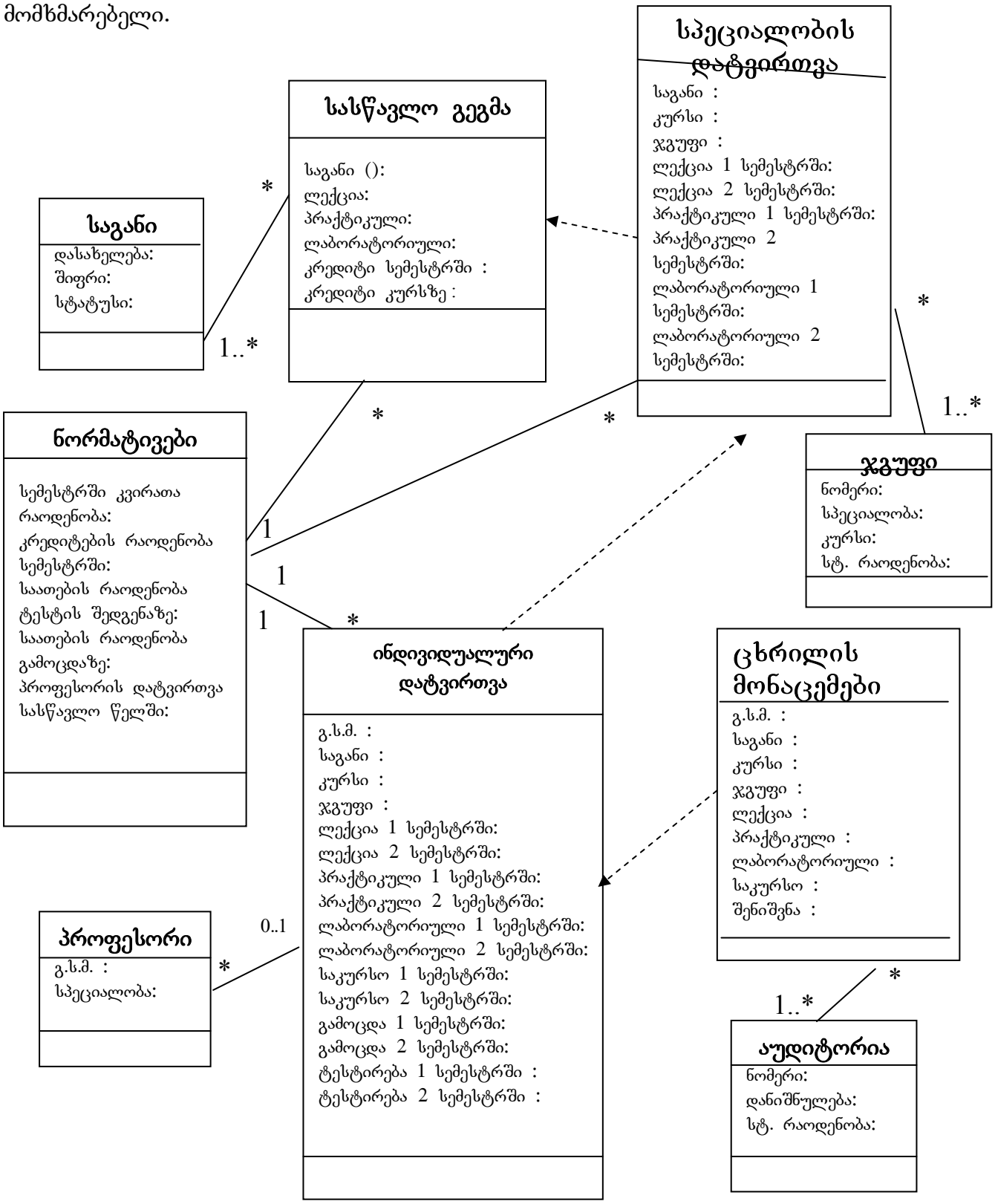
კლასებს შორის ***საგანი*** და ***სასწავლო გეგმა*** არსებობს ასოციაცია, რომელიც გვიჩვენებს, რომ ***სასწავლო გეგმა*** შეიცავს ***საგანს***. ნებისმიერი ***სასწავლო გეგმა*** შეიცავს საგნების ერთ ან ნებისმიერ რაოდენობას.

ანლოგიურად კლასებს ***პროფესორი*** და ***ინდივიდუალური დატვირთვა*** განსაზღვრულია ასოციაცია, რომელიც გვიჩვენებს, რომ ნებისმიერ პროფესორს უნდა გააჩნდეს ***ინდივიდუალური დატვირთვა***.

კლასებს ***სასწავლო გეგმა*** და კლასებს ***დატვირთვა*** და ***ინდივიდუალური დატვირთვა*** გვაქვს დამოკიდებულების მიმართება. რითაც ნაჩვენებია, რომ ყოველი ***დატვირთვა*** დგება ***სასწავლო გეგმის*** საფუძველზე, ხოლო ***ინდივიდუალური დატვირთვა*** თავის მხრივ ***დატვირთვის*** საფუძველზე.

სისტემის დაპროექტებისას მნიშვნელოვანია აიგოს სისტემები ამოცანათა მკაფიო გამიჯვნით. ეს გულისხმობს, რომ სისტემის განვითარებისას ცვლილებას მის ერთ ნაწილში არ შეეხოს დანარჩენს. ამ მიზნის მისაღწევათ აუცილებელია სისტემის დამაკავშირებელი კვანძების სპეციფიცირება, რომლებსაც უკავშირდებიან

დამოუკიდებლად ცვალებადი ნაწილები. დადგინდება რა შემდეგში უკეთესი რეალიზაცია, შესაძლებელი იქნება შეიცვალოს ძველი, ისე რომ არ შეწუხდეს მომხმარებელი.

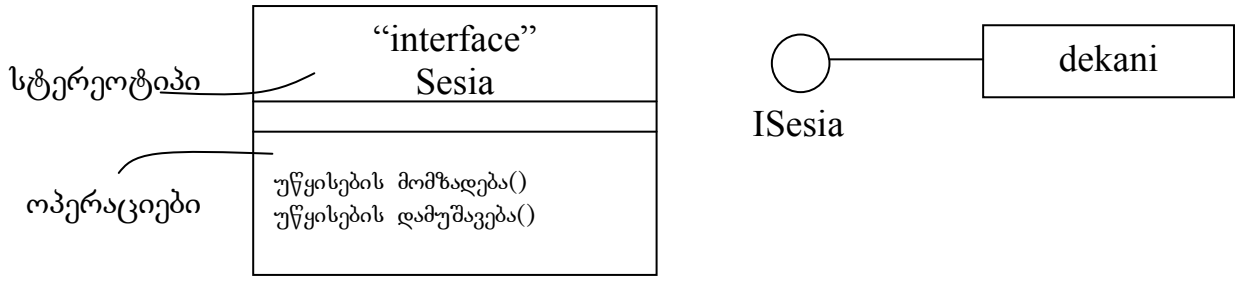


ნახ.3.1.

დამაკავშირებელი კვანძების მოდელირებისათვის გამოიყენება ინტერფეისები. **ინტერფეისები** – ეს ოპერაციების ერთობლიობაა, რომლებიც ახდენენ მომსახურების სპეციფიცირებას, რომელსაც წარმოადგენენ კლასები. ფაქტიურად ინტერფეისის გამოცხადებით დგინდება აბსტრაქციის სასურველი ქცევა, მათი რეალიზაციისაგან დამოუკიდებლად. კლიენტებს შეუძლიათ დაეყრდნონ გამოცხადებულ ინტერფეისებს, ხოლო დამპროექტებელს შეეძლება შექმნას ან იყიდოს მისი ნებისმიერი რეალიზაცია. მთავარია მხოლოდ მან შეასრულოს მოვალეობები, გამოცხადებული ინტერფეისით.

ამრიგად, ინტერფეისებს უწოდებენ ოპერაციების ნაკრებს, რომლებიც კლასების მიერ გამოიყენება მომსახურების სპეციფიცირებისათვის.

გრაფიკულად ინტერფეისი წარმოიდგინება წრის სახით. გარდა ამისა, იგი შეიძლება წარმოვადგენილ იქნას როგორც სტერეოტიპული კლასი, რათა გაიხსნას ოპერაციები და სხვა თვისებები.



ყოველ ინტერფეისს უნდა გააჩნდეს სხვებისაგან განსხვავებული სახელი. ინტერფეისის სახელი წარმოიდგინება ტექსტური სტრიქონის სახით. იმისათვის, რომ განასხვავონ კლასისაგან ინტერფეისის სახელს დასაწყისში ემატება I.

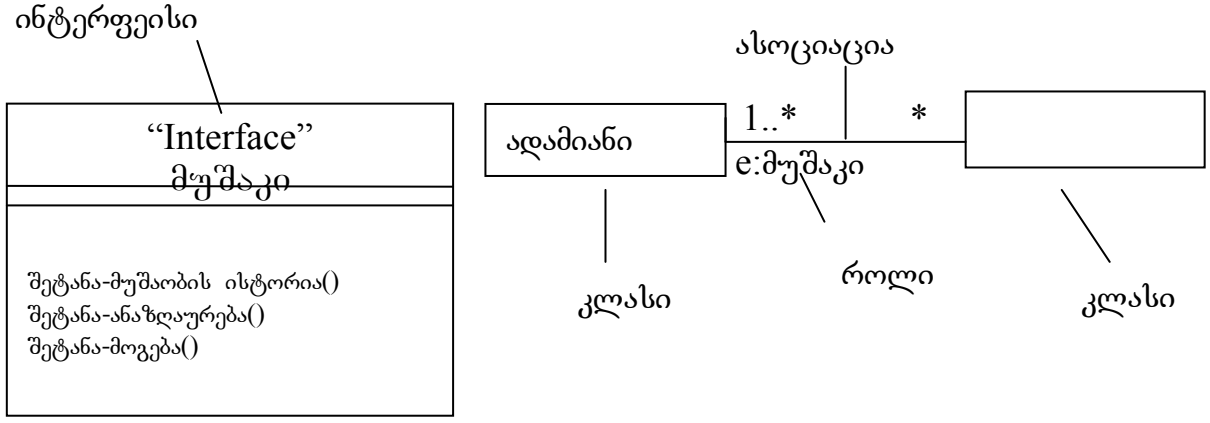
ინტერფეისი ახდენს კლასის სპეციფიცირებას, მაგრამ არ ახდენს არავითარ შეზღუდვას მის რეალიზაციაზე. კლასი შეიძლება რამოდენიმე ინტერფეისების რეალიზებას ახდენდეს. ამასთან ისინი მოვალეობას იღებენ შეასრულონ თავიანთი ყველა კონტრაქტები, უნდა შეიცავდნენ მეთოდებს ინტერფეისით გამოცხადებული ოპერაციების რეალიზებისათვის. ასევე კლასი შესაძლებელია დამოკიდებული იყოს რამოდენიმე ინტერფეისისაგან. ამ დროს ის ელოდება, რომ გამოცხადებული კონტრაქტები შესრულდებიან მათი რეალიზებადი გარკვეული კომპონენტების ნაკრებით. სწორედ ამიტომ ამბობენ, რომ ინტერფეისი წარმოადგენს სისტემაში

დამაკავშირებელ კვანძს. იგი განსაზღვრავს კონტრაქტის პირობებს, რომლითაც ორივე მხარეს კლიენტს და მომწოდებელს შეუძლიათ იმოქმედონ ერთმანეთისაგან დამოუკიდებლად, მთლიანად დაეყრდნონ რა ურთიერთ მოვალეობებს.

კლასი, რომელიც იყენებს ინტერფეისს, მასთან დაკავშირება ხდება დამოკიდებულების მიმართებით. იქმნება რა ახალი ინტერფეისი, პირველ რიგში უყურებენ ოპერაციათა სიმრავლეს, რომლებიც განსაზღვრავენ კლასის სერვისს. ინტერფეისი განსაზღვრავს კონტრაქტის პირობებს და კლასის ყველა ეგზემპლიარები უნდა იცავდნენ ამ პირობებს. ამასთან კონკრეტულ კონტექსტში ეგზემპლიარს შეუძლია წარმოადგინოს მხოლოდ ის ინტერფეისები, რომლებიც მოცემულ სიტუაციას შეესაბამება. ეს ნიშნავს, რომ ყოველი ინტერფეისი განსაზღვრავს როლს, რომელსაც თამაშობს ობიექტი. როლი – ეს გარკვეული არსის ქცევაა კონკრეტულ კონტექსტში ან სხვა სიტყვებით – პირი, რომლითაც აბსტრაქცია მიმართულია სამყაროს მიმართ. მაგალითად, განვიხილოთ კლასი ***ადამიანის*** ეგზემპლიარი. კონტექსტისგან დამოკიდებულებით ამ კლასის ეგზემპლიარი შესაძლებელია თამაშობდეს მუშაკის, მყიდველის, მენეჯერის, მომღერლის და ა.შ. როლს. შესაბამისად, ობიექტი წარუდგენს სამყაროს ამა თუ იმ "სახეს" და მასთან ურთიერთქმედებაში მყოფი კლიენტები ელოდებიან მისგან შესაბამის ქცევას. მაგალითად, კლასი ***ადამიანის*** ეგზემპლიარი მენეჯერის როლში ფლობს სხვა თვისებებს, ვიდრე მას ექნება მომღერლის როლში.

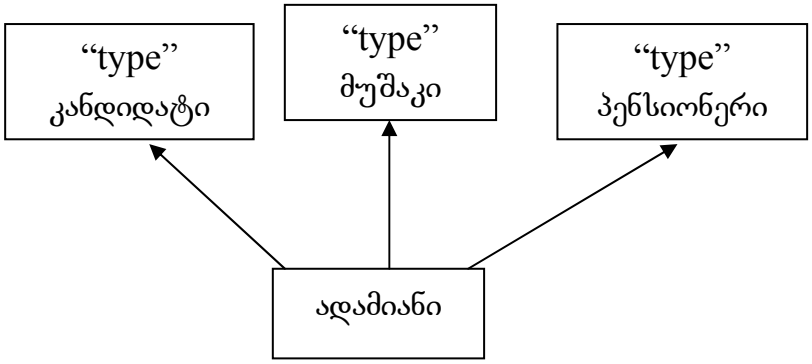
როლი, რომელსაც ერთი აბსტრაქცია თამაშობს მეორეს მიმართ, შეიძლება აღვწეროთ შევავსებთ რა ასოციაციის შესაბამის ბოლო წერტილს ინტერფეისის სახელით. მაგალითისათვის ნახაზზე ნაჩვენებია ინტერფეისი ***მუშაკი***, რომლის განსაზღვრა მოიცავს სამ ოპერაციას. კლასებს შორის **ადამიანი** და **კომპანია** არსებობს ასოციაცია, რომლის კონტექსტში **ადამიანი** თამაშობს **e** როლს, რომელიც ეკუთვნის ტიპს **მუშაკი**. სხვა ასოციაციაში ეს კლასი შესაძლებელია "სხვა სახით იყოს წარმოდგენილი". მაგრამ ამ შემთხვევაში, კლასი **ადამიანი** თამაშობს კლასისათვის **კომპანია** მუშაკის როლს და მოცემულ კონტექსტში

კომპანიისათვის ხილვადია და არსებითი იქნება მხოლოდ თვისებები, რომლებიც განისაზღვრებიან მოცემული როლით.



აბსტრაქციების სემანტიკის ფორმალური მოდელირებისათვის და მათი გარკვეულ ინტერფეისთან შესაბამისობისათვის გამოიყენება სტერეოტიპი **type**. ეს არის კლასის სტერეოტიპი, რომლის მეშვეობით განისაზღვრება ობიექტების მოქმედების არე ოპერაციებთან ერთად. ტიპის კონცეფცია მჭიდროდ არის დაკავშირებული ინტერფეისის კონცეფციასთან, მხოლოდ იმ განსხვავებით, რომ ტიპის აღწერა შესაძლებელია შეიცავდეს ატრიბუტებს, ხოლო ინტერფეისის აღწერა – არა.

ქვედა ნახაზზე მოყვანილია როლები, რომელსაც კლასი ***ადამიანი*** თამაშობს რესურსების მართვის სისტემის კონტექსტში. მოცემული ნახაზიდან ჩანს, რომ კლასი ***ადამიანის*** ეგზემპლიარები შესაძლებელია მივაკუთვნოთ სამიდან ერთ ტიპს – კანდიდატი, მუშაკი და პენსიონერი, რომლებიც წარმოიდგინებიან სტერეოტიპული კლასების სახით.



3.4. ობიექტების დიაგრამა

ობიექტების დიაგრამა საშუალებას იძლევა მოვახდინოთ იმ არსების ეგზემპლიართა მოდელირება, რომლებიც არსებობენ კლასების დიაგრამაზე. ობიექტების დიაგრამაზე ნაჩვენებია ობიექტების სიმრავლე და მათ შორის მიმართებები დროის რომელიღაც მომენტში. ობიექტების დიაგრამით ახდენენ ობიექტების სტრუქტურის მოდელირებას.

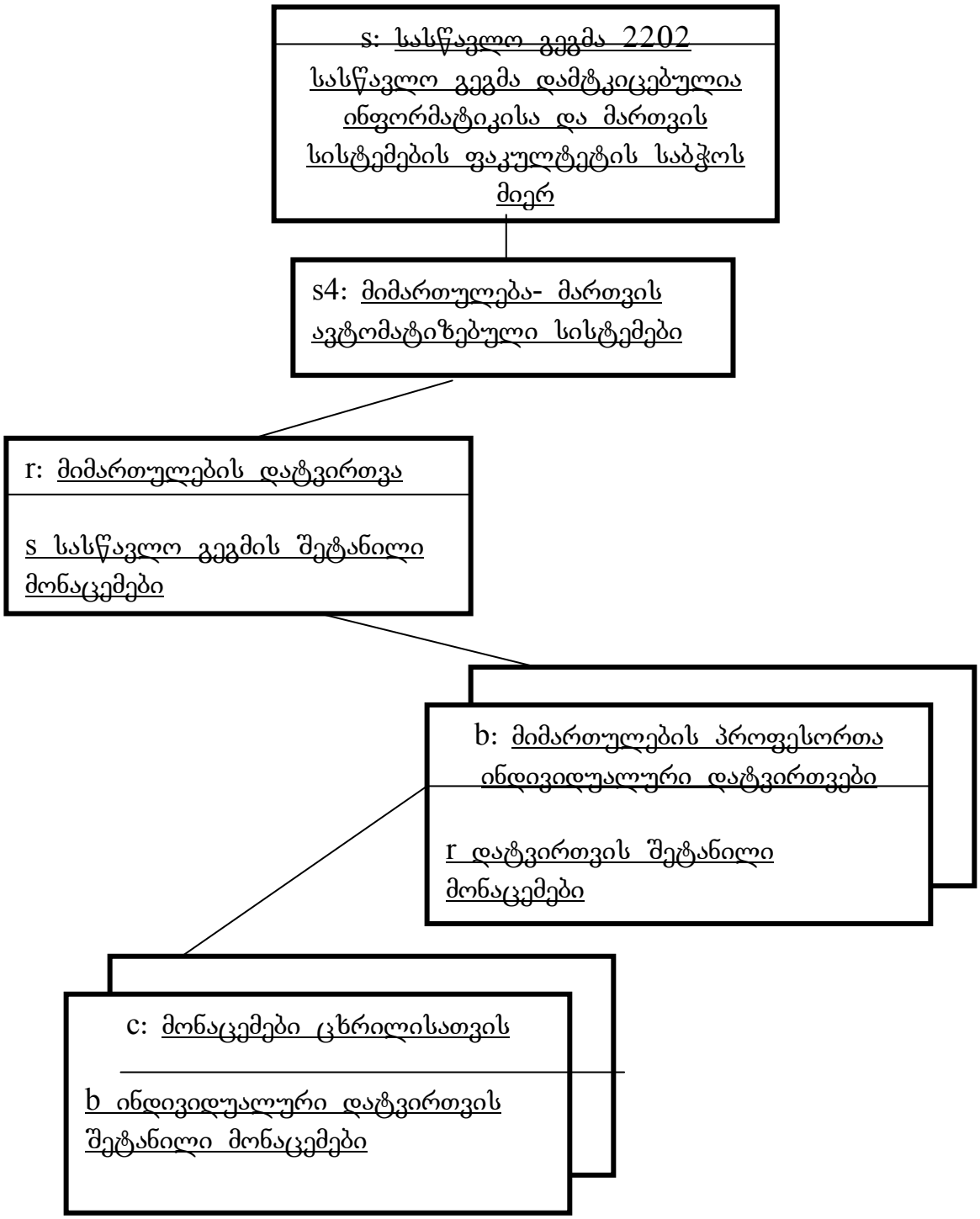
ობიექტთა სტრუქტურის მოდელირება გულისხმობს სისტემის ობიექტთა "სურათის" მიღებას დროის მოცემულ მომენტში. ობიექტების დიაგრამა წარმოადგენს დინამიური სცენარის სტატიკურ საფუძველს, რომელიც აღიწერება ურთიერთქმედების დიაგრამით.

ობიექტური სტრუქტურების მოდელირებისათვის თავიდან ახდენენ იმ მექანიზმების იდენტიფიცირებას, რომლის მოდელირებასაც აპირებენ. მექანიზმი წარმოადგენს გარკვეულ ფუნქციას ან სამოდელო სისტემის ქცევის ნაწილს, რომელშიც მონაწილეობენ კლასები და სხვა არსები.

ყოველი მექანიზმისათვის ახდენენ კლასების და კოოპერაციაში მონაწილე სხვა ელემენტების და მათ შორის მიმართებების იდენტიფიცირებას. განიხილავენ მექანიზმის მუშაობის ერთ-ერთ სცენარს. დააფიქსირებენ ამ სცენარს დროის გარკვეული მომენტისათვის და გამოსახავენ ყველა ობიექტებს, რომლებიც მონაწილეობენ მექანიზმში. მიუთითებენ ყოველი ობიექტის მდგომარეობას და ატრიბუტების მნიშვნელობებს, თუ ეს საჭიროა სცენარის გაგებისათვის. მიუთითებენ აგრეთვე კავშირებს ამ ობიექტებს შორის, რომლებიც წარმოადგენენ არსებული ასოციაციების ეგზემპლიარებს.

მაგალითისათვის ნახ.3.2.-ზე მოყვანილია ობიექტების ერთობლიობა, აღებული უმაღლეს სასწავლებლეში სასწავლო პროცესის ორგანიზაციისას. როგორც ნახაზიდან ჩანს, ერთი ობიექტი შეესაბამება სასწავლო გეგმას (s, კლასი სასწავლო გეგმის ეგზემპლიარი), რომელიც დამტკიცებულია ფაკულტეტის საბჭოს

მიერ. ეს ობიექტი დაკავშირებულია ეგზემპლიართან s4 კლასისა მიმართულება. თავის მხრივ ობიექტი s4 დაკავშირებულია ობიექტთან r დატვირთვა, r უკავშირდება b ინდივიდუალურ დატვირთვებს, ხოლო ეს უკანასკნელები c მონაცემებს ცხრილისათვის. აღნიშნული მონაცემები შესაძლებელია მიუთითოდ ურთიერთქმედების დიაგრამაზე (იხ. §2.3.).



ნახ. 3.2.

§ 3.5. მართვის რამდენიმე ნაკადის მოდელირება

რეალურ სამყაროში გარკველი მოვლენები სრულდებიან ერთდროულად. ამიტომ სისტემის მოდელირებისას უნდა გათვალისწინებულ იქნას მართვის რამოდენიმე ნაკადის პარალელური შესრულება.

თუ მიმდევრობით სისტემებში გვაქვს მხოლოდ ერთი მართვის ნაკადი და დროის ყოველ მომენტში სრულდება ერთი და მხოლოდ ერთი მოქმედება, პარალელურ სისტემაში მართვის ნაკადები რამდენიმეა, ე.ი. დროის ერთსა და იმავე მომენტში სრულდება სხვადასხვა მოქმედება. თითოეული, ერთდროულად შესრულებადი მართვის ნაკადებიდან, იწყება დამოუკიდებელ პროცესში ან ძაფში შესვლის წერტილიდან.

ობიექტ – ორიენტირებულ სისტემებში პროცესი (**Process**) – ეს რესურსებით უზრუნველყოფილი მართვის ნაკადია, რომელიც სრულდება სხვა პროცესების პარალელურად; ძაფი (**Thread**) – ეს შედარებით მცირე მართვის ნაკადია, რომელიც სრულდება სხვა ძაფების პარალელურად ერთ და იმავე პროცესის ფარგლებში.

პროცესის წარმოსადგენად, რომლის კონტექსტშიც სრულდება დამოუკიდებელი მართვის ნაკადი, მუშაობს სხვების პარალელურად და სარგებლობს მისი თანაბარი უფლებებით, გამოიყენება აქტიური კლასი.

აქტიური კლასი წარმოადგენს დამოუკიდებელ მართვის ნაკადს, მაშინ როდესაც ჩვეულებრივი კლასი არ არის მასთან კავშირში. აქტიურებისგან განსხვავებით, ჩვეულებრივ კლასებს უწოდებენ პასიურებს, რადგან მათ არ აქვთ საშუალება მოახდინოს დამოუკიდებელი მართვის ნაკადის ინიცირება.

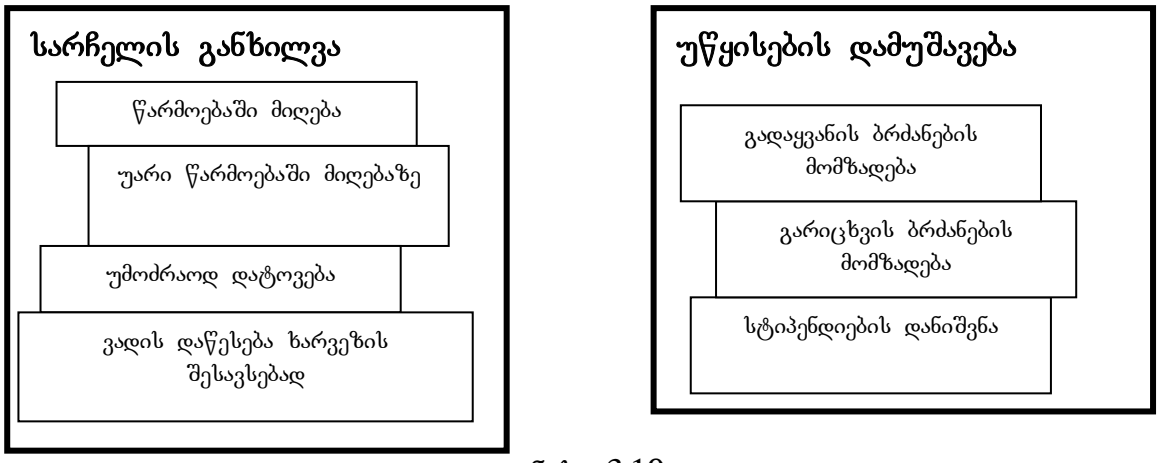
პროცესების მოდელირებისათვის გამოიყენება აქტიური კლასები. ტექნიკურად ეს ნიშნავს, რომ აქტიური ობიექტი – კლასის ეგზემპლარი ახდენს პროცესის მატერიალიზებას. პარალელური სისტემების მოდელირებისას აქტიური ობიექტებით

ჩვენ ვანიჭებთ სახელს ყოველ დამოუკიდებელ მართვის ნაკადს. როდესაც აქტიური ობიექტი იქმნება, გაიშვება მასთან ასოცირებული მართვის ნაკადი. როდესაც აქტიური ობიექტი ისპობა, ეს ნაკადი სრულდება.

აქტიური ობიექტი – ეს ობიექტია, რომელიც ფლობს პროცესს ან ძაფს და შეუძლია მმართველი ზემოქმედების ინიცირება. აქტიური კლასი – ეს კლასია, რომლის ეგზემპლარები წარმოადგენენ აქტიურ ობიექტებს.

აქტიური კლასები ფლობს იმავე თვისებებს, რასაც სხვა კლასები. მათ შესაძლებელია გააჩნდეს ეგზემპლარები, ატრიბუტები და ოპერაციები, ასევე მონაწილეობა მიიღონ დამოკიდებულების, განზოგადების და ასოციაციის მიმართებებში.

მაგალითისათვის 3.10. ნახაზზე მოყვანილია პროცესი და ძაფები, რომლებიც გვხვდება სამოქალაქო სამართალწარმოებაში და სასწავლო პროცესის ორგანიზაციისას უმაღლეს სასწავლებელში.



ნახ. 3.10.

პოცესების მატერიალიზაცია მოითხოვს პარალელურად შესრულებადი პროცესებისა და ძაფების დადგენას. როგორც ნახაზიდან ჩანს პროცესები *სარჩელის განხილვა, უწყისების დამუშავება* შეიცავს შედარებით მცირე ზომის პროცესებს (ძაფებს). თითოეული პროცესი (აქტიური კლასი) შეიძლება სრულდებოდეს ერთ კვანძზე ან სხვადასხვა კვანძებზე. თუ კვანძი შეიცავს რამდენიმე პროცესორს, მაშინ მასზე მიიღწევა ჭეშმარიტი პარალელიზმი. მაგრამ,

თუ გვაქვს მხოლოდ ერთი პროცესორი, მაშინ შესაძლებელია მხოლოდ პარალელიზმის ილუზია, რომელსაც უზრუნველყოფს ოპერაციული სისტემა.

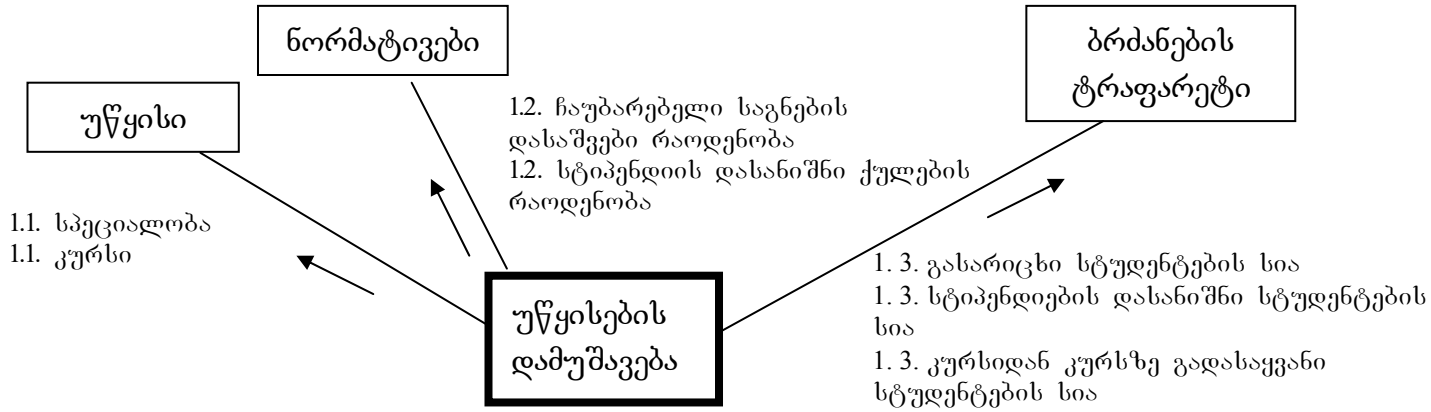
რეალურ პარალელიზმს შესაძლებელია მივაღწიოთ სამი საშუალებით: - გავანაწილოთ აქტიური ობიექტები სხვადასხვა კვანძებზე, - მოვათავსოთ აქტიური ობიექტები კვანძებზე რამდენიმე პროცესორებით და ბოლოს, მოვახდინოთ ორივე მეთოდის კომბინირება.

სისტემის აგებისას რამდენიმე მართვის ნაკადით უნდა გადაწყდეს, თუ როგორ გადავანაწილოთ სამუშაო პარალელურ აქტიურ ელემენტებს შორის, ასევე უნდა დადგინდეს კომუნიკაციისა და სინქრონიზაციის სწორი მექანიზმები სისტემის აქტიურ და პასიურ ობიექტებს შორის, რომლებიც უზრუნველყოფს მათი ქცევის სისწორეს მართვის რამდენიმე ნაკადის დროს.

მართვის რამდენიმე ნაკადის მოდელირებისას სარგებლობენ სამუშაო პროცესებისა და ოპერაციების აღწერით (მოღვაწეობის დიაგრამა), ამ დიაგრამებიდან სიძნელეს არ წარმოადგენს დადგინდეს მოქმედებათა პარალელიზმის შესაძლებლობა (იხ.§ 2.1.). მაგალითისათვის შეგვიძლია მოვიყვანოთ სამუშაო პროცესის აღწერა "უწყისების დამუშავება" (იხ.ნახ.2.9.), რომელზედაც კარგად ჩანს პარალელურად შესრულებადი ნაკადები (მოქმედებები, რომლებიც მოთავსებულია პარალელურ მსხვილ ხაზებს შორის) *გადაყვანის ბრძანების მომზადება, გარიცხვის ბრძანების მომზადება, სტიპენდიის დანიშვნის ბრძანების მომზადება.* ამის შემდეგ შესაძლებელია მართვის ნაკადის მატერიალიზაცია აქტიური კლასის სახით და შესაბამისად დაჯგუფდეს აქტიური ობიექტების საერთო სიმრავლე აქტიურ კლასში. ისევე როგორც ჩვეულებრივი კლასებისათვის განიხილავენ მოვალეობების განაწილების ბალანსს აქტიურ კლასებს შორის, ხოლო შემდეგ, რომელ სხვა აქტიურ და პასიურ კლასებთან კოოპერირდება სტატიურად ყოველი მათგანი. სტატიკური გადაწყვეტილებები შესაძლებელია გამოიხატოს კლასების დიაგრამის სახით. შემდეგ განიხილავენ, თუ როგორ კოოპერირდება დინამიკურად თითოეული კლასი სხვა კლასებთან. ამ გადაწყვეტილებებს

გამოხატავენ ურთიერთქმედების დიაგრამაზე. ჩვენი მაგალითისათვის გამოვყოფთ აქტიურ კლასს ‘‘უწყისების დამუშავება’’, რომლის მოვალეობაა სტუდენტთა მოსწრების გამოვლენა. ამ მოვალეობის უზრუნველსაყოფად იგი ურთიერთქმედებს კლასებთან ***უწყისი, ნორმატივები, ბრძანების ტრაფარეტი.*** ობიექტი ***უწყისის*** მეშვეობით აქტიური კლასი ადგენს გასარიცხი სტუდენტების სიას, სტიპენდიების დასანიშნი სტუდენტების სიას, კურსიდან კურსზე გადასაყვანი სტუდენტების სიას, ობიექტი ***ნორმატივებიდან*** მიღებული მონაცემების საფუძველზე. ობიექტით ***ბრძანების ტრაფარეტი*** აქტიური კლასი ადგენს შეტყობინების შესაბამის ბრძანების ტრაფარეტს.

3.11. ნახაზზე მოყვანილია კოოპერაციის დიაგრამა, რომელზეც ნათლად არის მითითებული აქტიური ობიექტი როგორც საწყისი წერტილი შესაბამისი მართვის ნაკადისა.



ნახ.3.11.

განსაკუთრებულ ყურადღებას მოითხოვს აქტიურ ობიექტებს შორის კომუნიკაციების დადგენა, რომელზედაც დამოკიდებულია სისტემის ეფექტური ფუნქციონირება.

§ 3.5.1. პროცესებს შორის კომუნიკაცია და მართვის ნაკადების სინქრონიზაცია

ერთმანეთთან კოოპერირებადი ობიექტები ურთიერთქმედებს შეტყობინებების გაცვლით. სისტემებში სადაც არის აქტიური და პასიური ობიექტები, განიხილავენ ოთხ შესაძლო კომბინაციას.

პირველი, შეტყობინება შესაძლებელია გადაიცეს ერთი პასიური ობიექტიდან მეორეზე. იმის გათვალისწინებით, რომ დროის ნებისმიერ მომენტში არსებობს მხოლოდ ერთი მართვის ნაკადი, რომელიც გადის ორივე ობიექტზე, ასეთი ურთიერთქმედება შეესაბამება უბრალოდ ოპერაციის გამოძახებას.

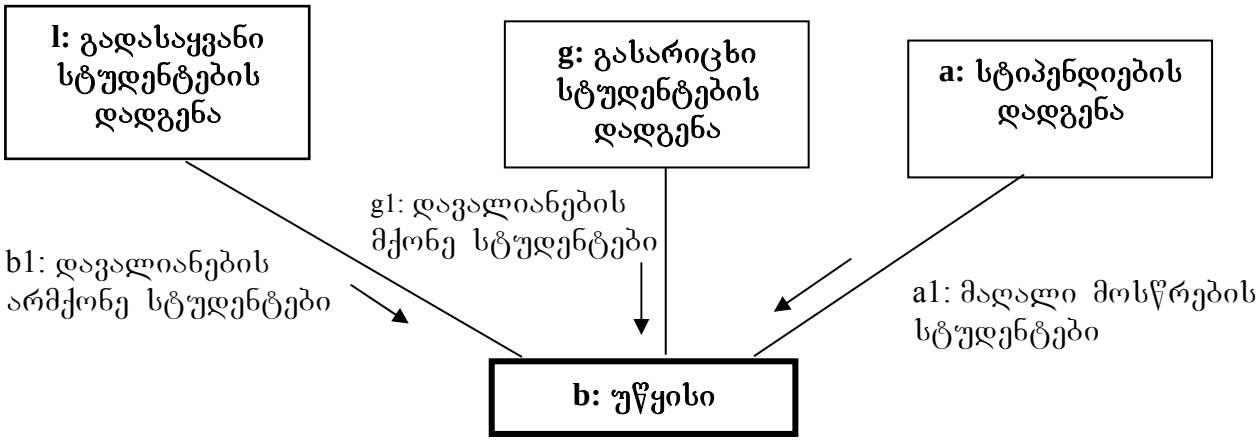
მეორე, შეტყობინება შესაძლებელია გადაიცეს ერთი აქტიური ობიექტიდან მეორეზე. ამ შემთხვევაში ხდება პროცესებს შორის კომუნიკაცია, რომელიც შესაძლებელია განხორციელდეს ორი საშუალებით. პირველ ვარიანტში რომელიმე აქტიურ ობიექტს შეუძლია სინქრონულად გამოიძახოს მეორეს ოპერაცია. ასეთ საშუალებას აქვს სემანტიკა რანდევუ (**Rendezvous**): გამომძახებელი ობიექტი მოითხოვს ოპერაციის შესრულებას და ელოდება, სანამ მიმღები მხარე მიიღებს გამოძახებას, შეასრულებს ოპერაციას და დააბრუნებს გარკვეულ ობიექტს (თუ ასეთი არის); შემდეგ ორივე ობიექტი აგრძელებს მუშაობას ერთმანეთისაგან დამოუკიდებლად. გამოძახების შესრულების მთელი დროის განმავლობაში ორივე მართვის ნაკადი ბლოკირებულია.

მეორე ვარიანტში ერთ აქტიურ ობიექტს შეუძლია ასინქრონულად გაუგზავნოს სიგნალი მეორეს ან გამოიძახოს მისი ოპერაცია. ასეთი საშუალების სემანტიკა შეესაბამება საფოსტო ყუთს (**Mailbox**). გამომძახებელი მხარე აგზავნის სიგნალს ან იძახებს ოპერაციას, რის შემდეგ აგრძელებს შესრულებას. ამ დროს მიმღები მხარე იღებს სიგნალს ან გამოძახებას, როგორც კი იქნება ამისათვის მზად. სანამ ის ამუშავებს მოთხოვნას, ყველა ახლად შემოსული მოვლენები ან გამოძახებები დგება რიგში. მოახდენს რა რეაგირებას მოთხოვნაზე, მიმღები

ობიექტი განაგრძობს თავის მუშაობას. საფოსტო ყუთის სემანტიკა მჟღავნდება იმაში, რომ ორივე ობიექტი არა სინქრონიზებულია, უბრალოდ ერთი უტოვებს შეტყობინებას მეორეს. გრაფიკულად სინქრონული შეტყობინება გამოისახება მთლიანი ისრით, ხოლო ასინქრონული "ნახევარისრით".

3.12. ნახაზზე მოყვანილია უმაღლეს სასწავლებლებში სასწავლო პროცესის ორგანიზება პროცესების თვალთახედვით და მათ შორის კომუნიკაცია.

როგორც ნახაზიდან ჩანს რამდენიმე ობიექტი ერთდროულად ურთიერთქმედებს ობიექტთან *უწყისი,* რომელსაც მივეცით სახელი c. შესაბამისად, c უნდა დავაპროექტოდ ისე, რომ მან შეინარჩუნოს თავისი სემანტიკა რამდენიმე მართვის ნაკადისას.



ნახ.3.12.

მესამე, შეტყობინება შესაძლებელია გადაიცეს აქტიური ობიექტიდან პასიურზე. სიძნელე წარმოიშვება იმ შემთხვევაში, როდესაც ერთდროულად რამდენიმე აქტიური ობიექტი გადასცემს თავის მართვის ნაკადს ერთ და იმავე პასიურს. ასეთ შემთხვევაში საჭიროა ნაკადების სინქრონიზაციის ძალიან ფრთხილი მოდელირება, რომელიც განხილული იქნება ქვევით.

მეოთხე, პასიურ ობიექტს შეუძლია გადასცეს შეტყობინება აქტიურს. გასათვალისწინებელია ის, რომ ყოველი მართვის ნაკადი ეკუთვნის რომელიმე აქტიურ ობიექტს, შესაბამისად ხდება ნათელი, რომ პასიური ობიექტის მიერ

შეტყობინების გადაცემას აქტიურზე აქვს იგივე სემანტიკა, რაც შეტყობინებების გაცვლა ორ აქტიურ ობიექტს შორის.

სისტემაში რამდენიმე მართვის ნაკადის ჩართვისას საჭიროა განვიხილოთ მექანიზმები, რომელთა მეშვეობით ობიექტები სხვადასხვა მართვის ნაკადიდან ურთიერთქმედებს ერთმანეთთან.

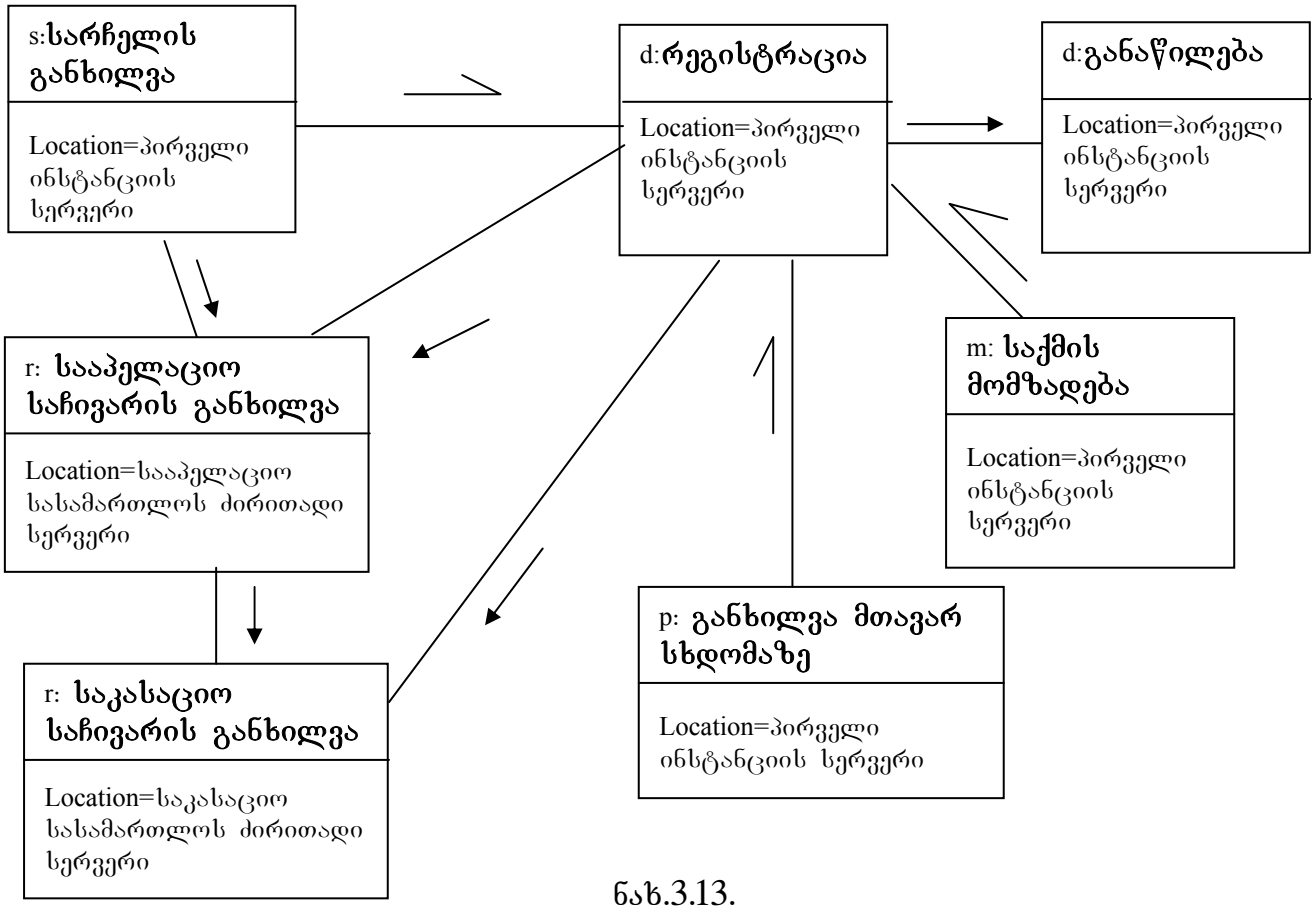
სირთულე პროცესებს შორის კომუნიკაციისა გამოწვეულია იმით, რომ განაწილებულ სისტემებში პროცესები შეიძლება სრულდებოდეს სხვადასხვა კვანძებზე. არსებობს ორი კლასიკური მიდგომა პროცესებს შორის კომუნიკაციისა: შეტყობინების გადაცემა და დაშორებული პროცედურების გამოძახება. ეს მექანიზმები მოდელირდება როგორც შესაბამისად სინქრონული და ასინქრონული მოვლენები.

მაგალითისათვის 3.13. ნახაზზე მოყვანილია სამოქალაქო სამართალწარმოების განაწილებული სისტემის ნაწილი, რომელშიც პროცესები სრულდება სამ კვანძზე. როგორც ნახაზიდან ჩანს კომუნიკაცია ობიექტებს შორის *სარჩელის განხილვა, საქმის მომზადება, განხილვა მთავარ სხდომაზე* და *სამოქალაქო საქმე* ასინქრონულია, ხოლო *სამოქალაქო საქმე, სააპელაციო საჩივრის განხილვა* და *საკასაციო საჩივრის განხილვა* სინქრონული.

ამრიგად, პროცესებს შორის კომუნიკაციის მოდელირებისას გამოდიან იმ მოსაზრებიდან, რომ შეტყობინებების გაცვლის მოდელირება უნდა მოხდეს ასინქრონული, ხოლო დაშორებული პროცედურების – სინქრონული კომუნიკაციის მეშვეობით.

სიძნელე პროცესებს შორის კომუნიკაციისა წარმოიშვება იმ შემთხვევაში, როდესაც ერთდროულად რამდენიმე აქტიური ობიექტი გადასცემს თავის მართვის ნაკადს ერთ და იმავე პასიურს. ერთ ოპერაციაში (შესაბამისად ერთ ობიექტში) შესაძლებელია ერთდროულად იმყოფებოდეს რამდენიმე მართვის ნაკადები, ასევე, შესაძლებელია, რომ სხვადასხვა ნაკადი იმყოფებოდეს სხვადასხვა ოპერაციაში, მაგრამ ერთ ობიექტში. სიფრთხილის გამოუჩენლობის შემთხვევაში, ნაკადებმა

შეიძლება ხელი შეუშალოს ერთმანეთს, რაც გამოიწვევს ობიექტის მდგომარეობის არაკორექტულ შეცვლას. ეს არის ურთიერთგამორიცხვის კლასიკური პრობლემა.



ნახ.3.13.

ამ პრობლემის გადასაწყვეტად ობიექტ-ორიენტირებულ სისტემებში ოპერაციებს, რომლებიც განსაზღვრულია კლასში, ენიჭება გარკვეული მასინქრონიზებელი თვისებები.

Sequential(მიმდევრობითი) – გამომძახებელი მხარე თავისი მოქმედების შესახებ კოორდინირებას უნდა ახდენდეს გამოსაძახებელ ობიექტში შესვლამდე, ისე რომ დროის ნებისმიერ მომენტში ობიექტის შიგნით იმყოფება ერთი მართვის ნაკადი. რამდენიმე მართვის ნაკადის არსებობისას ობიექტის სემანტიკისა და მთლიანობის გარანტია არ არის.

quarded(დაცული) - მართვის რამდენიმე ნაკადისას ობიექტის სემანტიკა და მთლიანობა გარანტირებულია ობიექტის ყველა დაცულ ოპერაციაზე გამოძახებათა

მოწესრიგების გზით. დროის ყოველ მომენტში ობიექტზე შესაძლებელია შესრულდეს მხოლოდ ერთი ოპერაცია.

Concurent(პარალელური) – მართვის რამდენიმე ნაკადისას ობიექტის სემანტიკა და მთლიანობა გარანტირებულია იმით, რომ ოპერაცია განიხილება როგორც ატომური.

§ 3.6. ობიექტების სასიცოცხლო ციკლების მოდელირება

სისტემის დინამიური ასპექტების ასახვის აქობამდე განხილული მოდელებით ხდებოდა ერთობლივად მომუშავე ობიექტების ქცევის მოდელირება. მაგრამ სისტემის დაპროექტებისას ხშირად საჭიროა ცალკეული ობიექტის ქცევის მოდელირება.

სხვადასხვა ობიექტების ქცევაზე დაკვირვებამ აჩვენა, რომ:

- მრავალი ობიექტი თავიანთი სიცოცხლის განმავლობაში გაივლიან გარკვეულ სტადიებს.
- განვითარების თანმიმდევრობა სტადიებზე სწორედ განაპირობებს მოცემული ობიექტისათვის დამახასიათებელი თვისებების ფორმირებას.
- დროის ნებისმიერ მომენტში ობიექტი იმყოფება ერთ გარკვეულ მდგომარეობაში.
- ობიექტები ერთი სტადიიდან მეორეზე გადადიან ნახტომისებურად.
- არსებობენ ინციდენტები, რომლებიც აიძულებენ ობიექტებს გადავიდნენ სტადიებს შორის.

შეიძლება დავასკვნათ, რომ ობიექტებს ჩვეულებრივ გააჩნიათ სიცოცხლის ხანგრძლივობა - სასიცოცხლო ციკლი. გარკვეული ობიექტები ჩნდებიან, განიცდიან ევოლუციას თავიანთი არსებობის გარკვეულ სტადიებზე და შემდეგ კვდებიან ან ქრებიან.

ძირითადათ ამ ობიექტებს მიეკუთვნებიან ობიექტები, რომელთა ქცევა ყველაზე კარგად გამოიხატება მათი რეაქციით საკუთარი კონტექსტის გარეთ მომხდარ მოვლენებზე და მათ უწოდებენ რეაქციულს.

რეაქციული ობიექტების მოდელირება გულისხმობს მთელი მათი სასიცოცხლო ციკლის მოდელირებას, დაწყებული შექმნის მომენტიდან და დამთავრებული მისი მოსპობით, განსაკუთრებული აქცენტით მდგრად მდგომარეობებზე, რომლებშიც შეიძლება იმყოფებოდეს ობიექტი. ამისათვის შესაძლებელია ავტომატების

გამოყენება. ავტომატი აღწერს ქცევას მიმდევრობითი მდგომარეობების სახით, რომლებზედაც გაივლის ობიექტი თავისი სიცოცხლის განმავლობაში.

ავტომატებს იყენებენ მოდელის ნებისმიერი ელემენტის ქცევის მოდელირებისათვის, მაგრამ ყველაზე ხშირად – კლასების, პრეცედენტების ან სისტემის მთლიანად.

ავტომატის ვიზუალირება შესაძლებელია ორი საშუალებით:

- გამოიყოს მართვის ნაკადის გადაცემა ერთი მოღვაწეობიდან მეორეზე (მოღვაწეობის დიაგრამის გამოყენებით);
- გამოიყოს ობიექტების პოტენციალური მდგომარეობები და გადასვლები მათ შორის (მდგომარეობათა დიაგრამის სახით).

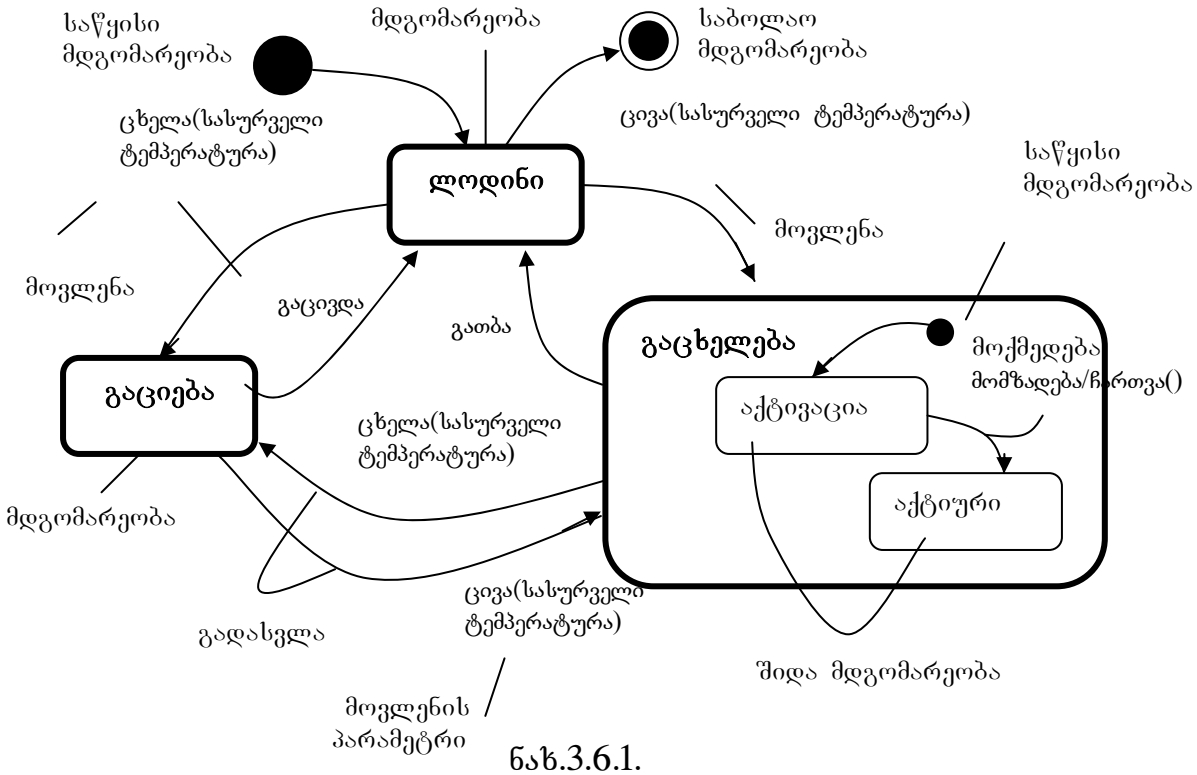
მდგომარეობათა დიაგრამა შედგენილია ელემენტებისაგან, რომელიც გააჩნია ნებისმიერ ავტომატს და მათზე გამოიყენება ავტომატის ყველა მახასიათებლები. ობიექტების მოდელირებისას მდგომარეობათა დიაგრამებით უნდა დადგინდეს ძირითადათ სამი ელემენტი:

- მდგრადი მდგომარეობები, რომელშიც შესაძლებელია იმყოფებოდეს ობიექტი;
- მოვლენები, რომლებიც ახდენენ გადასვლების ინიციირებას;
- მოქმედებები, რომლებიც სრულდება მდგომარეობის ყოველი შეცვლისას.

არსებობს საშუალებები მდგომარეობების, გადასვლების, მოვლენების და მოქმედებების გრაფიკული წარმოდგენისათვის, როგორც ეს მოყვანილია ნახ.3.6.1.-ზე. ეს საშუალებას იძლევა მოხდეს ობიექტის ქცევის ვიზუალირება ისე, რომ წარმოდგენილი იქნას მისი სასიცოცხლო ციკლის ყველაზე მნიშვნელოვანი ელემენტები.

მაგალითად, გამათბობელი სახლში შეიძლება იმყოფებოდეს სამიდან ერთერთ მდგომარეობაში: **ლოდინი** (ელოდება ბრძანებას "დავიწყო გაცხელება"); გაცხელება ორი ქვემდგომარეობით - **აქტივაცია** (გაზი მიეწოდება, მაგრამ არ არის მიღწეული საჭირო ტემპერატურა); **აქტიური** (გაზი ჩართულია); **გაცივება** (გაზი არ მიეწოდება). გამათბობელი შეიძლება იმყოფებოდეს მდგომარეობაში **გაცხელება,**

რომლის შიგნით არის კიდევ ერთი მდგომარეობა – **აქტივაცია**. ასეთ შემთხვევაში იტყვიან, რომ ობიექტი იმყოფება ერთდროულად მდგომარეობაში **გაცხელება** და **აქტივაცია**.



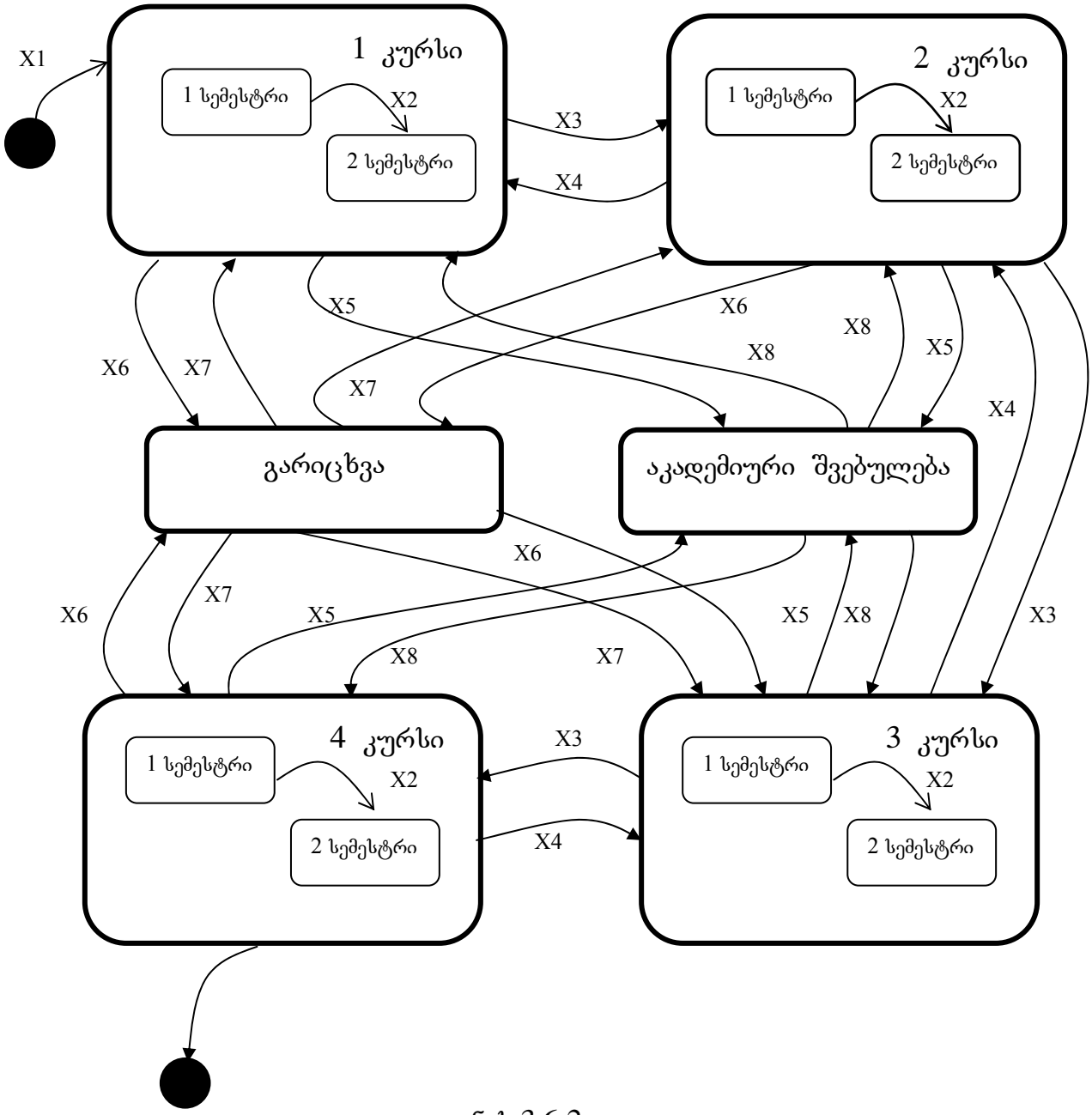
ნახ.3.6.1.

ასევე, უმაღლეს სასწავლებლებში სასწავლო პროცესის განხილვისას მნიშვნელოვანია იცოდნენ სტუდენტის მდგომარეობა, რომელსაც გააჩნია მკაფიოდ გამოხატული სასიცოცხლო ციკლი. სტუდენტი სწავლის პერიოდში გაივლის გარკვეულ მდგომარეობებს დაწყებული უმაღლეს სასწავლებელში შესვლიდან და დამთავრებული ბაკალავრის აკადემიური ხარისხის მინიჭებით. ასეთი ობიექტების ქცევის მოდელირებისათვის მნიშვნელოვანია ობიექტის – ***სტუდენტის*** სასიცოცხლო ციკლის მოდელირება მდგომარეობათა დიაგრამის გამოყენებით(ნახ.3.6.2.).

სადაც,

x1 - წარმატებით ჩააბარა ეროვნული გამოცდები;

x2 - ჩააბარა სემესტრით გათვალისწინებული საგნები (ან დარჩა დასაშვები რაოდენობით);



ნახ.3.6.2.

x3- ჩააბარა კურსით გათვალისწინებული საგნები (ან დარჩა დასაშვები რაოდენობით);

x4 - დავალიანება არ იქნა ლიკვიდირებული (დარჩა დასაშვებზე მეტი);

x5- წარმოადგინა ავადმყოფობის ცნობა;

x6- დავალიანება არ იქნა ლიკვიდირებული (ლიკვიდაციის ვადა ამოიწურა);

x7- აღდგენილ იქნა;

x8- დაბრუნდა აკადემიური შვებულებიდან.

§ 3.6.1. ობიექტის მდგომარეობები

ობიექტის მდგომარეობა ეს სიტუაციაა მის ცხოვრებაში, რომლის განმავლობაშიც იგი აკმაყოფილებს გარკვეულ პირობებს, ახორციელებს გარკვეულ მოღვაწეობას ან ელოდება სხვა მოვლენას.

მდგომარეობას განსაზღვრავენ შემდეგი ელემენტები:

- ***დასახელება*** (ტექსტური სტრიქონი, რომელიც განასხვავებს ერთ მდგომარეობას ყველა დანარჩენისაგან);
- ***მოქმედება შესვლა/გამოსვლის*** *დროს* (მოქმედებები, რომლებიც სრულდება მდგომარეობაში შესვლისას და მისგან გამოსვლისას);

მოდელირებისას ხშირად იქმნება სიტუაციები, როდესაც აუცილებელია გარკვეული მოქმედების შესრულება დამოუკიდებლად იმისა რომელი გადასვლით იქნა შესრულებული შესვლა მდგომარეობაში. ასევე მდგომარეობიდან გამოსვლისას, საჭირო ხდება ერთი და იგივე მოქმედების შესრულება ნებისმიერი გადასვლისას. მოყვანილი ეფექტის მიღწევა შესაძლებელია უბრალო ავტომატების გამოყენებითაც, მოცემული მოქმედებების ასოციირებით ყოველ შემავალ და გამომავალ გადასვლებთან შესაბამისად. მაგრამ ასეთი მეთოდიკის დროს არ არის გამორიცხული შეცდომა – მთავარია არ იქნას დავიწყებული ეს მოქმედებები ყოველი ახალი მოქმედების დამატებისას. გარდა ამისა, მოდიფიკაციისას საჭირო იქნება გადავსინჯოთ მდგომარეობასთან დაკავშირებული ყველა გადასვლები.

ამის თავიდან ასაცილებლად სიმბოლოს, რომელიც აღნიშნავს მდგომარეობას, შეიძლება დაერთოს ნებისმიერი მოქმედება შესვლისას (იგი აღინიშნება სიტყვით **entry**) და მოქმედება გამოსვლისას (აღინიშნება სიტყვით **exit**). მაშინ მითითებული მოქმედებები შესრულდებიან შესაბამისად მდგომარეობაში შესვლისას და მისგან გამოსვლისას.

- ***შიდა გადასვლები*** (გადასვლები, რომლებიც მუშავდება მდგომარეობიდან გამოსვლის გარეშე);

სისტემა იმყოფება რა რომელიღაც მდგომარეობაში, შესაძლებელია მიიღოს მოვლენა, რომელიც სასურველია გადამუშავდეს მდგომარეობიდან გაუსვლელად. ასეთ დამუშავებას უწოდებენ შიდა გადასვლას. დაუშვათ, რომ აუცილებელია დამუშავდეს მოვლენა, შესვლისას და გამოსვლისას მოქმედების აღუძვრელად. პრინციპში ეს შესაძლებელია გაკეთდეს უბრალო ავტომატებითაც, მაგრამ ამ დროს საჭიროა ყურადღების გამოჩენა, თუ რომელი გადასვლებისათვის უნდა შესრულდეს ეს მოქმედებები, ხოლო რომლისთვის არა.

ამის თავიდან ასაცილებლად სიმბოლოს, რომელიც აღნიშნავს მდგომარეობას შეიძლება დაერთოს შიდა გადასვლა (სიტყვით **newtarget**). თუ იმყოფებით ასეთ მდგომარეობაში და ხდება მითითებული მოვლენა, მაშინ შესაბამისი მოქმედება სრულდება მდგომარეობიდან გამოსვლისა და ხელმეორედ შესვლის გარეშე, ანუ მოვლენა მუშავდება შესვლისა და გამოსვლის მოქმედებების აღუძვრელად.

- ***ქვემდგომარეობები*** (მდგომარეობების შიგნით შეიძლება არსებობდეს ქვემდგომარეობები, როგორც თანმიმდევრული - აქტივიზირდებიან მიმდევრობით, ისე პარალელური - აქტივიზირდებიან ერთდროულად).

უბრალოს უწოდებენ ისეთ მდგომარეობას, რომელსაც არ გააჩნია შინაგანი სტრუქტურა. მდგომარეობას, რომელსაც გააჩნია ქვემდგომარეობები, ანუ ჩაშენებული მდგომარეობები, უწოდებენ შედგენილს. მას შესაძლებელია გააჩნდეს როგორც პარალელური (დამოუკიდებელი), ისე მიმდევრობითი ქვემდგომარეობები. შედგენილი მდგომარეობები გამოიძახება ისევე, როგორც მარტივი, მაგრამ აქვს დამატებითი გრაფიკული განყოფილება, რომელშიც ნაჩვენებია ჩართული ავტომატი. მდგომარეობათა ჩართვის სიღრმე არ არის შეზღუდული.

როდესაც გადასვლა ხდება შედგენილ მდგომარეობაზე, ჩართული ავტომატის მოქმედება იწყება მისი საწყისი მდგომარეობიდან (თუ რასაკვირველია გადასვლას არ მივყავართ რომელიმე ქვემდგომარეობაში). მაგრამ ხშირად საჭიროა დავამოდელიროთ ობიექტი ისე რომ მან დაიმახსოვროს ის ბოლო მდგომარეობა, რომელშიც ის იმყოფებოდა შედგენილი მდგომარეობიდან გამოსვლისას.

ასეთი სიტუაციებისათვის გამოიყენება – ისტორიული მდგომარეობები(**History states**). ისტორიული მდგომარეობა საშუალებას აძლევს შედგენილ მდგომარეობას, რომელიც შეიცავს მიმდევრობით ქვემდგომარეობებს, დაიმახსოვროს რომელი ქვემდგომარეობა იყო მიმდინარე შედგენილ მდგომარეობიდან გამოსვლის მომენტში. ისტორიული მდგომარეობა წარმოიდგინება წრით, რომელშიც იწერება სიმბოლო **H**.

სიმბოლოთი აღინიშნება შედარებით ახლო ისტორია, რომელშიც დაიმახსოვრება წინამდებარე მდგომარეობა, მაგრამ შეიძლება განისაზღვროს შორეული (**deep**) ისტორია, რომელიც გამოიხატება სიმბოლოთი **H***. შორეული ისტორია საშუალებას იძლევა დავიმახსოვროთ ჩართული ქვეავტომატების საბოლოო მდგომარეობები ჩართვის ნებისმიერი დონისათვის.

§ 3.6.2. მოვლენები და სიგნალები

ნებისმიერი წარმოსახვა, რომელიც შესაძლებელია მოხდეს ობიექტზე, მოდელირდება როგორც მოვლენა.

მოვლენა – ეს არსებითი ფაქტის აღწერაა, რომელსაც აქვს გარკვეული ადგილი დროსა და სივრცეში. ავტომატების კონტექსტში მოვლენა ეს სტიმულია, რომელსაც შეუძლია გამოიწვიოს გადასვლა ერთი მდგომარეობიდან მეორეში. განასხვავებენ შიდა და გარე მოვლენებს. გარე მოვლენები გადაიცემა სისტემასა და მის მომხმარებელს(აქტიორს) შორის, რომლის მაგალითი შეიძლება იყოს ღილაკის დაჭერა. შიდა მოვლენები გადაიცემა სისტემის შიგნით არსებულ ობიექტებს შორის, რომლის მაგალითი შეიძლება იყოს გამორიცხვა.

ობიექტ-ორიენტირებულ სისტემებში ძირითადათ ხდება მოვლენათა ოთხი ტიპის მოდელირება: სიგნალები, გამოძახება, დროის შუალედის ამოწურვა და მდგომარეობის შეცვლა.

სიგნალები. სიგნალი ეს დასახელებული ობიექტია, რომელიც ასინქრონულად აღიძვრება ერთი ობიექტით და მიიღება მეორეთი. გამორიცხვა, რომელიც ფართოდ

გამოიყენება თანამედროვე დაპროგრამების ენების უმეტესობაში, ყველაზე გავრცელებული სახეა შიდა სიგნალებს შორის. ობიექტების შექმნა და მოსპობაც სიგნალის ნაირსახეობას მიეკუთვნება.

სიგნალი შეიძლება გაიგზავნოს როგორც მდგომარეობის გადასვლის მოქმედება ავტომატში ან როგორც შეტყობინების გაგზავნა ურთიერთქმედებისას. ოპერაციების შესრულებისას ასევე შეიძლება გაიგზავნოს სიგნალი. იმისათვის, რომ მიუთითონ ოპერაციის მიერ სიგნალის გაგზავნა, შეიძლება ვისარგებლოთ მიმართებით ***დამოკიდებულება*** სტერეოტიპით **send**.

გამოძახების მოვლენები. თუ სიგნალის მოვლენა წარმოადგენს სიგნალის აღძვრას, გამოძახების მოვლენა განკუთვნილია ოპერაციის შესრულების აღწერისათვის. ორივე შემთხვევაში მოვლენას შეუძლია გამოიწვიოს მდგომარეობის შეცვლა ავტომატში. იმ დროს როდესაც სიგნალი წარმოადგენს ასინქრონულ მოვლენას, გამოძახების მოვლენა ჩვეულებრივ სინქრონულია. ეს ნიშნავს იმას, რომ როდესაც ერთი ობიექტი ინიციირებას ახდენს ოპერაციის შესრულებაზე მეორე ობიექტზე, რომელსაც გააჩნია თავისი ავტომატი, მართვა გადაიცემა გამომგზავნიდან მიმღებზე, ამუშავდება შესაბამისი გადასვლა, შემდეგ ოპერაცია სრულდება, მიმღები გადადის ახალ მდგომარეობაში და უბრუნებს მართვას გამომგზავნს.

დროითი და ცვლილების მოვლენები. დროითი მოვლენა წარმოადგენს დროითი შუალედის ამოწურვას. მოდელირებისას იგი შეიძლება წარმოვადგინოთ გასაღებური სიტყვით **after**(შემდეგ), მას მოსდევს გამოსახულება, რომელიც ითვლის დროის გარკვეულ შუალედს. გამოსახულება შეიძლება იყოს მარტივი ან რთული. თუ არ არის მითითებული, დროის ათვლა იწყება მიმდინარე მდგომარეობაში შესვლის მომენტიდან.

ცვლილების მოვლენით აღიწერება მდგომარეობის შეცვლა ან გარკვეული პირობის შესრულება. იგი წარმოიდგინება გასაღებური სიტყვით **when**, რომელსაც

მოსდევს ბულის გამოსახულება. ასეთი გამოსახულება შესაძლებელია აბსოლუტური დროის მომენტის ან პირობის შესამოწმებლად (მაგალითად, **when t<5**).

მოვლენის გაგზავნა და მიღება. სიგნალისა და გამოძახების მოვლენებში უკიდურეს შემთხვევაში მონაწილეობენ ორი ობიექტი: ობიექტი, რომელიც აგზავნის სიგნალს ანუ ოპერაციის ინიციირებას ახდენს, და ობიექტი, რომელსაც ეგზავნება მოვლენა. რამდენადაც სიგნალები თავისი ბუნებით ასინქრონულია, ხოლო ასინქრონული გამოძახებები თავისთავად წარმოადგენენ სიგნალებს, მოვლენათა სემანტიკა უკავშირდება აქტიური და პასიური ობიექტების სემანტიკას.

ნებისმიერი კლასის ეგზემპლიარს შეუძლია გაგზავნოს სიგნალი მიმღებ ობიექტთან ან მოახდინოს მასში ოპერაციის ინიციირება. გაგზავნის რა სიგნალს მიმღებთან, იგი აგრძელებს თავის მართვის ნაკადს, არ ელოდება რა მისგან პასუხს. პირიქით, თუ ობიქტი ახდენს ოპერაციის ინიციირებას, იგი უნდა დაელოდოს მიმღებისაგან პასუხს. დეკანმა შესაძლებელია მოახდინოს ოპერაციის ***გადაყვანის*** ინიციირება კლასი სტუდენტის რომელიღაც ეგზემპლიარში, ღებულობს რა ***ბრძანებას*** კურსიდან კურსზე გადაყვანის შესახებ, რომლითაც ფაქტიურად ცვლის მის მდგომარეობას. რამდენადაც ეს სინქრონული გამოძახებაა, დეკანი დაელოდება, სანამ ოპერაცია დასრულდება.

ნებისმიერი კლასის ნებისმიერი ეგზემპლიარი შეიძლება იყოს მოვლენა სიგნალის ან მოვლენა გამოძახების მიმღები. თუ ეს სინქრონული მოვლენაა, გამგზავნი და მიმღები იმყოფებიან მდგომარეობაში რანდევუ(§ 3.5.1.) ოპერაციის შესრულების მთელი ხანგრძლივობისას. ეს ნიშნავს, რომ გამგზავნის მართვის ნაკადი ბლოკირდება მიმღების მართვის ნაკადით, სანამ ოპერაცია არ დასრულდება. თუ ეს სიგნალია, გამგზავნი და მიმღები არ შედიან მდგომარეობაში რანდევუ: გამგზავნი აგზავნის სიგნალს, მაგრამ არ ელოდება პასუხს მიმღებისაგან. გამოძახების მოვლენები, რომლებსაც ღებულობს ობიექტი, მოდელირდება როგორც ოპერაციები ამ ობიექტის კლასებზე. დასახელებული სიგნალები, რომლებსაც

ღებულობს ობიექტი, მოდელირდება ჩამონათვალის სახით კლასის დამატებით განყოფილებაში.

გამორიცხვები. კლასების ქცევის ვიზუალირების მნიშვნელოვან ნაწილს წარმოადგენს გამორიცხვების სპეციფიცირება, რომლებსაც შეუძლიათ აღაგზნონ მისი ოპერაციები. თუ გაქვთ კლასი ან ინტერფეისი, მაშინ ოპერაციები, რომლებიც შესაძლებელია მათზე გამოვიძახოთ ნათლად ჩანს აღწერიდან, მაგრამ იმის გაგება თუ რომელ გამორიცხვებს აღაგზნებენ ისინი არ არის ადვილი, თუ ეს ნათლად არ არის მითითებული მოდელში.

გამორიცხვები წარმოადგენენ სიგნალების კერძო შემთხვევებს და მოდელირდებიან სტერეოტიპული კლასებით. გამორიცხვები შესაძლებელია მიუერთოდ ოპერაციების სპეციფიკაციებს. გამორიცხვების მოდელირება წარმოადგენს ოპერაციას, გარკვეული აზრით საწინააღმდეგოს სიგნალთა სიმრავლის მოდელირებისა. სიგნალთა ოჯახის მოდელირების ძირითადი მიზანია – მოვახდინოთ იმის სპეციფიცირება, თუ რომელი სიგნალები შეუძლია მიიღოს აქტიურმა ობიექტმა; გამორიცხვების მოდელირების მიზანია უჩვენოთ, რომელი გამორიცხვები შეიძლება აღიძვრეს ობიექტით თავისი ოპერაციებიდან.

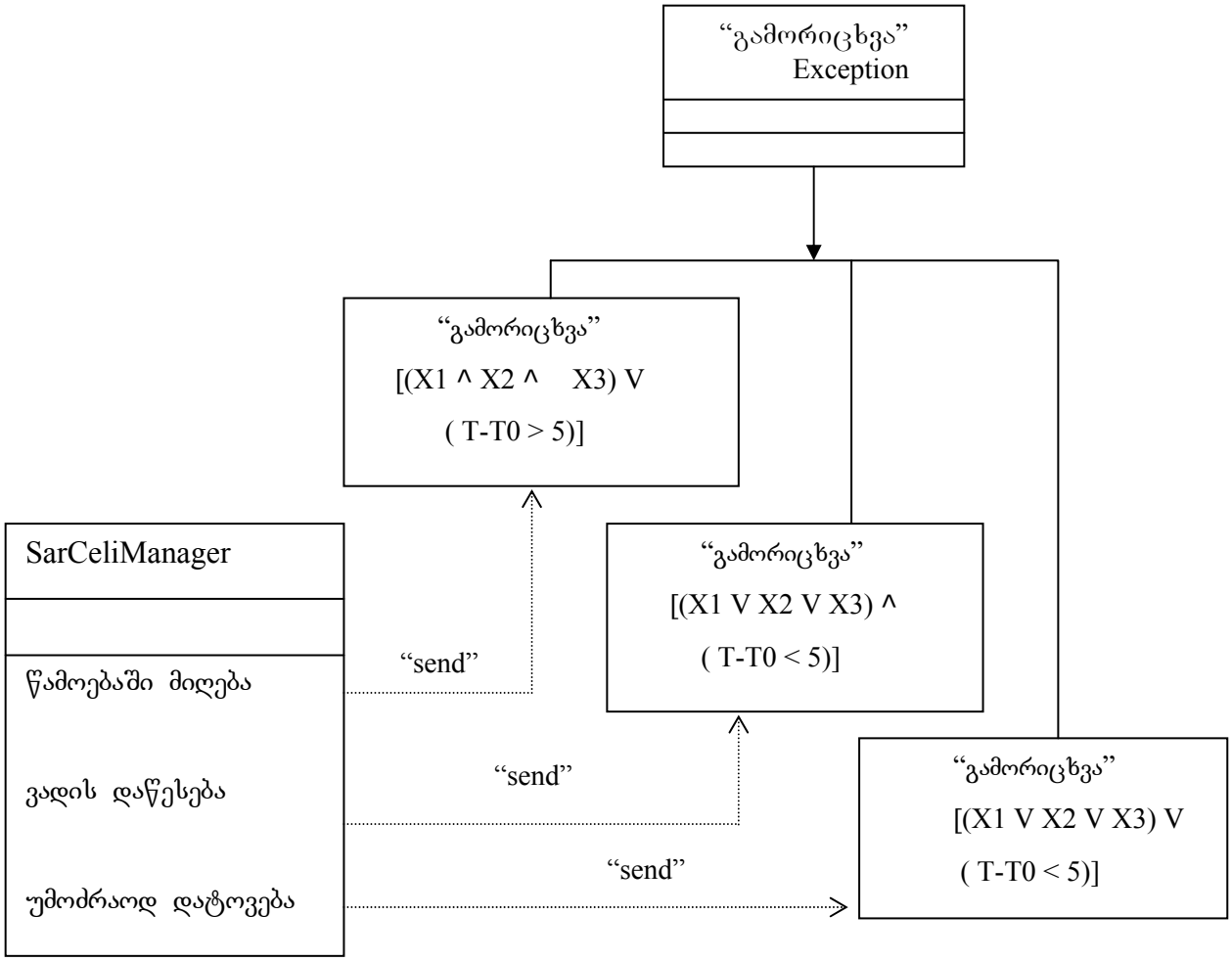
მაგალითისათვის ნახ. 3.6.3. –ზე წარმოდგენილია გამორიცხვების იერარქიული მოდელი, რომლებიც შესაძლებელია აღიძვრას კლასით **SarCeliManager**. ამ იერარქიის თავში იმყოფება აბსტრაქტული კლასი **Exception,** ხოლო ქვევით სპეციალიზირებული გამორიცხვები. ნახაზზე მიღებულია შემდეგი აღნიშვნები:

X1 – სარჩელის შინაარსი აკმაყოფილებს საქართველოს სამოქალაქო კოდექსის (სსკ) 178-ე მუხლს;

X2 – სახელმწიფო ბაჟი შეტანილია (სსკ-ის 178-ე მუხლი);

X3 – სრულდება სსკ-ის 186-ე მუხლი.

X3 = X31 ^ X32 ^ X33 ^ X34 ^ X35 ^ X36 ^ X37 ;



ნახ.3.6.3.

სადაც,

X31 – სარჩელი არ ექვემდებარება სასამართლო უწყებას;

X32 - არსებობს სასამართლო გადაწყვეტილება ან განჩინება მოსარჩელის მიერ სარჩელზე უარის თქმის, მოპასუხის მიერ სარჩელის ცნობის ან მხარეთა მორიგების დამტკიცების შესახებ;

X33 – ამ ან სხვა სასამართლოს წარმოებაშია საქმე დავაზე იმავე მხარეებს შორის, იმავე საგანზე და იმავე საფუძვლით;

X34 – მხარეებს დადებული აქვთ ხელშეკრულება, რომ მათ შორის დავა გადასაწყვეტად გადაეცეს კერძო არბიტრაჟს;

X35 – საქმე ამ სასამართლოს განსჯადი არ არის;

X36 – სარჩელი შეიტანა ქმედუუნარო პირმა;

X37 – დაინტერესებული პირის სახელით განცხადება შეიტანა პირმა, რომელსაც არა აქვს უფლებამოსილება საქმის წარმოებაზე.

T – მიმდინარე თარიღი;

T0 – მოსამართლეზე სარჩელის გადაცემის თარიღი.

§ 3.6.3. მოქმედებები და გადასვლები

გადასვლა ეს მიმართებაა ორ მდგომარეობას შორის, რომელიც გვიჩვენებს, რომ ობიექტმა, რომელიც იმყოფებოდა პირველ მდგომარეობაში, უნდა შეასრულოს გარკვეული მოქმედებები და გადავიდეს მეორე მდგომარეობაში, როგორც კი მოხდება მითითებული მოვლენა და დაკმაყოფილდება მითითებული პირობები. მდგომარეობის ასეთი ცვლილებისას ამბობენ, რომ ამუშავდა გადასვლა. სანამ გადასვლა არ ამუშავდა, ობიექტი იმყოფება საწყის მდგომარეობაში. ამუშავების შემდეგ ის იმყოფება მიზნობრივ მდგომარეობაში.

გადასვლას განმარტავენ ხუთი ელემენტით:

- ***საწყისი მდგომარეობა***, ანუ მდგომარეობა, რომლიდანაც წარმოებს გადასვლა. თუ ობიექტი იმყოფება საწყის მდგომარეობაში, მაშინ საწყისიდან გადასვლა ამუშავდება, როდესაც ობიექტი მიიღებს მოვლენა-ტრიგერს, რომელიც ახდენს ამ გადასვლის ინიციირებას, ამასთან უნდა შესრულდეს დამცავი პირობა (თუ იგი მოცემულია);
- ***მოვლენა-ტრიგერი.*** მოვლენა, რომლის მიღებისას საწყის მდგომარეობაში მყოფ ობიექტზე შესაძლებელია ამუშავდეს გადასვლა (ამასთან უნდა სრულდებოდეს დამცავი პირობა). არსებობენ არატრიგერული გადასვლები, რომლებისთვისაც არ არის არავითარი მოვლენა-ტრიგერი. არატრიგერული გადასვლა, რომელსაც კიდევ უწოდებენ გადასვლას დასრულებისას ინიციირდება, როდესაც მუშაობა საწყის მდგომარეობაში დამთავრდება.

- ***დამცავი პირობა***. ბულის გამოსახულება, რომელიც გამოითვლება მოვლენა-ტრიგერის მიღებისას. თუ მნიშვნელობა ჭეშმარიტია, მაშინ გადასვლას ნება ეძლევა ამუშავდეს, თუ მცდარია - გადასვლა არ იმუშავებს. თუ ამ დროს არ არის მოცემული სხვა გადასვლა, რომლის ინიციირება იმავე მოვლენით ხდება, მაშინ მოვლენა იკარგება. დამცავი პირობები გამოიხატება ბულის გამოსახულებით, მოთავსებული კვადრატულ ფრჩხილებში, მოვლენა-ტრიგერის შემდეგ. დამცავი პირობა ითვლება მხოლოდ მოვლენა-ტრიგერის აღძვრის შემდეგ, რომელიც ახდენს შესაბამისი გადასვლის ინიციირებას. ამიტომ ერთი მდგომარეობიდან შესაძლებელია რამოდენიმე გადასვლა ერთი და იგივე მოვლენა-ტრიგერით იმ შემთხვევაში, თუ არცერთი ორი დამცავი პირობიდან არ ხდებიან ერთდროულად ჭეშმარიტი. დამცავი პირობა გამოითვლება ერთხელ ყოველი გადასვლისათვის მოვლენის დადგომის მომენტში, მაგრამ შესაძლებელია გამოითვალოს ხელახლა თუ გადასვლა ინიციირდება ხელმეორედ. ბულის გამოსახულებაში შეიძლება ჩავრთოთ პირობები, რომელშიც ფიგურირებენ ობიექტის მდგომარეობები.
- ***მოქმედება*** ან ელემენტარული გამოთვლა, რომელსაც შეუძლია უშუალოდ იმოქმედოს ავტომატის მფლობელ ობიექტზე ან მოახდინოს ირიბი ზემოქმედება სხვა ობიექტებზე, რომლებიც იმყოფებიან მის მხედველობაში. მოქმედებებს მიეკუთვნებიან ოპერაციათა გამოძახება, სხვა ობიექტის შექმნა და მოსპობა, ან სიგნალის გაგზავნა ობიექტთან. სიგნალის გაგზავნის აღნიშვნისათვის განსაზღვრულია სპეციალური ნოტაცია – სიგნალის სახელს წინ უსწრებს გასაღებური სიტყვა **send**. მოქმედება ყოველთვის ელემენტალურია, ე.ი. არ შეიძლება შეწყდეს სხვა მოვლენით და შესაბამისად სრულდება დამთავრებამდე. ამით იგი განსხვავდება მოღვაწეობისაგან, რომლის შესრულება შესაძლებელია შეწყდეს სხვა მოვლენით.

მოღვაწეობა. როდესაც ობიექტი იმყოფება გარკვეულ მდგომარეობაში, ის ჩვეულებრივ უმოქმედოა და ელოდება გარკვეული მოვლენების აღძვრას. მაგრამ

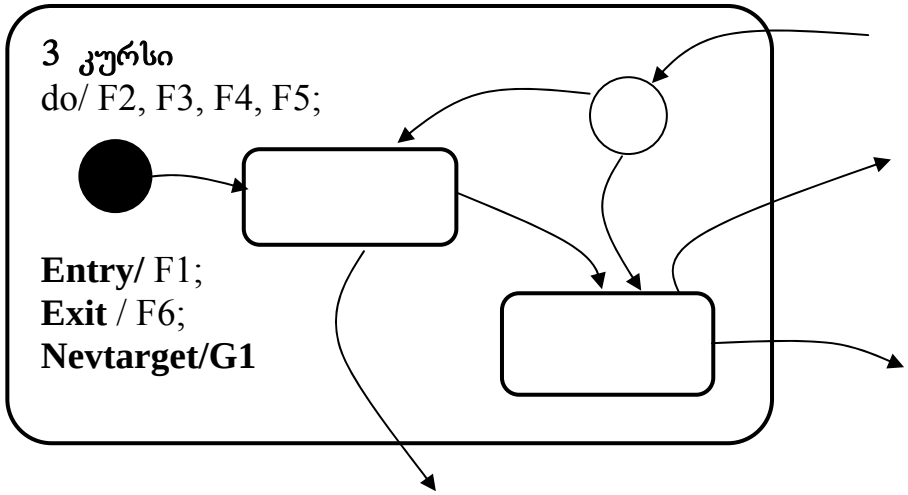
ხშირად საჭიროა უწყვეტად მიმდინარე პროცესების მოდელირება. ვიმყოფებით რა ასეთ მდგომარეობაში, ობიექტი დაკავებულია მანამდე, სანამ ეს მოღვაწეობა არ შეწყდება მოვლენით. გადასვლით **do** (შევასრულოთ) აღინიშნება ის სამუშაოები, რომლებიც უნდა შესრულდეს მდგომარეობის შიგნით მას შემდეგ როგორც კი დამუშავდება მდგომარეობაში შესვლის მოქმედება. მოღვაწეობა დაკავშირებული **do** გადასვლასთან შესაძლებელია იყოს სხვა ავტომატის სახელი. დასაშვებია მიმდევრობითი მოქმედებებიც, მაგ. **do**/op1;op2;op3. მითითებული მოქმედებების შესრულებისას მომცავ მდგომარეობას შეუძლია დაამუშაოს მოვლენები და არ არის გამორიცხული, რომ ეს გამოიწვევს მდგომარეობიდან გამოსვლას.

- ***მიზნობრივი მდგომარეობა***, რომელიც დგება აქტიური გადასვლის დამთავრების შემდეგ.

გადასვლას შეიძლება გააჩნდეს რამოდენიმე საწყისი მდგომარეობა, ასევე რამოდენიმე მიზნობრივი მდგომარეობა.

მაგალითისათვის ნახ.3.6.4.-ზე მოყვანილია კლასი სტუდენტის ერთ - ერთი მდგომარეობა ***3 კურსი,*** თავისი ქვემდგომარეობებით. მოცემულ მდგომარეობას გააჩნია:

- მოქმედება შესვლისას **Entry/** F1, სადაც F1- ლექცია-სემინარების ჩატარების ცხრილის გაცნობა.
- შიდა გადასვლა **Nevtarget/G1,** სადაც G1- სტუდენტის აღდგენა კურსზე;
- მოქმედება გამოსვლისას **Exit** / F6, სადაც F6- ზაფხულის სასესიო გამოცდებში მონაწილეობა;
- მოღვაწეობა, რომელიც გამოისახება შემდეგი მოქმედებებით do/ F2, F3, F4, F5, F5, სადაც, F2- ლექცია-სემინარებზე დასწრება; F3- პრაქტიკული და ლაბორატორიული სამუშაოების შესრულება; F4- სარეიტინგო გამოკითხვებში მონაწილეობა; F5- ზამთრის სასესიო გამოცდებში მონაწილეობა.



ნახ.3.6.4.

მოყვანილ მაგალითში ნაჩვენებია აგრევე ისტორიული მდგომარეობა იმ შემთხვევისათვის, როდესაც სტუდენტი ერთ-ერთ ქვემდგომარეობაში იღებს აკადემიურ შვებულებას. ასეთ შემთხვევაში ხდება გამოსვლა მოცემული შედგენილი მდგომარეობიდან. აკადემიურ შვებულების გავლის შემდეგ სტუდენტი ბრუნდება მოცემულ მდგომარეობაში. იმისათვის, რომ მიუთითოთ შედგენილი მდგომარეობის რომელი მდგომარეობიდან მოხდა გამოსვლა გამოყენებული არის შესაბამისი აღნიშვნა.

§ 3.6.4. სასიცოცხლო ციკლის აგების წესი

რეაქტიული ობიექტის ქცევის მოდელირებისას შესაძლებელია მოქმედების სპეციფიცირება, დაუკავშირებენ რა მას გადასვლას ან მდგომარეობის შეცვლას. ლიტერატურაში ავტომატი, რომლის ყველა მოქმედება მიბმულია გადასვლებთან, უწოდებენ **მილის** მანქანას, ხოლო ავტომატი რომლის მოქმედებები მიბმულია მდგომარეობებთან, **მურის** მანქანას. მდგომარეობათა დიაგრამის დამუშავებისას ჩვეულებრივ გამოიყენება **მილის** და **მურის** მანქანების კომბინაცია.

ობიექტის სასიცოცხლო ციკლის მოდელირებისათვის თავიდან ამოირჩევა კონტექსტი ავტომატისათვის, ეს იქნება კლასი, პრეცედენტი ან სისტემა მთლიანობაში. დადგინდება ობიექტისათვის საწყისი და საბოლოო მდგომარეობები,

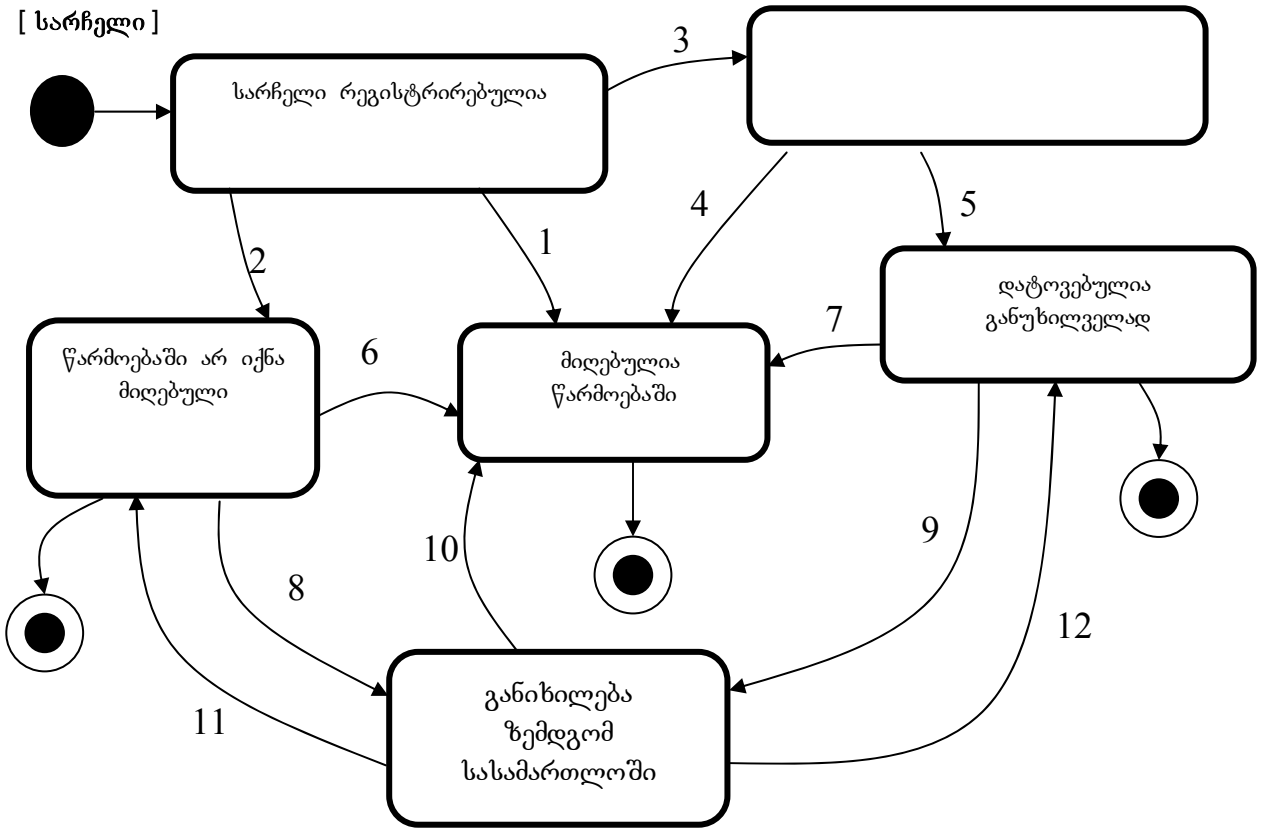
ობიექტის მდგრადი მდგომარეობები, ისეთი რომლებშიც ის შეიძლება იმყოფებოდეს განუსაზღვრელად დიდი დროის განმავლობაში. იწყებენ ზედა დონის მდგომარეობებიდან, ხოლო შემდეგ გადადიან ქვემდგომარეობებზე. მოახდენენ მდგრადი მდგომარეობების მოწესრიგებას ობიექტის მთელი სასიცოცხლო ციკლის განმავლობაში. შემდეგ დადგინდება რომელ მოვლენებს შეუძლიათ მდგომარეობებს შორის გადასვლების ინიციირება. წარმოადგენენ ამ მოვლენებს როგორც გადასვლების ტრიგერებს. დაუკავშირებენ მოქმედებებს გადასვლებს(მილის მანქანა) და/ან მდგომარეობებს(მურის მანქანა). თუ შესაძლებელია მოახდენენ ავტომატის გამარტივებას ქვემდგომარეობებით, განშტოებებით, შერწყმით და ისტორიული მდგომარეობებით. გასინჯავენ, რომ ნებისმიერი მდგომარეობის მიღწევა შესაძლებელია მოვლენათა გარკვეული კომბინაციისას ანუ დარწმუნდნენ ჩიხური სიტუაციების არ არსებობაში, ანუ ისეთების რომლებიდანაც არ არის გადასვლა მოვლენის არც ერთი კომბინაციისას. ამისათვის, შეამოწმებენ ავტომატს ხელით ან ინსტრუმენტალური საშუალებით და დადგინდება როგორ იქცევა ის მოვლენათა მოსალოდნელი თანმიმდევრობისას და მათზე რეაქციისას.

ბოლოს უნდა შემოწმდეს რომ ყველა მოქმედებები, რომლებიც მოხსენებულია ავტომატში, შეთავსებულია მომცავი ობიექტის მიმართებებით, მეთოდებით და ოპერაციებით. შემოწმებისას შესაძლებელია შეიცვალოს ავტომატის სტრუქტურა, ხელახლა გაისინჯოს იგი მოსალოდნელ მოვლენათა თანმიმდევრობაზე, რათა დარწმუნდნენ, რომ ობიექტის სემანტიკა შეიცვალა.

მაგალითისათვის ნახ.3.6.5.-ზე მოყვანილია მდგომარეობათა დიაგრამა სამოქალაქო სამართალწარმოების ერთ-ერთი ობიექტისათვის – ***სარჩელი***, რომელიც აღწერს ობიექტი - ***სარჩელის*** სასიცოცხლო ციკლს.

მოყვანილი დიაგრამა აღწერს ავტომატს, რომლის საწყისია სარჩელის შემოტანა სასამართლოში, ხოლო მიზნობრივი მდგომარეობაა ***სარჩელი მიღებულია წარმოებაში.***

ქვემოთ მოყვანილია თითოეული გადასვლის შესაბამისი მოვლენა-ტრიგერი, დამცავი პირობა და მოქმედება, რომელიც თან ახლავს ამ გადასვლას. მოცემულ შემთხვევაში ეს მოქმედებები წარმოადგენენ განჩინებებს, რომლებიც აფიქსირებენ სარჩელის ერთი მდგომარეობიდან მეორეში გადასვლას.



ნახ.3.6.5.

მოცემულ მაგალითში გამოყენებულია იგივე აღნიშვნები, რომელიც გვქონდა წინა პარაგრაფში.

1. გადაეცა მოსამართლეს(5) [(X1 ^ X2 ^ X3) V (T-T0 > 5)] / განჩინება1
2. გადაეცა მოსამართლეს(5) [(X1 VX2 V X3) ^ (T-T0 < 5)] / განჩინება2
3. გადაეცა მოსამართლეს(5) [(X1V X2 V X3) V (T-T0 < 5)] / განჩინება3

შემოსული სარჩელი თავდაპირველად ხვდება მდგომარეობაში ***რეგისტრაცია***, რომლის შემდეგ სარჩელი გადაეცემა მოსამართლეს, სწორედ აღნიშნული მოვლენა წარმოადგენს მოვლენა-ტრიგერს, რომელიც ინიციირებას უწევს მომდევნო სამ მდგომარეობაში გადასვლას – ***დაუწესდა ვადა ხარვეზის შესავსებად***, ***მიღებულია***

წარმოებაში და ***უარი ეთქვა წარმოებაში მიღებაზე***. თითოეულ მათგანს გააჩნია დამცავი პირობა და მოქმედება, რომელიც მიბმულია შესაბამის გადასვლებთან (მილის მანქანა). მოსამართლეს განჩინების გამოტანისათვის გამოყოფილი აქვს 5 დღიანი ვადა, რაც მითითებულია მოვლენა-ტრიგერის პარამეტრში. განხილვის ვადის გადაჭარბების შემთხვევაში სარჩელი ნაკლოვანებების მიუხედავად ითვლება წარმოებაში მიღებულად. სწორედ ეს ფაქტი არის მითითებული 1-ლი გადასვლის დამცავ პირობაში.

მდგომარეობიდან ***დაუწესდა ვადა ხარვეზის შესავსებათ*** გადასვლით 4 (ხარვეზი შევსილია [T1 < T2] / განჩინება4) ხდება გადასვლა მდგომარეობაში ***მიღებულია წარმოებაში***, ხოლო გადასვლით 5 (ხარვეზი არ არის შევსილი [T1 < T2] / განჩინება5) მდგომარეობაში ***დატოვებულია განუხილველად.***

მდგომარეობებში ***უარი ეთქვა წარმოებაში მიღებაზე, დატოვებულია განუხილველად*** დასაშვებია კერძო საჩივრის შეტანა. თუ კერძო საჩივარი არ იქნა შეტანილი პროცესი მთავრდება, ხოლო თუ შეტანილ იქნა 12 დღის ვადაში მას იხილავს იგივე სასამართლო. თუ სარჩელი დაკმაყოფილდა ამუშავდება 6(7) გადასვლა:

6. კერძო საჩივარი (12) & დაკმაყოფილდა [Ť < 12] / განჩინება6

7. კერძო საჩივარი (12) & დაკმაყოფილდა [Ť < 12] / განჩინება7

აქ Ť – თი აღნიშნულია კერძო საჩივრის შეტანის თარიღი. იმ შემთხვევაში თუ საჩივარი არ დაკმაყოფილდება ამუშავდება 8(9) გადასვლა, რითაც მოხდება გადასვლა მდგომარეობაში ***განიხილება ზემდგომ სასამართლოში:***

8. კერძო საჩივარი (12) & არ დაკმაყოფილდა [Ť < 12] / განჩინება8

9. კერძო საჩივარი (12) & არ დაკმაყოფილდა [Ť < 12] / განჩინება9

აღნიშნულ მდგომარეობას გააჩნია მოქმედება შესვლისა და გამოსვლისას:

entry. საჩივარი(სარჩელი) გადაიგზავნოს 5 დღის ვადაში ზემდგომ სასამართლოში;

exit. სარჩელი და მიღებული განჩინება საჩივარზე უბრუნდება სასამართლოს.

როგორც ნახაზიდან ჩანს თუ ზემდგომ სასამართლოში საჩივარი დაკმაყოფილდა ამუშავდება გადასვლა 10, წინააღმდეგ შემთხვევაში 11(12) გადასვლები:

10. კერძო საჩივარი დაკმაყოფილდა / განჩინება10
11. კერძო საჩივარი არ დაკმაყოფილდა / განჩინება11
12. კერძო საჩივარი არ დაკმაყოფილდა / განჩინება12

3.7. მოთხოვნათა რეალიზების მექანიზმების მოდელირება

ახალი სისტემის არქიტექტურის აგების ან არსებულის განვითარებისას, ნებისმიერ შემთხვევაში, დამპროექტებელი არასდროს არ იწყებს ნოლიდან. პირიქით, წინა გამოცდილება და მიღებული შეთანხმებები აძლევს საფუძველს გამოიყენოს ტიპიური საშუალებები ტიპიური პრობლემის გადაწყვეტისათვის. მაგალითად, დეშიფრაციის სისტემის შექმნისას თავისი სარგებლიანობა დაამტკიცა არქიტექტურამ "საკლასო დაფის" საფუძველზე, რომელიც კარგად მიესადაგება რთული მოცანების გადაწყვეტას მეთოდით ცდა და შეცდომა. რაც წარმოადგენს ნიმუშის მაგალითს - ტიპიურ გადაწყვეტას ტიპიური ამოცანისათვის მოცემულ კონტექსტში.

ნებისმიერი კარგად სტრუქტურირებული ობიექტ-ორიენტირებული სისტემა მოიცავს ნიმუშებს აბსტრაქციის ნებისმიერ დონეზე. მკაფიოდ გამოვყოფთ რა ნიმუშებს სისტემაში, იგი გახდება უფრო მარტივი და გასაგები. მაგალითად, თუ აიღებთ რომელიმე პროგრამის უცნობ საწყის ტექსტს და მოინდომებთ მის მოდიფიცირებას, დაგჭირდებათ საკმაოდ დიდი ჯაფა რათა გაერკვიოთ თუ როგორ არიან მისი ნაწილები დაკავშირებული ერთმანეთთან. მაგრამ, თუ მოგცემენ იგივე ტექსტს და აგიხსნიან რა მექანიზმით ურთიერთქმედებენ ჩამოთვლილი კლასები ერთმანეთთან, მიიღებთ უფრო მკაფიო წარმოდგენას იმის შესახებ, თუ როგორ მუშაობს სისტემა. იგივე შეიძლება ითქვას მთლიანად სისტემისათვის. ამიტომ, სისტემის აგებისას სასურველი იქნება აღიწეროს მისთვის დამახასიათებელი ნიმუშები, რათა დავეხმაროთ მათ, ვინც შემდგომში მოინდომებს მის ხელმეორედ გამოყენებას ან მოახდინოს მისი მოდიფიცირება. პრაქტიკაში ინტერეს წარმოადგენს ორი სახის ნიმუშები: პროექტირების ნიმუშები(მექანიზმები) და არქიტექტურული ნიმუშები(კარკასები). პროექტირების ნიმუშები აღწერენ სტრუქტურას და ქცევას კლასების ნაკრებისათვის, ხოლო არქიტექტურული ნიმუშები სტრუქტურას და ქცევას მთლიანად სისტემისათვის.

მექანიზმების მოდელირებას ახდენენ კოოპერაციის მეშვეობით. სისტემის არქიტექტურის კონტექსტში კოოპერაცია საშუალებას იძლევა მივანიჭოთ სახელი გარკვეულ კონცეპტუალურ ფრაგმენტს, რომელიც მოიცავს როგორც სტატიკურ, ისე დინამიკურ ასპექტებს. კოოპერაციას უწოდებენ კლასების და სხვა ელემენტების ერთობლიობას, რომლებიც მუშაობს ერთობლივად კოოპერაციული ეფექტის უზრუნველსაყოფად, უფრო მნიშვნელოვანის, ვიდრე შემადგენელთა ჯამია. შესაბამისად კოოპერაციას გააჩნია სტრუქტურული და ქცევითი შემადგენელი.

კოოპერაციის სტრუქტურული შემადგენელი შეიცავს კლასებს, მაგრამ ამავე დროს კოოპერაცია თვითონ არ ფლობს არც ერთ სტრუქტურულ ელემენტს. იგი მხოლოდ მიუთითებს კლასებს, ინტერფეისებს, კომპონენტებს, კვანძებს და სხვა სტრუქტურულ ელემენტებს, რომლებიც გამოცხადებულია სხვა ადგილზე, ან იყენებენ მათ. ამიტომ კოოპერაცია ასახელებს სისტემური არქიტექტურის კონცეპტუალურ და არა ფიზიკურ ფრაგმენტს. უფრო მეტიც, ერთმა და იმავე ელემენტმა შეიძლება მონაწილეობა მიიღოს რამოდენიმე კოოპერაციაში.

თუ გვაქვს კოოპერაცია, რომელიც ასახელებს სისტემის კონცეპტუალურ ფრაგმენტს, შეგვიძლია გავხსნათ იგი და ვნახოთ მისი დამალული სტრუქტურული დეტალები. გახსნისას დგინდება კლასები, რომლებიც შესაძლებელია აღიწეროს კლასების დიაგრამაზე.

თუ სტრუქტურული შემადგენელი კოოპერაციისა ჩვეულებრივ გამოისახება კლასების დიაგრამის სახით, მისი ქცევითი შემადგენელი გამოისახება ურთიერთქმედების დიაგრამის სახით. ეს დიაგრამა აღწერს ურთიერთქმედებას, რომელიც შეესაბამება ქცევას, შედგენილს შეტყობინებების გაცვლაში ობიექტებს შორის გარკვეულ კონტექსტში. ურთიერთქმედების კონტექსტს ადგენს მომცავი კოოპერაცია. შეგვიძლია ნებისმიერი ქცევითი დიაგრამის გამოყენება, რამდენადაც უმეტეს შემთხვევაში ისინი სემანტიკურად ექვივალენტურია.

კოოპერაციის მეშვეობით არა მარტო ასახელებენ სისტემურ მექანიზმებს, არამედ გამოიყენება პრეცედენტების და ოპერაციების რეალიზებისათვის. პრეცედენტების, რომლის მეშვეობითაც აღიწერება სისტემისადმი წაყენებული მოთხოვნები, რეალიზება ხდება კოოპერაციის მეშვეობით.

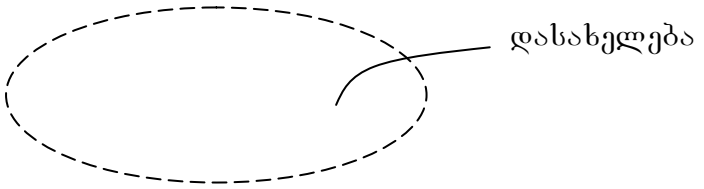
მაგალითად, პრეცედენტში ***სპეციალობის დატვირთვის გაანგარიშება*** მონაწილეობას ღებულობს V1 - სასწავლო გეგმა, V2 - სტუდენტთა კონტიგენტი, V3 - ნორმატივი, V4 - დატვირთვის მენეჯერი, V5 - დატვირთვის უწყისი(§ 2.3.1.). მოყვანილი არსების ერთობლივად მუშაობით ხორციელდება პრეცედენტი ***სპეციალობის დატვირთვის გაანგარიშება*** სემანტიკის რეალიზება. ზოგიერთი მათგანი შესაძლებელია ჩართული იყოს სხვა მექანიზმებშიც ან სხვა პრეცედენტების რეალიზებაში, მაგალითად პრეცედენტის ***სასწავლო გეგმის შედგენა, ინდივიდუალური დატვირთვის გაანგარიშება***(§ 2.1.2.).

მოყვანილ მაგალითში არსებს შორის ურთიერთქმედება შესაძლებელია სულ მცირე ორი სცენარით გამოვსახოთ:

- დისციპლინის დატვირთვა (პრაქტიკული და ლაბორატორიული მეცადინეობები) სასწავლო გეგმით გათვალისწინებული საათებით (Q1);
- დისციპლინის დატვირთვა (პრაქტიკული და ლაბორატორიული მეცადინეობები) სასწავლო გეგმით გათვალისწინებული საათების გაორმაგებით (Q2).

2.3.2. ნახაზზე (§ 2.3.1.) მოყვანილია ურთიერთქმედების დიაგრამა, რომელიც შეესაბამება ძირითად ნაკადს (Q1). ძირითადი ნაკადისაგან განსხვავებით მეორე შემთხვევაში ჯგუფში არსებული სტუდენტების რაოდენობა იყოფა შუაზე და პროცესი გრძელდება ძირითადი ნაკადის ანალოგიურად.

ასეთი კლასების ერთობლიობა V={V1, V2, V3, V4, V5} (სტრუქტურული შემადგენელი), აღებული ურთიერთქმედებით მათ შორის Q= {Q1, Q2} (ქცევითი შემადგენელი), ქმნის მექანიზმს, რომელსაც წარმოადგენს კოოპერაცია Ur. გრაფიკულად კოოპერაცია გამოისახება ელიფსის სახით წყვეტილი კონტურით.



ნახ. 3.7.1.

სისტემის ანალიზისას ვხელმძღვანელობთ იმით, თუ როგორ შეიძლება მოცემული პრეცედენტის გამოყენება. გადავდივართ რა რეალიზაციის ეტაპზე, საჭირო ხდება იდენტიფიცირებული პრეცედენტების რეალიზება კონკრეტული სტრუქტურებისა და ქცევების სახით. საერთო ჯამში ყოველი პრეცედენტი რეალიზებულ უნდა იქნას ერთი ან რამდენიმე კოოპერაციით.

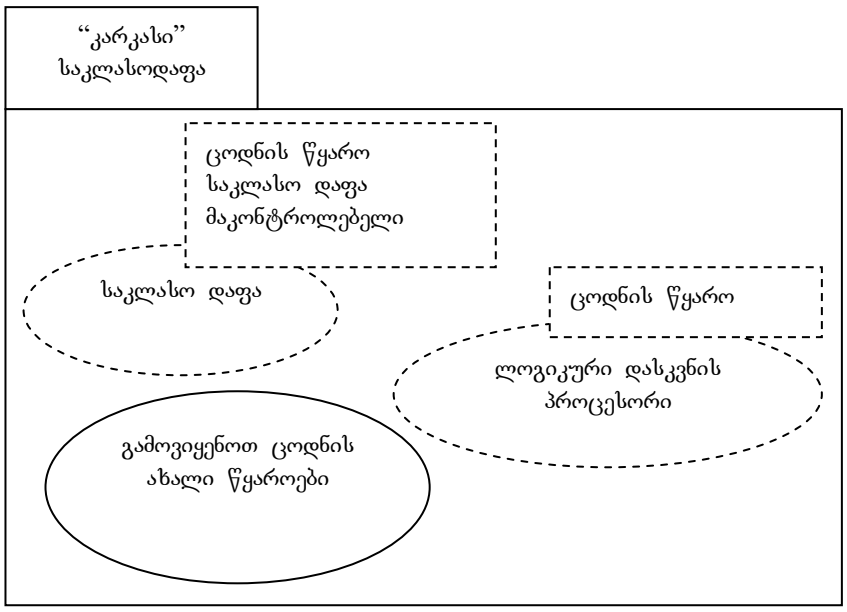
პრეცედენტების რეალიზების მოდელირებისათვის თავიდან უნდა მოხდეს იმ სტრუქტურული ელემენტების იდენტიფიცირება, რომლებიც შეადგენს პრეცედენტის სემანტიკას. შემდეგ მოახდენენ ამ სტრუქტურული ელემენტების ორგანიზებას კლასების დიაგრამაში და განიხილავენ ცალკეულ სცენარებს, რომლებიც წარმოადგენს მოცემულ პრეცედენტებს. ამ სცენარების დინამიკას გამოსახავენ ურთიერთქმედების დიაგრამაზე. ამისათვის სარგებლობენ მიმდევრობითობის და კოოპერაციის დიაგრამებით. ბოლოს მოახდენენ ამ სტრუქტურული და ქცევითი ელემენტების ორგანიზებას როგორც კოოპერაციისა, რომელიც შესაძლებელი იქნება დაუკავშიროთ პრეცედენტს რეალიზაციით.

თუ მექანიზმები (კოოპერაციები) გამოსახავენ პროექტირების ნიმუშებს, რომლებიც წარმოადგენენ სისტემური არქიტექტურის კონცეპტუალურად დამთავრებულ ფრაგმენტებს, კარკასები ეს არქიტექტურული ნიმუშებია, რომლებიც წარმოადგენენ გარკვეულ სფეროში დანართების გაფართოებად შაბლონებს.

კარკასი შეიცავს მექანიზმების სიმრავლეს, რომლებიც ერთობლივად მუშაობენ მოცემულ საგნობრივ სფეროში არსებული პრობლემის გადაწყვეტისათვის. ასახელებენ რა კარკასს, ამით ფაქტიურად აღწერენ არქიტექტურის ჩონჩხს მთელი თავისი მმართველი ორგანოებით, რომლებიც გაიხსნება მომხმარებლის მიერ, როცა მას სურს მოახდინოს მოცემული კარკასის ადაპტირება სასურველ კონტექსტში.

კარკასს გამოსახავენ სტერეოტიპული პაკეტის სახით. პაკეტის გახსნისას, ჩანს მექანიზმები, რომლებიც არსებობენ სისტემური არქიტექტურის ნებისმიერ სახეობაში.

3.7.2. ნახაზზე ნაჩვენებია არქიტექტურული ნიმუშის სპეციფიკაცია ***საკლასო დაფა.*** როგორც დოკუმენტაციიდან ირკვევა ეს ნიმუში "განკუთვნილია საწყისი მონაცემების გარდაქმნისათვის მაღალდონიან სტრუქტურებში, რომლებსაც არა აქვთ მარტივი დეტერმინირებული გადაწყვეტა".



ნახ.3.7.2.

არქიტექტურას საფუძველად უდევს პროექტირების ნიმუში ***საკლასო დაფა***, რომელიც განსაზღვრავს კლასების ***ცოდნის წყარო, საკლასო დაფა*** და ***მაკონტროლებელი*** ერთობლივ მუშაობას. კარკასი კიდევ შეიცავს პროექტირების ნიმუშს ***ლოგიკური დასკვნის პროცესორი***, რომელიც განსაზღვრავს კლასის ***ცოდნის წყარო*** მუშაობის საერთო მექანიზმს. ბოლოს, როგორც ნახაზიდან ჩანს, კარკასი ხსნის გამოყენების ერთ ვარიანტს ***გამოვიყენოთ ცოდნის ახალი წყაროები***, რომელიც განუმარტავს კლიენტს, თუ როგორ შეიძლება მისი ადაპტირება.

მთლიანად კარკასის მოდელირება სისტემის არქიტექტურის მოდელირებაზე მარტივი ამოცანა არ არის. გარკვეულ მიმართებებში უფრო რთულიც არის, რამდენადაც იმისათვის, რომ კარკასთან შესაძლებელი იყოს მუშაობა, უნდა გაიხსნას მისი მართვისა და შეთავსების ყველა ელემენტი, ასევე წარმოვადგინოთ მეტა-პრეცედენტები, რომლებიც ხსნიან, თუ როგორ აიწყობა კარკასი და უბრალო პრეცედენტები, რომლებიც ხსნიან მის ქცევას.

თავი 4

სისტემის რეალიზება

ავტომატიზებული სისტემის პროექტირებისას საჭირო ხდება განვიხილოთ როგორც ლოგიკური, ისე მისი ფიზიკური ასპექტები. ლოგიკურ ელემენტებს განეკუთვნებიან ისეთი არსები, როგორიც არის კლასები, ინტერფეისები, კოოპერაციები, ურთიერთქმედებები და ავტომატები, ხოლო ფიზიკურს – კომპონენტები (ლოგიკური არსების ფიზიკური დაჯგუფება) და კვანძები (აპარატურა, რომელზედაც განლაგდებიან და სრულდებიან კომპონენტები).

4.1. კომპონენტები. კომპონენტების დიაგრამა

ზემოთ განხილული ყველა დიაგრამები გამოსახავდნენ სისტემის მოდელის აგების კონცეპტუალურ ასპექტებს და მიეკუთვნებოდნენ წარმოდგენის ლოგიკურ დონეს. ლოგიკური წარმოდგენის დამახასიათებელი ნიშანია ის, რომ იგი ოპერირებს ცნებებით, რომლებსაც არა აქვთ დამოუკიდებელი მატერიალური განხორციელება. სხვა სიტყვებით, ლოგიკური წარმოდგენის სხვადასხვა ელემენტები, როგორიც არის კლასები, ასოციაცია, მდგომარეობა, შეტყობინება, არ არსებობენ მატერიალურად ან ფიზიკურად. ისინი მხოლოდ ასახავენ ფიზიკური სისტემის სტრუქტურის გაგებას ან მისი ქცევის ასპექტებს მომხმარებლის თვალთახედვით.

ლოგიკური წარმოდგენის ძირითადი დანიშნულება მდგომარეობს იმაში, რომ მოახდინონ სისტემის მოდელის ელემენტებს შორის სტრუქტურული და ფუნქციონალური მიმართებების ანალიზი. მაგრამ კონკრეტული ფიზიკური სისტემის შესაქმნელად აუცილებელია რაიმე საშუალებებით მოხდეს ლოგიკური წარმოდგენის ყველა ელემენტების რეალიზება კონკრეტულ მატერიალურ არსებში.

იმისათვის, რომ ავხსნათ განსხვავება ლოგიკურ და ფიზიკურ წარმოდგენებს შორის, განვიხილოთ ზოგადათ რომელიმე პროგრამული სისტემის დამუშავების პროცესი. მისი ლოგიკური წარმოდგენის საწყისია ალგორითმებისა და პროცედურების სტრუქტურული სქემები, ინტერფეისების აღწერა და მონაცემთა ბაზების კონცეპტუალური სქემები. მაგრამ ამ სისტემის რეალიზებისათვის აუცილებელია დამუშავდეს პროგრამის საწყისი ტექსტი დაპროგრამების რომელიმე ენაზე(C++, Pascal, Java).

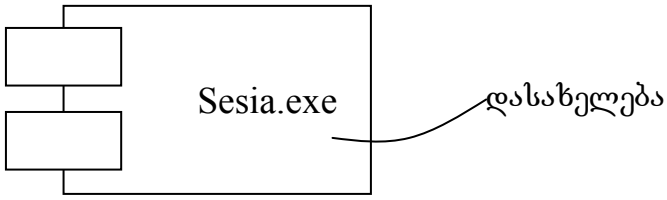
ამასთან პროგრამის საწყისი ტექსტები ჯერ კიდევ არ წარმოადგენენ პროექტის საბოლოო რეალიზაციას, თუმც კი ჩაითვლება მისი ფიზიკური რეალიზების ფრაგმენტად. პროგრამული სისტემა ჩაითვლება რეალიზებულად იმ შემთხვევაში, როცა მას შესწევს უნარი შეასრულოს თავისი მიზნობრივი დანიშნულების ფუნქციები. ხოლო ეს კი შესაძლებელია, თუ პროგრამული კოდი რეალიზებული იქნება შესრულებადი მოდულების სახით, კლასებისა და პროცედურების ბიბლიოთეკებში, მონაცემთა ფაილებში. სწორედ ეს კომპონენტები წარმოადგენენ ფიზიკური წარმოდგენის აუცილებელ ელემენტებს.

მაშასადამე, სისტემის სრული პროგრამული პროექტი შესდგება ლოგიკური და ფიზიკური წარმოდგენების ერთობლიობისაგან, რომლებიც ერთმანეთთან შეთანხმებული უნდა იყოს.

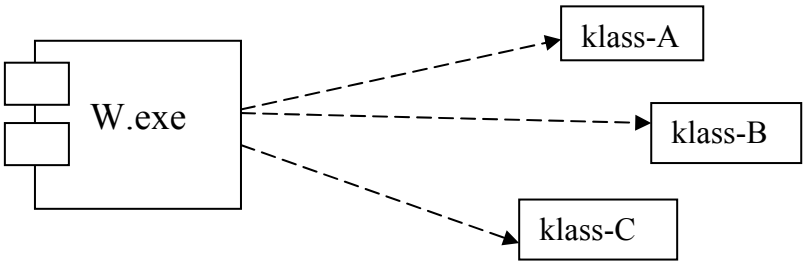
ფიზიკური არსების მოდელირებისათვის გამოიყენებიან კომპონენტები. მათ მიეკუთვნებიან შესრულებადი მოდულები, ბიბლიოთეკები, ცხრილები, ფაილები და დოკუმენტები.

ხიდს ლოგიკურ და ფიზიკურ მოდელებს შორის ქმნიან ინტერფეისები. კომპონენტი ახდენს გარკვეული ინტერფეისების რეალიზებას. შესაბამისად, ლოგიკურ მოდელში სპეციფიცირებული რომელიმე კლასის ინტერფეისი, შემდეგ რეალიზებული უნდა იქნას გარკვეული კომპონენტით.

გრაფიკულად კომპონენტი გამოისახება შემდეგი სახით



ბევრი მიმართებით კომპონენტები მსგავსია კლასების. ერთსაც და მეორესაც გააჩნია სახელი, შეუძლიათ ინტერფეისების რეალიზება, შევიდნენ დამოკიდებულების, განზოგადების და ასოციაციის მიმართებებში, მიიღონ მონაწილეობა ურთიერთქმედებაში. მაგრამ კომპონენტებსა და კლასებს შორის არის არსებითი განსხვავებაც. კერძოდ, კლასები წარმოადგენენ ლოგიკურ აბსტრაქციებს, ხოლო კომპონენტები – ფიზიკურ არსებს. ეს კი მიუთითებს, რომ კომპონენტებსა და კლასებს შორის არსებობს გარკვეული მიმართება, კერძოდ კომპონენტები ახდენენ კლასების რეალიზებას. კავშირს კომპონენტებსა და მათში რეალიზებულ კლასებს შორის წარმოადგენენ დამოკიდებულების მიმართებით. ასეთ ინფორმაციას აქვს დიდი მნიშვნელობა ლოგიკური და ფიზიკური წარმოდგენების შეთანხმებისათვის. ნახ.4.1.-ზე მოყვანილია ასეთი დამოკიდებულების ფრაგმენტი, როდესაც გარკვეული კომპონენტი დამოკიდებულია შესაბამისი კლასებისაგან.



ნახ.4.1.

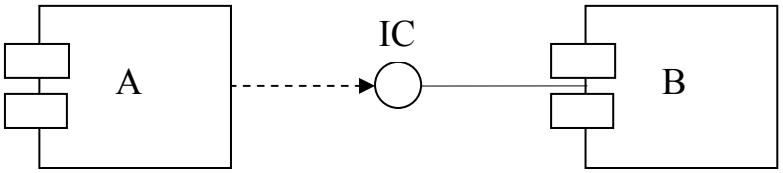
ცხადია, ცვლილებებმა კლასების აღწერის სტრუქტურაში შეიძლება გამოიწვიოს კომპონენტის შეცვლა.

მოყვანილი განმარტებიდან გამომდინარე კომპონენტები შესაძლებელია განლაგდნენ კვანძებში, ხოლო კლასები – არა. კლასებს შეიძლება გააჩნდეთ ატრიბუტები და ოპერაციები. კომპონენტები ფლობენ მხოლოდ ოპერაციებს, რომლებიც მისაღწევია მათი ინტერფეისებიდან.

ინტერფეისი, რომელიც რეალიზდება კომპონენტით, უწოდებენ ექსპორტირებულ ინტერფეისს. ეს ნიშნავს, რომ კომპონენტი მოცემული ინტერფეისიდან წარუდგენს რიგ მომსახურებებს სხვა კომპონენტებს. კომპონენტს შეუძლია ბევრი ინტერფეისების ექსპორტირება. ინტერფეისი, რომლითაც კომპონენტი სარგებლობს უწოდებენ იმპორტირებულს. ეს ნიშნავს, რომ კომპონენტი შეთავსებულია ინტერფეისთან და დამოკიდებულია მისგან თავისი ფუნქციების შესრულებისას. კომპონენტს შეუძლია სხვადასხვა ინტერფეისების იმპორტირება, ამასთან მისთვის დასაშვებია ერთდროულად ექსპორტირება და იმპორტირება.

კონკრეტული ინტერფეისი შეიძლება ექსპორტირებული იყოს ერთი კომპონენტით და იმპორტირებული მეორეს მიერ. თუ ორ კომპონენტს შორის განლაგებულია ინტერფეისი, მათი უშუალო ურთიერთდამოკიდებულება იყოფა. კომპონენტი, რომელიც იყენებს მოცემულ ინტერფეისს, იფუნქციონირებს კორექტულად (განურჩევლად იმისა, რომელი კომპონენტით იქნება რეალიზებული ინტერფეისი). ცხადია, კომპონენტი შეიძლება გამოყენებულ იქნას გარკვეულ კონტექსტში მხოლოდ მაშინ, როდესაც ყველა მისი იმპორტირებული ინტერფეისები ექსპორტირებულნი არიან რომელიღაც სხვა კომპონენტებით.

ნახ.4.2.-ზე მოყვანილია კომპონენტები, რომელთაგან B კომპონენტი ახდენს IC



ნახ.4.2.

ინტერფეისის რეალიზებას (ექსპორტირებას), ხოლო კომპონენტი A ახდენს მის იმპორტირებას (უკავშირდება მას დამოკიდებულების მიმართებით).

სისტემა შესაძლებელია შეიქმნას კომპონენტებისაგან, ხოლო შემდეგ განვავითაროთ ის, დავამატოთ რა ახალი კომპონენტები ან შევცვალოთ ძველი – დამატებითი კომპილაციის გარეშე. ამის მიღწევა შესაძლებელია ინტერფეისების

მეშვეობით. დადგინდება რა ინტერფეისი შესაძლებელია ჩაისვას მუშა სისტემაში მუშა კომპონენტი, რომელიც შეთავსებულია ამ ინტერფეისთან ან წარმოადგენს მას. სისტემა შესაძლებელია გაფართოვდეს ახალი კომპონენტების ჩართვით, რომლებიც უზრუნველყოფენ ახალ მომსახურებას დამატებითი ინტერფეისების მეშვეობით, ასევე კომპონენტებით, რომლებსაც უნარი აქვთ გამოიცნონ და გამოიყენონ ეს ახალი ინტერფეისები. ასეთი სემანტიკიდან გამომდინარე, კომპონენტი ეს სისტემის ფიზიკური შესაცვლელად დასაშვები ნაწილია, რომელიც შეთავსებულია ერთ ინტერფეისთან და რეალიზებას უწევს სხვებს.

გამოყოფენ სამი სახის კომპონენტებს:

1. **განლაგების კომპონენტები,** რომლებიც აუცილებელია და საკმარისი შესრულებადი სისტემის აგებისათვის. მათ რიცხვს მიეკუთვნება დინამიურად დამაკავშირებელი ბიბლიოთეკები (**DLL**) და შესრულებადი პროგრამები (**EXE**).
2. **კომპონენტები – მუშა პროდუქტები**. ეს დამუშავების პროცესის გვერდითი შედეგია. მას შეიძლება მივაკუთვნოთ ფაილები პროგრამის საწყისი ტექსტებით და მონაცემებით, რომლებისგანაც იქმნება განლაგების კომპონენტები. ასეთი კომპონენტები არ ღებულობენ უშუალო მონაწილეობას შესრულებადი სისტემის მუშაობაში, მაგრამ წარმოადგენენ სამუშაო პროდუქტს, რომლებისგანაც იქმნება შესრულებადი სისტემა.
3. **შესრულების კომპონენტები**. ისინი იქმნებიან როგორც სისტემის მუშაობის შედეგი.

კომპონენტები შესაძლებელია დაჯგუფებულ იქნას პაკეტში და მათ შორის შესაძლებელია დამოკიდებულების, განზოგადების, ასოციაციის და რეალიზაციის მიმართებები.

მოდელირების უნიფიცირებულ ენაში**(UML)** განსაზღვრულია ხუთი სტანდარტული სტერეოტიპი კომპონენტებთან მიმართებაში და თვითეულს შეესაბამება თავისი გრაფიკული გამოსახვა:

- **executable**(შესრულებადი) – განსაზღვრავს კომპონენტს, რომელიც შესაძლებელია შესრულდეს კვანძში.
- **Library**(ბიბლიოთეკა) – განსაზღვრავს სტატიკურ ან დინამიურ ობიექტურ ბიბლიოთეკას;
- **table**(ცხრილი) - განსაზღვრავს კომპონენტს, რომელიც წარმოადგენს მონაცემთა ბაზის ცხრილს.
- **file**(ფაილი) - განსაზღვრავს კომპონენტს, რომელიც წარმოადგენს დოკუმენტს, საწყისი ტექსტით და მონაცემებით.
- **document**(დოკუმენტი) - განსაზღვრავს კომპონენტს, რომელიც წარმოადგენს დოკუმენტს.

დამუშავებული კომპონენტების ორგანიზაციისა და მათ შორის დამოკიდებულებების ასახვისათვის გამოიყენება კომპონენტების დიაგრამა.

აქ შედის კვანძზე განლაგებული ფიზიკური არსების, მაგალითად პროგრამების, ბიბლიოთეკების, ცხრილების, ფაილების და დოკუმენტების მოდელირება. არსებითად, კომპონენტების დიაგრამა – ეს კლასების დიაგრამაა, ფოკუსირებული სისტემურ კომპონენტებზე.

გრაფიკულად კომპონენტების დიაგრამა ჩვეულებრივ შეიცავს კომპონენტებს, ინტერფეისებს და მიმართებებს მათ შორის (დამოკიდებულების, განზოგადების, ასოციაციის და რეალიზაციის).

4.2. კომპონენტების ფორმირება და მათი გამოყენება სისტემის რეალიზებისათვის

თუ სისტემა შედგება ერთი შესრულებადი ფაილისაგან, კომპონენტების მოდელირება საჭირო არ არის. ხოლო თუ სისტემა შედგება რამოდენიმე კომპონენტისაგან და მასთან ასოცირებული ობიექტური ბიბლიოთეკებისაგან, მაშინ კომპონენტების მოდელირება დაგვეხმარება ფიზიკური სისტემის აწყობისათვის.

ზემოთ მოყვანილი დაყოფიდან გამომდინარე, სისტემის რეალიზება გულისხმობს განლაგების, მუშა პროდუქტების და შესრულების კომპონენტების ფორმირებას.

ყველაზე ხშირად კომპონენტები გამოიყენებიან განლაგების კომპონენტების მოდელირებისათვის, რომლებიც შეადგენენ სისტემის რეალიზებას. განლაგების კომპონენტების ფორმირებისათვის თავდაპირველად განისაზღვრება ოპერაციული სისტემის გარემო(Windows, Linux) და შეირჩევა დაპროგრამების ენა, რომლის გარემოშიც უნდა მოხდეს კოდების აგება.

თანამედროვე ეტაპზე ყველაზე გავრცელებულია დაპროგრამების ვიზუალური ენები, რომლებიც აგებულია ობიექტ-ორიენტირებული კონცეფციის ბაზაზე. ამჟამად ასეთი მძლავრი საშუალებების შექმნაში ერთმანეთს ეჯიბრებიან ისეთი ფირმები, როგორიცაა Microsoft და Borland. Microsoft-ის პროდუქციაში აღსანიშნავია Visual Basic 7.0 და Visual C++, C# და სხვა .NET ტექნოლოგიები. ხოლო Borland-ის პროდუქციაში Delphi 5.0 და C++ Builder X. C-ის და C++ -ის სპეციალისტებმა შეიძლება გამოიყენონ Microsoft Visual C++ ან Borland C++ Builder, Pascal-ის და Object Pascal-ის სპეციალისტები უპირატესობას Borland Delphi 5.0 ანიჭებენ, ხოლო ბეისიკისა Microsoft Visual Basic 6.0-ს, ისინი დღევანდელ პროგრამულ ბაზარზე თითქმის ერთ დონეზე დგას.

Borland Delphi 5.0 სისტემას გააჩნია დიდი საშუალება მართოს მონაცემთა ბაზები, როგორც პერსონალური კომპიუტერისათვის, ასევე განაწილებული ლოკალური და გლობალური ქსელებისათვის. დაწყებული 3.0 ვერსიიდან ამ

სისტემაში დაინერგა კლიენტ\სერვერის ტექნოლოგია და შემოვიდა ახალი კომპონენტები, რომლებიც ეფექტურად უკავშირდება ქსელში მოთავსებულ მონაცემთა ბაზების სერვერებს. ესენია COM (Component Object Model) და DCOM (Distributed Component Object Model) რომლებიც საშუალებას აძლევს სხვა კომპონენტებს დაუკავშირდეს ერთმანეთს სტანდარტული COM ინტერფეისით. ბოლო 5.0 ვერსიიდან Delphi-ში დაინერგა ახალი ტექნოლოგია ADO (ActiveX Data Object), რომელიც საშუალებას იძლევა პირდაპირი კავშირით დამყარდეს კონტაქტი ნებისმიერი დონის მონაცემთა ბაზების მართვის სისტემასთან და ეს კავშირი რამდენჯერმე სწრაფია ODBC (Open DataBase Connectivity)-სთან შედარებით.

პროგრამული კოდების დამუშავება შესაძლებელია განხორციელდეს ავტომატიზებულ რეჟიმშიც. კერძოდ, პროგრამათა ტექსტური (cpp, csh, ...) და სათავო (h) ფაილები ავტომატურად იწერება თვით სისტემის მიერ, თუ გამოიყენება მაგალითად, UML–ის MsVisio ინსტრუმენტი, Visual Studio.NET პაკეტის სრული ვერსია.

პროგრამული კოდების შემდეგ მუშავდება მონაცემთა ბაზის, მისი ცხრილებისა და ინდექსური ფაილების კომპონენტები. მონაცემთა ბაზების მართვის სისტემები (მბმს) ფართოდ გავრცელებული დაპროგრამების სისტემებია, რომელთა დანიშნულებას ინფორმაციის მიღება, შენახვა და კომპიუტერის საშუალებით დამუშავება წარმოადგენს. მონაცემთა ბაზების მართვის ზოგიერთი თანამედროვე სისტემა მონაცემთა ბაზების შექმნის ძლიერი მექანიზმებითაა აღჭურვილი. მათ განეკუთვნება მონაცემთა ბაზების დაპროექტების ავტომატიზებული ინსტრუმენტული საშუალებები მრავალფანჯრიანი რეჟიმითა და შეტანილი ინფორმაციის სინტაკსური კონტროლის გათვალისწინებით.

ამჟამად გამოყენებაშია SQL Server (Standart Query Lanquaqe) მონაცემთა ბაზების მართვის სისტემა, რომელიც გამოირჩევა სიმარტივით და დიდი სისწრაფით.

ქსელური ვარიანტისთვის ასარჩევია ორი ალტერნატივა InterBase Server 7.0 ან Microsoft SQL Server 7.0.

Borland Delphi 5.0-ში არსებობს კომპონენტები, რომლებიც გათვალისწინებულია სწორედ InterBase მბმს-თვის, მაგრამ InterBase Server-ს არ გააჩნია მოქნილი დიზაინი და არც მძლავრია. სამაგიეროთ Microsoft SQL Server 7.0-ს გააჩნია დიდი სიმძლავრე და იგი გაცილებით სწრაფია.

SQL Server 7.0 წარმოადგენს პრინციპულად ახალ მიდგომას მონაცემთა ბაზების მართვაში Microsoft Foxpro, რომელიც ფუნქციონირებს Windows - გარემოში. იგი გვთავაზობს მონაცემთა ბაზების დამუშავების სრულიად ახალ სტილს, რომელიც ბევრად გაუმჯობესებული და დახვეწილია.

SQL Server 7.0-ში არსებობს სისტემის შექმნის რამდენიმე დონე. თუ სისტემა არც თუ ისე რთულია და მოკლე ვადებშია შესაქმნელი, იყენებენ Wizard-ებს შემავალი ინფორმაციის, მენიუს და სხვა ელემენტების შესაქმნელად. თუ Wizard-ი არ აკმაყოფილებს მოთხოვნებს, უფრო რთული სისტემის შესაქმნელად გამოიყენება Designer-ი ფორმების, ანგარიშების და სხვა ელემენტებისათვის. მსხვილი პროექტებისათვის უფრო მიზანშეწონილია კლასების გამოყენება.

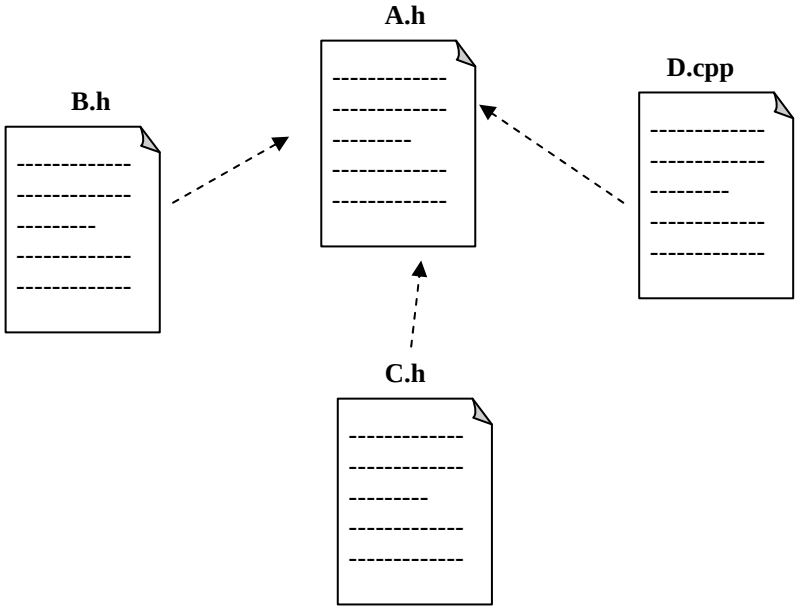
დამუშავებული კომპონენტები ერთიანდება კონტექსტურ დიაგრამებში და მზადდება სერვერ-მონაცემთა ბაზებში განსათავსებლად. ცენტრალურ სერვერ-ბაზასთან ერთად გათვალისწინებული უნდა იყოს სარეზერვო, არქივირებული ბაზის არსებობა. სერვერ-ბაზების ადმინისტრირების რეგლამენტი უნდა შემუშავებული იქნას სისტემის დანერგვის შემდეგ, გამომდინარე ფუნქციური ავტომატიზებული სამუშაო ადგილების რაოდენობისა და მენეჯმენტის მიზნებიდან.

გარდა სპეციალური პროგრამული და საბაზო ფაილების დამუშავებისა, ფართოდ გამოიყენება სტანდარტული და სერვისული ბიბლიოთეკების პროგრამები, რაც ხორციელდება ობიექტ-ორიენტირებული და სტრუქტურული დაპროგრამების პრინციპებით.

კომპონენტების ფორმირება აუცილებლად ითვალისწინებს მათ ტესტირებას და საბოლოო მუშა სახემდე მიყვანას, აგრეთვე დოკუმენტირებისა და ინსტრუქციების მომზადებას მომხმარებლებისათვის, სისტემისა და მონაცემთა ბაზების ადმინისტრატორებისათვის.

კომპონენტების დიაგრამა შესაძლებელია გამოვიყენოთ სისტემის რეალიზების სხვადასხვა დონეზე, პირველ რიგში საწყისი კოდის მოდელირებისათვის. უმრავლესობა ობიექტ- ორიენტირებულ პროგრამირების ენებში კოდი იწერება დამუშავების ინტეგრირებულ გარემოში, რომლებიც ინახავენ საწყის ტექსტებს ფაილებში. კომპონენტების დიაგრამები შესაძლებელია გამოვიყენოთ ამ ფაილების კონფიგურაციის მართვის მოდელირებისათვის, რომლებიც წარმოადგენენ კომპონენტებს – სამუშაო პროდუქტებს.

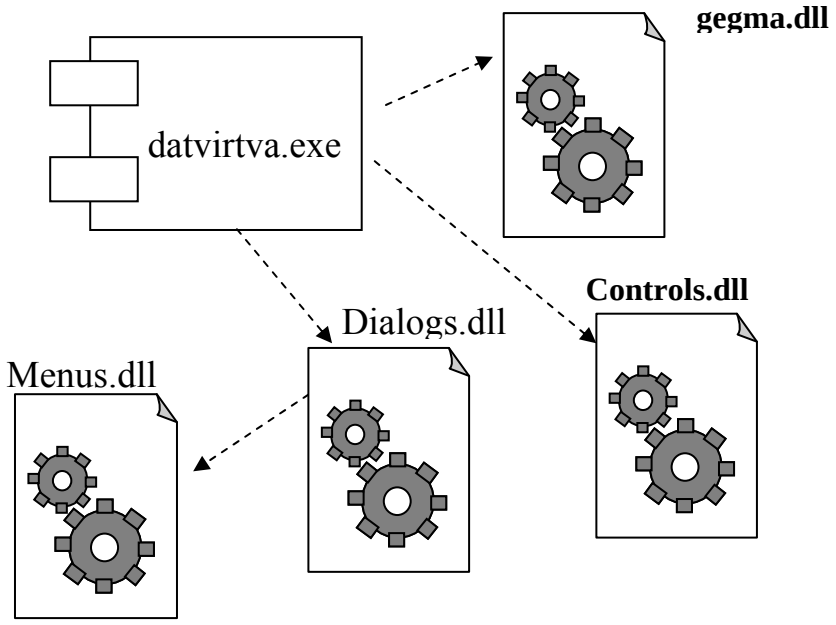
4.3. ნახაზზე ნაჩვენებია საწყისი ფაილები, რომლებიდანაც იქმნება ბიბლიოთეკა A.dll. ნახაზზე ჩანს სამი სათაო ფაილები(A.h, B.h, C.h), რომლებიც წარმოადგენენ გარკვეული კლასების სპეციფიკაციებს, ასევე ფაილი D.cpp, რომელიც წარმოადგენს ერთ-ერთი კლასის რეალიზაციას.



ნახ.4.3.

ნახ. 4.4.-ზე მოყვანილია კომპონენტების ნაკრები სასწავლო პროცესის ორგანიზებისათვის. დიაგრამა შედგება ერთი შესრულებადი პროგრამისაგან

(**datvirtva.exe**) და ბიბლიოთეკებისაგან (**gegma, Controls, Menus, Dialogs**). კომპონენტების გამოსახვისათვის გამოყენებულია სტანდარტული ელემენტები შესრულებადი პროგრამებისა და ბიბლიოთეკებისათვის.

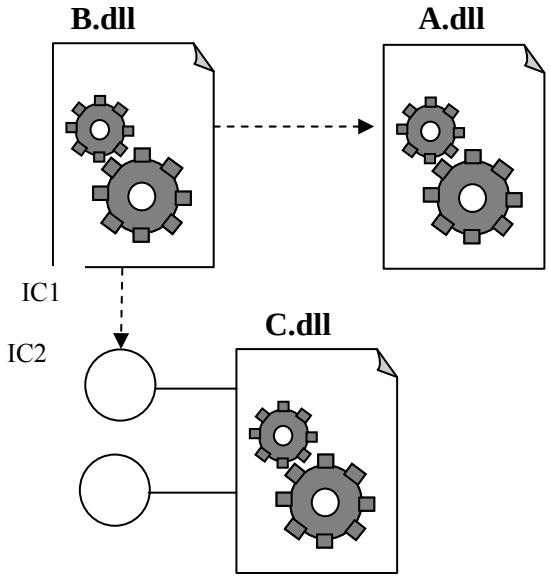


ნახ.4.4.

კომპონენტების მოდელირების მნიშვნელობა კიდევ უფრო იზრდება, თუ სისტემის განვითარებასთან ერთად საჭირო ხდება მისი სხვადასხვა ნაწილების კონფიგურაციის მართვა და ვერსიების კონტროლი. ვერსია – ეს შედარებით სრული და შეთანხმებული არტეფაქტების ნაკრებია, რომელიც წარედგინება შიდა ან გარე მომხმარებელს. სისტემისათვის, რომელიც შედგენილია კომპონენტებისაგან, ვერსია უპირველეს ყოვლისა გულისხმობს იმ ნაწილებს, რომლებიც აუცილებელია დავაყენოთ მუშა სისტემის მისაღებათ. ვერსიების მოდელირებისას კომპონენტების დიაგრამის მეშვეობით ხდება გადაწყვეტილებების ვიზუალირება, რომლებიც მიიღება სისტემის ფიზიკური შემადგენლების, ესე იგი კომპონენტების განლაგების მიმართ.

4.5. ნახაზზე მოყვანილია შესრულებადი ვერსიის ნაწილის მოდელი. ნახაზიდან ჩანს ერთი კომპონენტი A.dll, რომელიც ახდენს ინტერფეისის IC1 ექსპორტირებას, რომლის იმპორტსაც ახდენს სხვა კომპონენტი B.dll. C.dll ახდენს კიდევ ერთი ინტერფეისის ექსპორტირებას IC2, რომელიც შესაძლებელია გამოყენებულ იქნას სხვა კომპონენტების მიერ სისტემაში. დიაგრამაზე ნაჩვენებია კიდევ ერთი

კომპონენტი A.dll, რომელიც ასევე ახდენს ინტერფეისებს, თუმცა მათი დეტალები დამალულია, ნაჩვენებია მხოლოდ დამოკიდებულება B.dll –სა A.dll-გან.

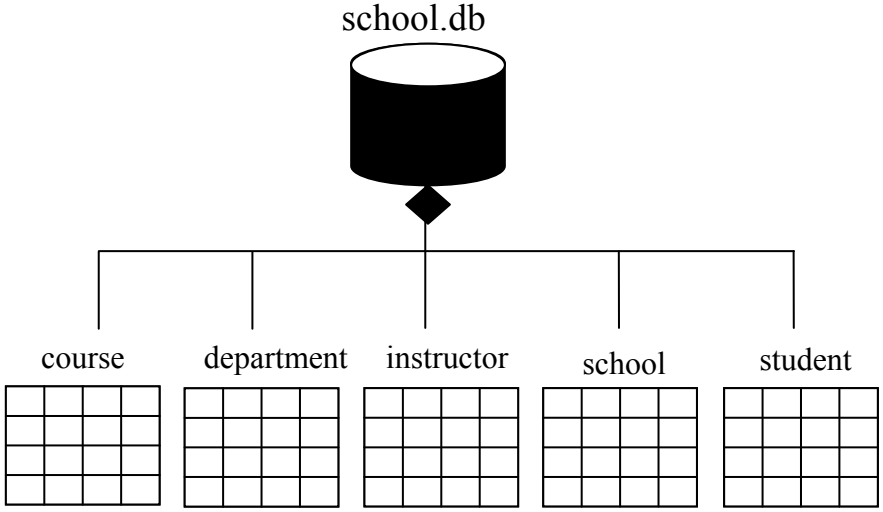


ნახ.4.5.

მნიშვნელოვანია კომპონენტების დიაგრამის როლი მონაცემთა ბაზის ფიზიკური მოდელირებისას. მონაცემთა ბაზის ლოგიკური სქემა აღწერს შენახული მონაცემების ლექსიკონს, ასევე მათ შორის კავშირის სემანტიკას. ფიზიკურად ყველაფერი ეს ინახება მონაცემთა ბაზაში. ლოგიკური სქემის ასახვა ობიექტ-ორიენტირებულ მონაცემთა ბაზაზე სიძნელეს არ წარმოადგენს, შედარებით რთულია ასახვა რელაციურზე. სიძნელე განპირობებულია იმით თუ როგორ გამოვსახოთ კლასები ცხრილების მეშვეობით და ოპერაციები, განსაზღვრულნი ლოგიკურ სქემაზე. ამ დროს ხელმძღვანელობენ შემდეგი მოსაზრებით - უბრალო ოპერაციები შექმნა, განახლება და ამოგდება რეალიზდება **SQL** da **ODBC** სტანდარტული საშუალებებით, ხოლო უფრო რთული ქცევა (მაგალითად, ბიზნეს წესები) გამოისახებიან ტრიგერებზე და შესანახ პროცედურებზე.

4.6. ნახაზზე ნაჩვენებია მონაცემთა ბაზის რამოდენიმე ცხრილი, აღებული უმაღლესი სასწავლებლის საინფორმაციო სისტემიდან. მასზე ასახულია ერთი ბაზა

(გამოსახული კომპონენტით სტერეოტიპით database), რომელიც შედგება ხუთი ცხრილისაგან(გამოსახულნი კომპონენტების სახით სტერეოტიპით interface).



ნახ.4.6.

ვიზუალირებისას კომპონენტების დიაგრამაზე კომპონენტები გამოისახებიან ცხრილის სტერეოტიპით და შესაძლებელია უჩვენოთ თითოეული ცხრილის შედგენილობა, თუმცა ეს მოცემულ მაგალითში ნაჩვენები არ არის. კომპონენტებს შესაძლებელია გააჩნდეთ ატრიბუტები, ამიტომ ფიზიკური მონაცემთა ბაზის მოდელირებისას ფართოდ გამოიყენება ატრიბუტები ცხრილის თითოეული სვეტის აღწერისათვის. ასევე კომპონენტებს შესაძლებელია ქონდეთ ოპერაციები, რომლითაც შეიძლება ვისარგებლოთ შესანახი პროცედურების აღნიშვნისათვის.

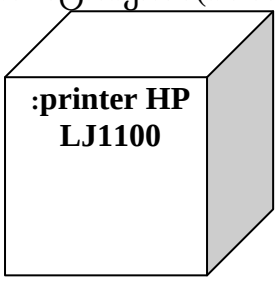
ბოლოს, კომპონენტების დიაგრამით შესაძლებელია ადაპტური სისტემების მოდელირება. ზოგიერთი სისტემები აბსოლუტურათ სტატიკურნი არიან - მათი კომპონენტები ჩნდებიან, ღებულობენ მონაწილეობას შესრულებაში, ხოლო შემდეგ ქრებიან სცენიდან. სხვები უფრო დინამიურნი არიან. ისინი შეიცავენ კომპონენტებს, რომლებიც მიგრაციას განიცდიან. ასეთი სისტემების წარმოდგენისათვის გამოიყენება კომპონენტების დიაგრამა.

დიდი სისტემებისათვის, რომლებიც რამოდენიმე კომპიუტერზე იმართება, საჭირო ხდება კომპონენტების განაწილების საშუალებების მოდელირება, დაუნიშნოთ რა თითოეულს კვანძი, რომელზედაც ის განლაგდება.

4.2. განლაგება. განლაგების დიაგრამა

კომპონენტები, რომელსაც ვიყენებთ, უნდა განლაგდნენ რომელიმე აპარატურაზე(კვანძზე), წინააღმდეგ შემთხვევაში ისინი ვერ შესრულდებიან. ავტომატიზებული სისტემა ამიტომაც შედგება პროგრამული და აპარატული უზრუნველყოფისაგან.

კვანძი – ფიზიკური ელემენტია, რომელიც არსებობს შესრულების დროს და წარმოადგენს გამოთვლით რესურს, რომელიც ჩვეულებრივ ფლობს როგორც მინიმუმ მეხსიერების გარკვეულ მოცულობას, ხოლო ხშირათ ასევე პროცესორსაც. გრაფიკულად კვანძი გამოისახება კუბის სახით. ყოველ კვანძს გააჩნია სახელი, რომელიც წარმოადგენს ტექსტურ სტრიქონს(იხ.ნახ.4.7.).



ნახ.4.7.

ობიექტების ან კომპონენტების სიმრავლე, რომლებიც მიწერილია კვანძზე როგორც ჯგუფზე, უწოდებენ განაწილების ელემენტს. კვანძებს შორის ყველაზე გავრცელებული მიმართებაა ასოციაცია. მოცემულ კონტექსტში ასოციაცია წარმოადგენს კვანძების ფიზიკურ შეერთებას.

კვანძები ძირითადათ გამოიყენებიან:

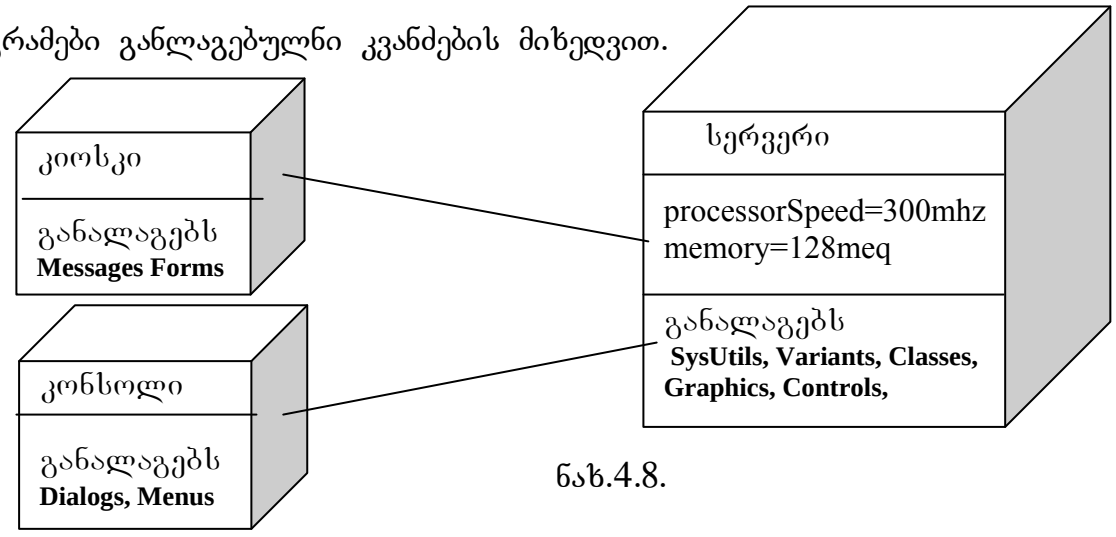
- პროცესორებისა და მოწყობილობების მოდელირებისათვის, რომლებიც ქმნიან ავტონომიური, კლიენტ-სერვერული და განაწილებული სისტემების ტოპოლოგიას.
- სისტემაში შემავალი კომპონენტების ფიზიკური განაწილების ვიზუალირებისა და სპეციფიცირებისათვის პროცესორებისა და კვანძების მიხედვით.

პროცესორი ეს კვანძია, რომელსაც შეუძლია მონაცემების დამუშავება, ანუ შეასრულოს კომპონენტი. მოწყობილობა ეს კვანძია, რომელსაც არ შეუძლია მონაცემების დამუშავება და საერთო ჯამში გამოიყენება რეალურ სამყაროსთან დაკავშირებული რაიმეს წარმოდგენისათვის.

პროცესორებისა და მოწყობილობების მოდელირებისას თავიდან ახდენენ გამოთვლითი ელემენტების იდენტიფიცირებას. თუ ეს ელემენტები წარმოადგენენ საერთო სახის პროცესორებსა და მოწყობილობებს მიუწერენ მათ შესაბამის სტანდარტულ სტერეოტიპებს. მაგრამ, თუ ეს პროცესორები და მოწყობილობები შედიან საპრობლემო სფეროს ლექსიკონში, შეუთავსებენ მათ შესაბამის სტერეოტიპებს. ისევე როგორც კლასების მოდელირებისას, დადგინდება ატრიბუტები და ოპერაციები, გამოყენებული თითოეული კვანძისათვის.

სისტემების ტოპოლოგიის მოდელირებისას ხშირათ სასარგებლოა სისტემის შემადგენლობაში შემავალ პროცესორებსა და მოწყობილობებზე კომპონენტების ფიზიკური განაწილების ვიზუალირება. ეს განსაკუთრებით შეეხება განაწილებულ სისტემებს, რათა გამოირიცხოს კომპონენტების განლაგების დუბლირების შესაძლებლობა. საკმაოდ გავრცელებულია შემთხვევა, როდესაც ერთი და იგივე კომპონენტები განლაგებულნი არიან ერთდროულად რამოდენიმე კვანძზე. ამიტომ სასურველია მიეწეროს ყოველი მნიშვნელოვანი კომპონენტი შესაბამის კვანძს ან ჩამოითვალოს კომპონენტები, განლაგებულნი კვანძზე, დამატებით განყოფილებაში.

ნახ.4.8.-ზე წარმოდგენილია დიაგრამა, რომლებზეც მოყვანილია შესრულებადი პროგრამები განლაგებულნი კვანძების მიხედვით.



ნახ.4.8.

სერვერი – ეს კვანძია სტერეოტიპით პროცესორი საერთო დანიშნულებით, კიოსკი და კონსოლი – კვანძებია სტერეოტიპით სპეციალიზირებული პროცესორები. ყოველი პროცესორისათვის გამოყოფილია დამატებითი განყოფილება, მათზე განლაგებული კომპონენტების აღნიშვნისათვის. ამასთან სერვერი წარმოდგენილია სიჩქარის და მეხსიერების ატრიბუტებით.

აუცილებელი არ არის კომპონენტები კვანძების მიხედვით განაწილებული იყვნენ სტატიურად. შესაძლებელია კომპონენტების ერთი კვანძიდან მეორეში დინამიური მიგრაციის მოდელირება.

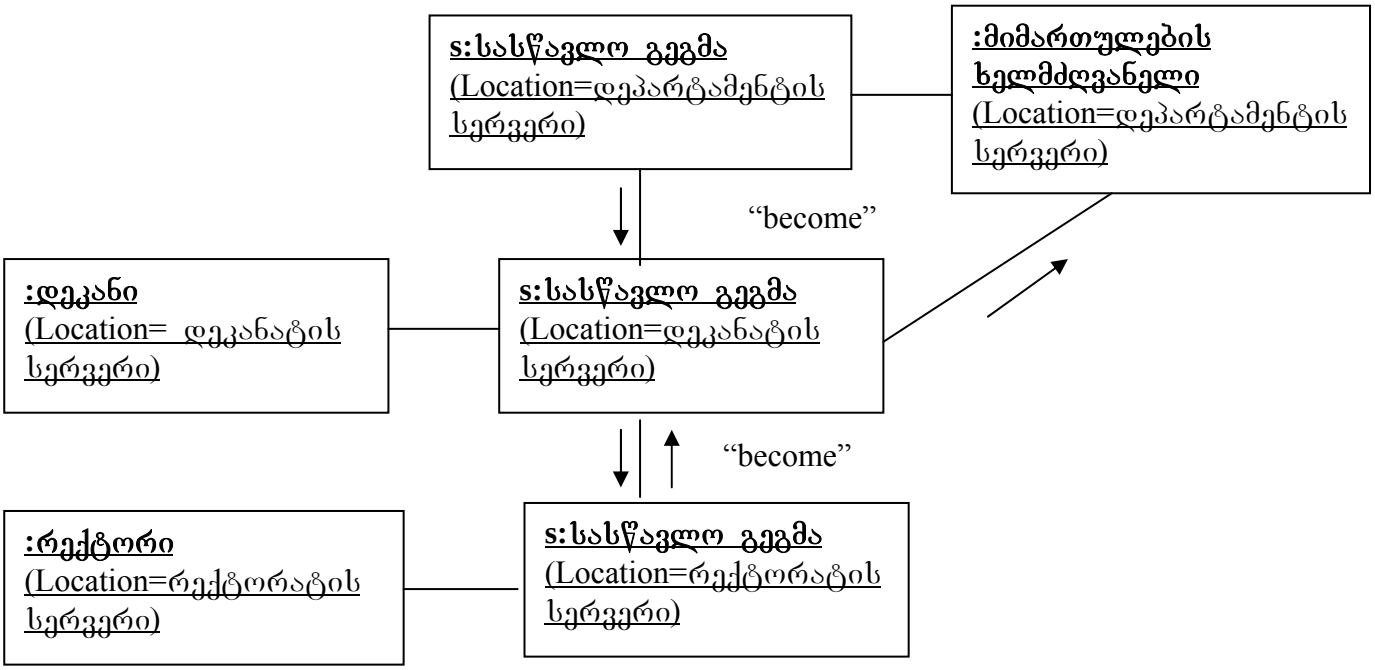
ბევრ განაწილებულ სისტემებში კომპონენტები და ობიექტები არ იცვლიან თავის ადგილმდებარეობას თავდაპირველი განლაგების შემდეგ. მაგრამ გვხვდება განაწილებული სისტემების კატეგორია, რომლებშიც სხვადასხვა არსები გადაადგილდებიან.

პირველ რიგში, ობიექტები მიგრირებენ, რათა მიუახლოვდნენ აქტიორებს და სხვა ობიექტებს, რომლებთანაც ისინი კოოპერირდებიან სამუშაოს უკეთესი შესრულებისათვის. მაგალითად, სასწავლო პროცესის ორგანიზების სისტემაში ***სასწავლო გეგმა*** გადაადგილდება დეპარტამენტიდან დეკანატში და იქიდან შესაძლებელია რექტორატში.

მეორეს მხრივ, ობიექტები მიგრაციას განიცდიან ამა თუ იმ კვანძის მწყობრიდან გამოსვლის გამო ან დატვირთვის ბალანსირებისათვის სხვადასხვა კვანძებს შორის. ამიტომ სისტემის მდგრადობის უზრუნველსაყოფად ხდება ყველა ელემენტების გადაადგილება სხვა კვანძებზე. ამ დროს შესაძლებელია წარმადობის და გამტარუნარიანობის შემცირება, მაგრამ უზრუნველყოფილი იქნება მისი მდგრადი - ნორმალური მუშაობა.

4.9. ნახაზზე მოყვანილია კოოპერაციის დიაგრამა, რომელიც წარმოადგენს სასწავლო გეგმის მიგრაციას კვანძებს შორის. კლასის ***სასწავლო გეგმის*** ეგზემპლარი (სახელით **s**) მიგრაციას განიცდის მისი დამტკიცების შესაბამისად.

გზაზე ეს ობიექტი ურთიერთქმედებს კლასების ***მიმართულების ხელმძღვანელის, დეკანის, რექტორის*** ეგზემპლიარებთან ყოველ კვანძზე და ბოლოს, დამტკიცების შედეგს აბრუნებს ობიექტზე ***მიმართულება***, რომელიც განთავსებულია კვანძზე დეპარტამენტის სერვერი.



ნახ.4.9.

კვანძების კონფიგურაციის და აგრეთვე მათზე განლაგებული კომპონენტების გამოსახვისათვის გამოიყენება განლაგების დიაგრამა. იგი საშუალებას იძლევა მოახდინოთ აპარატული საშუალებების ტოპოლოგიის მოდელირება, რომელზედაც სრულდება სისტემა. ამრიგად, განლაგების დიაგრამა შეიცავს კვანძებს და მიმართებებს (დამოკიდებულების და ასოციაციის) მათ შორის.

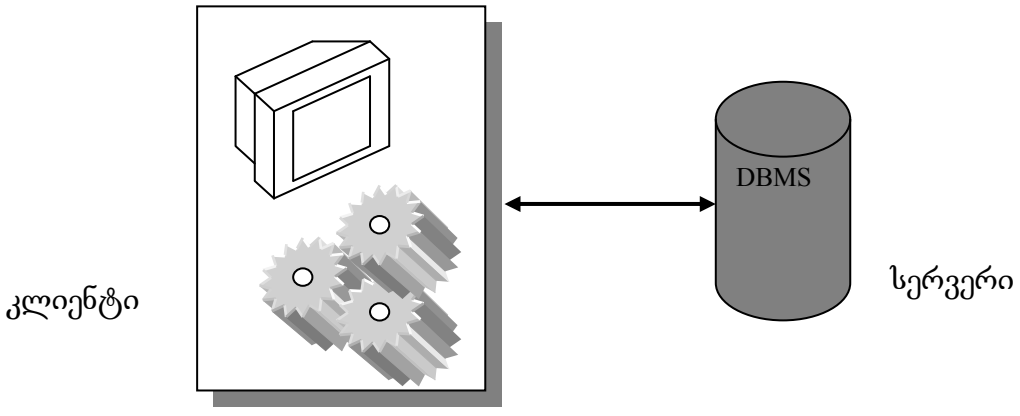
განლაგების დიაგრამას განსაკუთრებული მნიშვნელობა ენიჭება კლიენტ-სერვერული და მთლიანად განაწილებული სისტემების მოდელირებისას. კლიენტ-სერვერული სისტემა ტიპიური მაგალითია არქიტექტურის, სადაც ძირითადი ყურადღება ეთმობა მოვალეობების მკაფიო გამიჯვნას მომხმარებლის ინტერფეისსა, რომელიც არსებობს კლიენტზე, და სისტემის შენახულ მონაცემებს შორის, რომელიც არსებობენ სერვერზე. ამ დროს მნიშვნელოვანია გადაწყდეს თუ როგორ

დაკავშირდნენ კლიენტები და სერვერები ქსელით, ასევე დადგინდეს თუ როგორ არიან განაწილებულნი პროგრამული კომპონენტები კვანძებს შორის. განლაგების დიაგრამები საშუალებას იძლევიან მოახდინოთ ასეთი სისტემების ტოპოლოგიის მოდელირება.

4.2.1. კლიენტ - სერვერული სისტემების მოდელირება

თანამედროვე მიდგომით რთული ორგანიზაციული სისტემების ავტომატიზაცია უნდა განხორციელდეს კომპიუტერული ქსელის გამოყენებით, განაწილებული სამუშაო ადგილებით. განაწილებულ სისტემებში პროგრამული კომპონენტების ოპტიმალური განაწილებისათვის გამოიყენება კლიენტ – სერვერული არქიტექტურა.

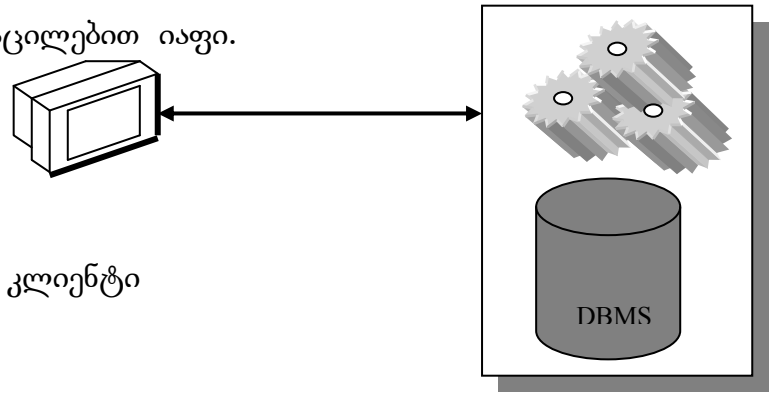
თავიდან, როდესაც კლიენტ-სერვერულმა არქიტექტურამ დაიწყო გავრცელება, ძირითადათ იყო ორ რგოლიანი სისტემები, რომლებიც შედგებოდა რამდენიმე კლიენტისა და ერთი სერვერისაგან. სერვერი ასრულებდა მხოლოდ მონაცემთა სერვერის როლს. ყველაფერი, რაც სისტემის მუშაობის ლოგიკას, ინფორმაციის სისწორეს და დაცვას ეხებოდა, თავმოყრილი იყო კლიენტურ ნაწილში. ასეთი არქიტექტურა წინგადადგმული ნაბიჯი იყო მონოლიტურ პროგრამებთან შედარებით, რადგან ასეთ სისტემებში ინფორმაცია თავმოყრილი იყო ერთ ადგილზე და ყველა კლიენტი მუშაობდა მონაცემთა ერთდაიმავე ასლთან.



ნახ.4.10.

მაგრამ იმის გამო, რომ ბიზნეს-ლოგიკა თავმოყრილი იყო კლიენტურ ნაწილში, საჭირო ხდებოდა შედარებით ძლიერი კლიენტური კომპიუტერის შეძენა, რადგან ყველა კლიენტური პროგრამა თვითონ ახდენდა როგორც ამოცანის ლოგიკასთან დაკავშირებულ გამოთვლებს, ასევე ინფორმაციის ასახვას მომხმარებლის ეკრანზე (ნახ.4.10.).

კლიენტ-სერვერული სისტემების შემდეგ თაობაში მოხდა ბიზნეს-ლოგიკის გადატანა სერვერის მხარეს. ამან გამოიწვია სერვერული რგოლის საკმაოდ გართულება და საჭირო გახდა კიდევ უფრო მძლავრი კომპიუტერი, რომელზეც იგი იმუშავებდა. სამაგიეროდ, ყველა ის კომპიუტერი, რომელზეც სისტემის კლიენტურ რგოლს უნდა ემუშავა, შეიძლება ყოფილიყო დაბალი სიმძლავრის, რადგან სისტემის კლიენტურ ნაწილს მხოლოდ ეკრანზე ინფორმაციის გამოტანა და ბრძანებების მიღება ევალებოდა (ნახ.4.11.). ამან საგრძნობლად გააიაფა ასეთი სისტემის ექსპლუატაცია, იმიტომ რომ ქსელში მხოლოდ ერთი კომპიუტერი იყო მძლავრი და შესაბამისად, შედარებით ძვირი, ხოლო დანარჩენი შეიძლებოდა ყოფილიყო გაცილებით იაფი.



კლიენტი

სერვერი

ნახ.4.11.

შემდეგი თაობის კლიენტ-სერვერულ სისტემებში ბიზნეს-ლოგიკა გამოიტანეს ცალკე, შუალედურ რგოლში. ასეთ არქიტექტურას ის უპირატესობა აქვს, რომ მისი ყველა ნაწილი მაქსიმალურად დამოუკიდებელია ერთმანეთისაგან. ამიტომ მის რომელიმე რგოლში ცვლილება არ იწვევს სხვა რომელიმე რგოლის ცვლილებას.

მაგალითად, თუ მოხდა ცვლილება პროგრამის მუშაობის ლოგიკაში, მაშინ ეს გამოიწვევს მხოლოდ შუალედური რგოლის ცვლილებას და არ შეეხება სხვა რგოლებს. აგრეთვე ასეთი სტრუქტურის დროს შესაძლებელია არსებობდეს რამოდენიმე შუალედური რგოლი და რამოდენიმე მონაცემთა რგოლი, რაც იძლევა დატვირთვის ბალანსისა და სისტემის მდგრადობის კონტროლის საშუალებას. აგრეთვე გაცილებით ადვილია ასეთი სისტემის მოდერნიზაცია, რადგან შეიძლება ახალი ფუნქციონალურობის დამატება უბრალოდ ახალი კომპონენტის დამატებით.

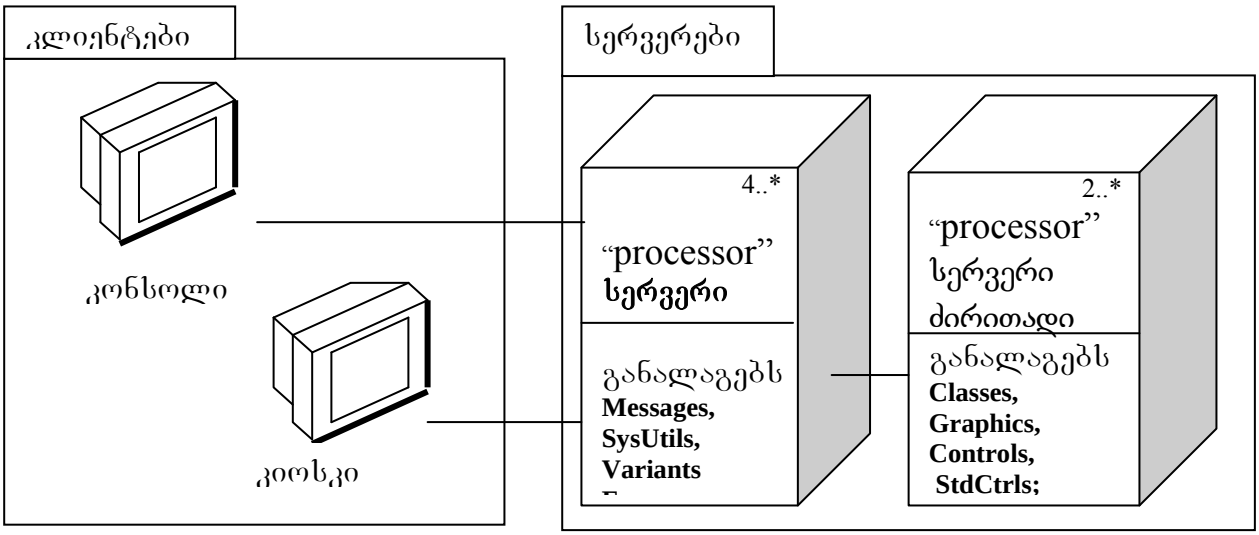
განასხვავებენ კლიენტ-სერვერული სისტემის სხვადასხვა ვარიაციებს. მაგალითად, "თხელი" კლიენტი, რომლის გამოთვლითი რესურსები შეზღუდულია და ძირითადათ დაკავებულია მომხმარებელთან ურთიერთქმედებით და ინფორმაციის ასახვით. "თხელ" კლიენტებს შეიძლება არ გააჩნდეთ საკუთარი კომპონენტები. ამის მაგიერ ისინი ტვირთავენ კომპონენტებს სერვერიდან მოთხოვნილებისამებრ. მეორეს მხრივ, შეიძლება ავირჩიოთ "მსხვილი" კლიენტი, რომელსაც გამოთვლითი რესურსები მეტი აქვს და რომელსაც ამის გამო შეუძლია დაკავდეს არა მარტო ვიზუალირებით. ამორჩევა "თხელ" და "მსხვილ" კლიენტს შორის ეს არქიტექტორული გადაწყვეტაა, რომელზეც გავლენას ახდენს სხვადასხვა ტექნიკური, ეკონომიკური და პოლიტიკური ფაქტორები.

ნებისმიერ შემთხვევაში სისტემის გაყოფისას კლიენტურ და სერვერულ ნაწილად გვიხდება გადაწყვეტილების მიღება, თუ სად განვათავსოთ ფიზიკურად კომპონენტები და როგორ უზრუნველვყოთ პასუხისმგებლობის ბალანსი მათ შორის. მაგალითად, მართვის სისტემების უმრავლესობის არქიტექტურა ინფორმაციით სამდონიანია, ანუ მომხმარებლის ინტერფეისი, ბიზნეს-ლოგიკა და მონაცემთა ბაზა ფიზიკურად განცალკევებულია ერთმანეთისაგან. გადაწყვეტილება იმის შესახებ, თუ სად უნდა განლაგდეს ინტერფეისი და მონაცემთა ბაზა გასაგებია, ხოლო სად უნდა იმყოფებოდნენ კომპონენტები, რომლებიც ბიზნეს - ლოგიკის რეალიზებას ახდენენ, შედარებით რთულია.

განლაგების დიაგრამა შესაძლებელია გამოვიყენოთ კლიენტ-სერვერული სისტემის ტოპოლოგიის აღწერისა და დადგენისათვის, ასევე პროგრამული კომპონენტების განაწილებისათვის კლიენტსა და სერვერს შორის.

კლიენტ-სერვერული სისტემის მოდელირებისას თავიდან ახდენენ კვანძების იდენტიფიცირებას, რომლებიც წარმოადგენენ კლიენტისა და სერვერის პროცესორებს. გამოყოფენ იმ მოწყობილობებს, რომლებიც ასე თუ ისე გავლენას ახდენენ სისტემის ქცევაზე. სტერეოტიპების მეშვეობით დაამუშავებენ ვიზუალურ აღნიშვნებს პროცესორებისა და მოწყობილობებისათვის. ბოლოს, დაამოდელირებენ კვანძების ტოპოლოგიას განლაგების დიაგრამაზე.

ნახ.4.12.-ზე ნაჩვენებია სისტემის ტოპოლოგია, რომელიც შეესაბამება კლასიკურ კლიენტ-სერვერულ არქიტექტურას. როგორც ნახაზიდან ჩანს, კლიენტი და სერვერი მკაფიოდ არის გამიჯნული პაკეტების(*კლიენტი,სერვერი*) გამოყენებით. პაკეტი *კლიენტი* შეიცავს ორ კვანძს(*კონსოლი, კიოსკი*). პაკეტი სერვერი ასევე შეიცავს ორ კვანძს(*სერვერი, ძირითადი სერვერი*). თითოეულზე აღწერილია კომპონენტები, რომლებიც განლაგებულია კვანზძე. ასევე, ორივე კვანძისათვის მითითებულია ჯერადობა, რომელითაც მიეთითება თუ რამდენი ეგზემპლიარია მოსალოდნელი კონკრეტულ კონფიგურაციაში.



ნახ.4.12.

4.2.2. მთლიანად განაწილებული სისტემის მოდელირება

განაწილებული სისტემები შეიძლება იყვნენ სხვადასხვა სახის - დაწყებული უბრალო ორპროცესორიანიდან დამთავრებული განშტოებული, განლაგებულნი მრავალ გეოგრაფიულად დაშორებულ კვანძებში.

უკანასკნელნი როგორც წესი არ არიან სტატიკურნი. კვანძები ჩნდებიან და ქრებიან პროცესორების მწყობრიდან გამოსვლის გამო. იქმნება ახალი, უფრო ჩქარი კავშირგაბმულების არხები, რომლებიც ფუნქციონირებენ ნელის პარალელურად, რომელთა ბოლოს და ბოლოს დემონტირებას ახდენენ. იცვლება არა მარტო სისტემის ტოპოლოგია, არამედ პროგრამული კომპონენტების განაწილება. მაგალითად, მონაცემთა ბაზის ცხრილები შეიძლება გადაადგილდეს სერვერებს შორის რათა დაუახლოვოთ ისინი ინფორმაციის მომხმარებლებს.

განაწილებული სისტემები თავისი ბუნებით შესდგებიან კომპონენტებისაგან, ფიზიკურად გაბნეულნი სხვადასხვა კვანძებზე. ბევრი სისტემებისათვის ადგილმდებარეობა (**Location**) კომპონენტების ფიქსირდება სისტემის დაყენების მომენტში. მაგრამ გვხვდება ისეთი სისტემებიც, რომლებშიც კომპონენტები მიგრირებენ ერთი კვანძიდან მეორეში.

ელემენტის ადილმდებარეობის მითითება შესაძლებელია ორი საშუალებით - მიუთითოდ კვანძის დამატებით განყოფილებაში ან ვისარგებლოთ მონიშნული მნიშვნელობით **Location** იმ კვანძის აღნიშვნისათვის, რომელზეც განლაგდება კლასი.

განაწილებული სისტემის ტოპოლოგიის მოდელირებისას მიზანშეწონილია განვიხილოთ ეგზემპლიარების როგორც კომპონენტების, ასევე კლასების ფიზიკური განლაგება. თუ ყურადღების ცენტრში არის გაშლილი სისტემის კონფიგურაციის მართვა, კომპონენტების განაწილების მოდელირება განსაკუთრებით მნიშვნელოვანია ისეთი ფიზიკური არსების სპეციფიცირებისათვის, როგორიც არის შესრულებადი მოდულები, ბიბლიოთეკები და ცხრილები. თუ უფრო აინტერესებთ

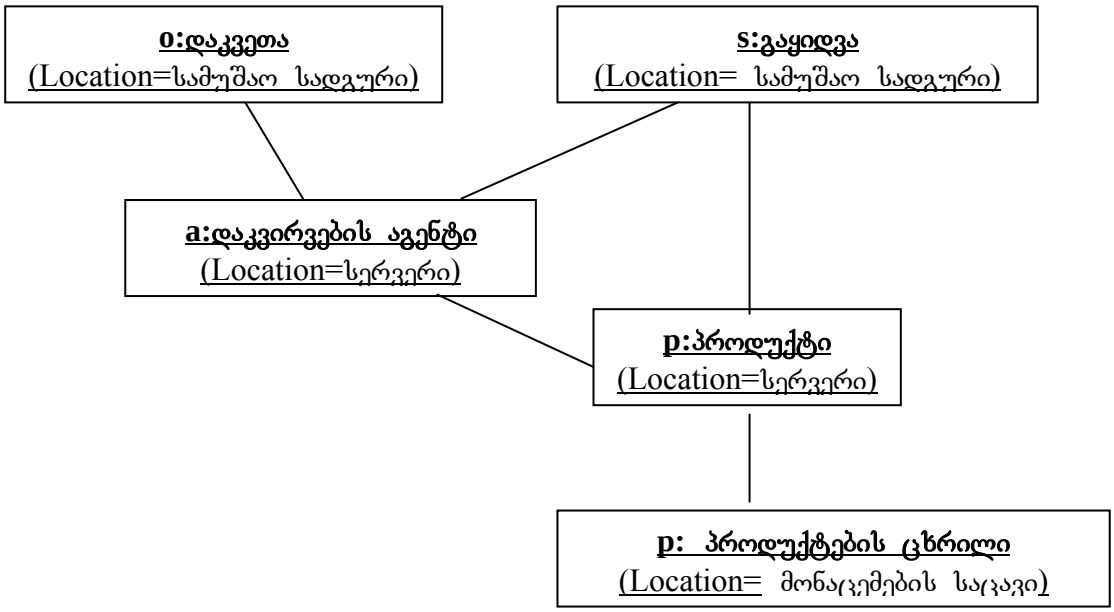
ფუნქციონალურობა, მასშტაბურობა და სისტემის გამტარუნარიანობა, მაშინ უფრო მნიშვნელოვანია ობიექტთა განაწილების მოდელირება.

გადაწყვეტილება თუ როგორ განაწილდეს ობიექტები სისტემაში მნიშვნელოვანი პრობლემაა და მჭიდროდ არის დაკავშირებული პარალელიზმის საკითხებთან.

ობიექტების განაწილების მოდელირებისას ყოველი ინტერესმქონე კლასისათვის სისტემაში განიხილავენ მიმართვების ლოკალურობას – სხვა სიტყვებით, გამოავლენენ ყველა მეზობლებს და მათ ადგილმდებარეობას. ძლიერკავშირიანი ლოკალურობა ნიშნავს, რომ ლოგიკურად მეზობლური ობიექტები იმყოფებიან გვერდიგვერდ, ხოლო სუსტკავშირიანი – რომ ისინი ფიზიკურად დაშორებულნი არიან ერთმანეთისაგან. ასევე უნდა ეცადოთ განალაგოთ ობიექტები აქტიორების გვერდით, რომლებიც მანიპულირებენ მათზე.

შემდეგ განიხილავენ ტიპიურ ურთიერთქმედებებს ურთიერთდაკავშირებულ ობიექტთა სიმრავლეებს შორის და განალაგებენ ობიექტების სიმრავლეს ურთიერთქმედების მაღალი ხარისხით გვერდიგვერდ, იმისათვის რომ შემცირდეს კომუნიკაციის ღირებულება. ობიექტები, რომლებიც სუსტად ურთიერთქმედებენ ერთმანეთს შორის, განაცალკევებენ სხვადასხვა კვანძებზე. ბოლოს, განიხილავენ პასუხისმგებლობის განაწილებას სისტემაში. გადაანაწილებენ ობიექტებს ისე, რომ დაბალანსდეს ყოველი კვანძის დატვირთვა. უნდა გათვალისწინებული იქნას მომსახურების უსაფრთხოება, მობილურობა და ხარისხი. ეს შეხედულებები უნდა აისახოს ობიექტების განლაგების დროს. ობიექტები დიაგრამაზე შესაძლებელია გამოიხატოს ერთი ორი შესაძლო საშუალებიდან:

- ჩავრთოთ ობიექტები უშუალოდ კვანძებში განლაგების დიაგრამაზე.
- ნათლად მიუთითოდ ობიექტის მდგომარეობა მონიშნული მნიშვნელობის მეშვეობით.

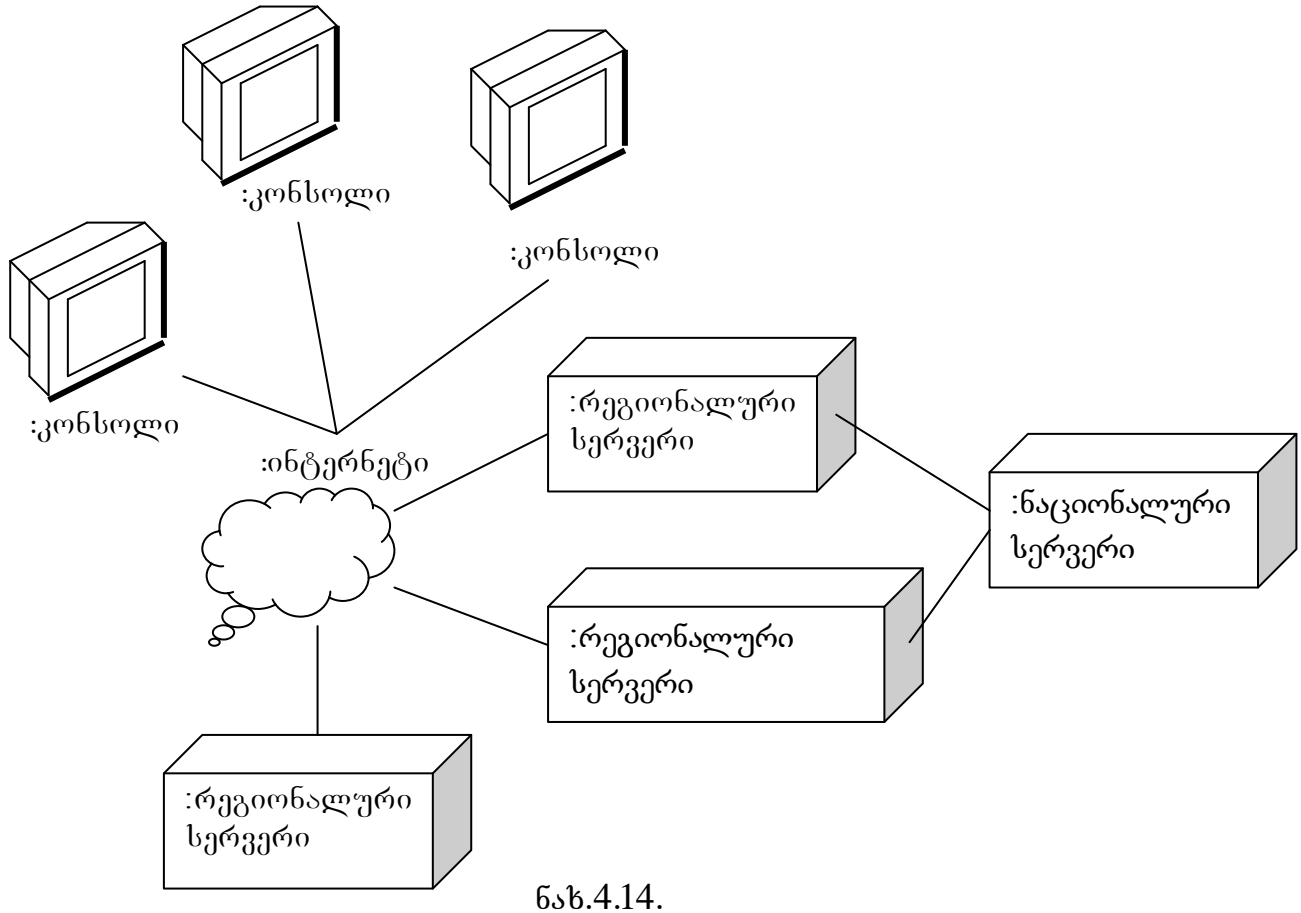


ნახ.4.13. ობიექტების განაწილების მოდელირება

4.13. ნახაზზე მოყვანილია ობიექტების დიაგრამა, რომელიც ახდენს საცალო ვაჭრობის სისტემაში ობიექტების განაწილების მოდელირებას. ამ დიაგრამის ფასი იმაშია, რომ იგი საშუალებას იძლევა მოვახდინოთ ძირითადი ობიექტების ფიზიკური განლაგების ვიზუალირება.

როგორც ჩანს, ორი ობიექტი *დაკვეთა* და *გაყიდვა* იმყოფებიან კვანძში *სამუშაო სადგური*, ორი სხვა *დაკვირვების აგენტი* და *პროდუქტი* – კვანძში *სერვერი* და ერთი *პროდუქტების ცხრილი* – კვანძში *მონაცემების საცავი*.

მთლიანად განაწილებული სისტემის მოდელირებისას მოახდენენ მოწყობილობების და პროცესორების იდენტიფიცირებას, ისევე როგორც კლიენტ-სერვერული სისტემის შემთხვევაში. განსაკუთრებული ყურადღება უნდა მიექცეს კვანძების დაჯგუფებას, რისთვისაც შესაძლებელია პაკეტების გამოყენება. წარმოადგენენ რა მოწყობილობებს და პროცესორებს განლაგების დიაგრამის სახით, ყველგან სადაც შესაძლებელია უნდა გამოყენებულ იქნას ინსტრუმენტალური საშუალებები სისტემის ქსელური ტოპოლოგიის გახსნისათვის.



ნახ.4.14.

ნახ.4.14.-ზე გამოსახულია მთლიანად განაწილებული სისტემის ტოპოლოგია. მასზე გამოსახულია სამი კონსოლი, რომლებიც დაკავშირებულნი არიან ინტერნეტთან. მეორეს მხრივ, გვაქვს სამი რეგიონალური სერვერი, რომლებიც ანხორციელებენ ინტერფეისს ნაციონალურ სერვერთან.

თ ა ვ ი 5

ავტომატიზებული სისტემის აგება და დამუშავების ორგანიზება

5.1. სისტემის არქიტექტურის აგება მოდელირების უნიფიცირებული ენის გამოყენებით

ამჟამად სტანდარტულ ინსტრუმენტს პროგრამული უზრუნველყოფის შესაქმნელად წარმოადგენს მოდელირების უნიფიცირებული ენა - Unified Modeling Language (**UML**).

UML შეიძლება გამოვიყენოთ ნებისმიერი სისტემის მოდელირებისათვის დაწყებული ინფორმაციული სისტემებიდან დამთავრებული განზოგადებული **WEB**-წინადადებით. იგი საშუალებას იძლევა განვიხილოთ სისტემა ყველა მხრიდან. მიუხედავად გამოხატვის ფართო შესაძლებლობისა ეს ენა ადვილი გასაგები და გამოსაყენებელია.

UML არ არის ვიზუალური დაპროგრამების ენა, მაგრამ მისი მეშვეობით შექმნილი მოდელები, შეიძლება უშუალოდ გადაყვანილ იქნას დაპროგრამების სხვადასხვა ენებზე. სხვა სიტყვებით რომ ვთქვათ **UML** მოდელი შეიძლება გამოვსახოთ ისეთ ენებზე როგორიც არის **JAVA, C++, Visual Basik** და რელაციურ მონაცემთა ბაზის ცხრილებზეც კი ან ობიქტ – ორიენტირებული მონაცემთა ბაზის მდგრად ობიექტებზე. ის მცნებები, რომლებიც უფრო ხელსაყრელია გადავცეთ გრაფიკულად, ასეთივე სახით წარმოიდგინებიან **UML**-ში, ხოლო ის მცნებები, რომლებიც უფრო ხელსაყრელია აღვწეროთ ტექსტური სახით, გამოისახებიან დაპროგრამების ენების მეშვეობით.

მოდელის ასეთი ასახვა დაპროგრამების ენებზე საშუალებას გვაძლევს განვახორციელოთ პირდაპირი პროექტირება- **UML** მოდელიდან კონკრეტული ენის კოდების გენერაცია. შეიძლება გადაწყდეს უკუამოცანაც- მოვახდინოთ მოდელის რეკონსტრუირება არსებული რეალიზაციიდან.

საპრობლემო სფეროს ასახვისათვის **UML**-ში ძირითადათ გამოიყენება სტრუქტურული(კლასები, პრეცედენტები, კოოპერაციები, კომპონენტები, კვანძები) და ქცევის(ურთიერთქმედება, ავტომატები) არსები, რომლებიც წარმოადგენენ ენის ძირითად ობიექტ- ორიენტირებულ ბლოკებს.

დამაკავშირებელ სამშენებლო ბლოკებს **UML**-ში წარმოადგენენ მიმართებები (დამოკიდებულება, ასოციაცია, განზოგადება, რეალიზაცია) და გამოიყენება კორექტული მოდელების-დიაგრამების შესაქმნელად.

მოდელი ეს რეალობის გამარტივებაა, აბსტრაქციაა, რომელიც იქმნება სისტემის უკეთ გაგებისათვის.

დიაგრამა **UML**-ში ეს არის გრაფიკული წარმოდგენა ელემენტების ერთობლიობისა, რომელიც გამოისახება შეკრული გრაფით, რომლის წვეროებია არსები და წიბოები კი მიმართებები.

UML წარმოადგენს ღია ენას, ანუ უშვებს კონტროლირებად გაფართოებას. **UML** - ის გაფართოების მექანიზმებია:

- სტერეოტიპები**(Stereotype) -** აფართოებს **UML**-ის ლექსიკონს იძლევა რა საშუალებას ენის არსებული ბლოკების საფუძველზე შევქმნათ ახალი, სპეციფიცირებული კონკრეტული პრობლემის გადასაწყვეტად.
- მონიშნული მნიშვნელობა(**Taged value**) - აფართოებს **UML**-ის სამშენებლო ბლოკების თვისებებს, საშუალებას იძლევა რა ჩავრთოთ ახალი ინფორმაცია ელემენტის სპეციფიკაციაში.
- შეზღუდვები(**Constraints**) - აფართოებს **UML**-ის სამშენებლო ბლოკების სემანტიკას, საშუალებას იძლევა რა განვსაზღვროთ ახალი ან შევცვალოთ არსებული წესები.

UML ტექნოლოგიით დასამუშავებელი სისტემა აღიწერება მოდელების ნაკრებით, რომლებიც შესაძლებლობის მიხედვით განიხილავენ მას სხვადასხვა თვალთახედვით.

სახე ან წარმოდგენა(View) - ეს მოდელია, რომელიც განიხილება გარკვეული თვალსაზრისით: მათში გამოსახულია ერთი არსები და გამოტოვებულია სხვები, რომლებსაც მოცემული თვალსაზრისით ინტერესი არ გააჩნიათ.

ელემენტების სიმრავლე დიდ სისტემებში შესაძლებელია დაიყოს უფრო მცირე ქვესისტემებათ, რომელთაგან თითოეული აბსტრაქციის უფრო დაბალ დონეებზე შეიძლება განხილულ იქნას როგორც დამოუკიდებელი სისტემა.

ქვესისტემა ეს წარმონაქმნია, რომელიც გამოიყენება რთული სისტემის დეკომპოზიციისათვის უფრო მარტივებათ, ერთმანეთზე სუსტად დამოკიდებულ შემადგენლებზე.

სისტემა არის ის არსი, რომელსაც ვამუშავებთ და რომლისთვისაც ვადგენთ მოდელს. სისტემა მოიცავს ყველა არტეფაქტს, რომლებიც შეადგენენ ამ არსს, მოდელისა და ამ მოდელის ელემენტების ჩათვლით, როგორიცაა კლასები და კავშირები მათ შორის.

სისტემას გამოსახავენ სტერეოტიპული პაკეტის სახით. წარმოადგენს რა სტერეოტიპულ პაკეტს, იგი ფლობს გარკვეულ ელემენტებს. თუ ჩაიხედავთ სისტემის შიგნით, შეიძლება დაინახოთ ყველა მისი მოდელი და ცალკეული ელემენტები. სისტემას შეიძლება გააჩნდეს ეგზემპლიარები (შეიძლება არსებობდეს რამოდენიმე სისტემა განლაგებულნი სხვადასხვა ადგილზე), ატრიბუტები და ოპერაციები (სისტემის გარე აქტიორებს საშუალება აქვთ იმოქმედონ სისტემაზე), პრეცედენტები, ავტომატები და კოოპერაციები. თითოეულ მათგანს შეუძლია მონაწილეობა მიიღოს სისტემის ქცევის სპეციფიცირებაში.

ის რაც აბსტრაქციის ერთ დონეზე გამოისახება სისტემად, მეორეზე უფრო მაღალ დონეზე შეიძლება განხილულ იქნას როგორც ქვესისტემა. ძირითადი დამოკიდებულება სისტემასა და ქვესისტემას შორის აგრეგირებაა. სისტემა შეიძლება შედგებოდეს ნული და მეტი ქვესისტემებისაგან. სისტემა ეს ყველაზე მაღალი დონის არსებაა მოცემულ კონტექსტში. მისი შემადგენელი ქვესისტემები ყოფენ სისტემას გადაუკვეთავ ნაწილებათ.

იმ დროს როდესაც ქვესისტემა წარმოადგენს დიდი სისტემის ელემენტების სიმრავლის დაყოფას დამოუკიდებელ ნაწილებათ, მოდელი ეს აბსტრაქციის სიმრავლის დაყოფაა, რომელიც გამოიყენება ამ სისტემის ვიზუალირებისათვის. განალაგებენ რა სისტემას ქვესისტემებათ, ახდენენ მის დამუშავებას ერთმანეთისაგან დამოუკიდებლად. სისტემის ან ქვესისტემის აბსტრაქციებს კი ყოფენ მოდელებათ იმისათვის, რომ უკეთესად გაიგონ თუ რის დამუშავებას და განლაგებას აპირებენ.

როგორც აღინიშნა ძირითადი დამოკიდებულება სისტემასა და ქვესისტემას შორის ეს აგრეგირებაა. ხოლო მიმართებების სპეციფიცირება ისეთ ელემენტებს შორის, როგორის არის კლასები, ინტერფეისები, კომპონენტები და კვანძები ეს მოდელის მნიშვნელოვანი სტრუქტურული შემადგენელია. რაც შეეხება კონცეპტუალურ კავშირებს ელემენტებს შორის, რომლებიც არსებობენ სხვადასხვა მოდელებს შორის, ისინი წარმოიდგინებიან ტრასირების მიმართების სახით. ტრასირება არ შეიძლება წარმოდგენილ იქნას ელემენტების მიმართ ერთი მოდელის ფარგლებში. ტრასირება წარმოიდგინება სტერეოტიპული დამოკიდებულებით. ტრასირების დამოკიდებულება გამოიყენება იმისათვის რათა ვუჩვენოთ გზა მოთხოვნიდან რეალიზაციამდე, რომელზედაც განლაგებულია ყველა შუალედური ვერსიები.

UML ტექნოლოგიით სისტემის არქიტექტურა ყველაზე ოპტიმალურად შესაძლებელია აღწერილი იყოს ხუთი ურთიერთ დაკავშირებული სახით ან წარმოდგენით, თითოეული მათგანი წარმოადგენს სისტემის ორგანიზაციის და სტრუქტურის ერთ-ერთ შესაძლო პროექციას და ყურადღებას ამახვილებს მისი ფუნქციონირების განსაზღვრულ ასპექტზე:

- შეხედულება პრეცედენტების თვალსაზრისით;
- შეხედულება დაპროექტების თვალსაზრისით;
- შეხედულება პროცესების თვალსაზრისით;
- შეხედულება რეალიზაციის თვალსაზრისით;

- შეხედულება განლაგების თვალსაზრისით.

თითოეული ამ ჩამონათვალიდან, შეიძლება ჩაითვალოს სავსებით დამოუკიდებელ ასპექტად, ისე რომ, პირები რომლებსაც კავშირი აქვთ სისტემის დამუშავებასთან, შეუძლიათ ყურადღება გაამახვილონ არქიტექტურის მხოლოდ იმ ასპექტების შესწავლაზე, რომლებიც უშუალოდ მათ ეხებიან. მაგრამ არ უნდა დაივიწყოთ, რომ ეს სახეები ურთიერთქმედებენ ერთმანეთში.

იმისათვის, რომ გამოისახოს სისტემა რომელიმე თვალთახედვით გამოიყენება დიაგრამები. ამჟამად განსაზღვრულია ცხრა ტიპის დიაგრამა, რომელთა კომბინირება შესაძლებელია საჭირო წარმოდგენის მისაღებათ. სისტემის სტატიკური ნაწილების განხილვისას გამოიყენება კლასების, ობიექტების, კომპონენტების და განლაგების დიაგრამები, ხოლო სისტემის დინამიურ ნაწილებთან სამუშაოდ გამოიყენება პრეცენდენტების, მიმდევრობის, კოოპერაციის, მდგომარეობის და მოღვაწეობის დიაგრამები, რომლებიც განხილული იყო წინა თავებში.

დიაგრამების აგებისათვის სარგებლობენ **MsVisio** პაკეტით. თუ **MsVisio** სამუშაო გარემოში კატეგორიაში Software and Database აირჩევა UML Model Diagram(Metric), ეკრანზე გამოვა ცალკეული დიაგრამების (Use case(Metric), Activity(Metric), Sequence(Metric), Collaboration(Metric), Static Structure(Metric), Component(Metric), Deploment(Metric), Statechart(Metric)) ასაწყობი პანელი და ცარიელი ფორმა. თითოეული დიაგრამის პანელიდან სამუშაო ფორმაზე მაოსით შესაძლებელია გადმოტანილ იქნას სქემის შესაბამისი ელემენტები სასურველი დიაგრამის ასაგებათ.

5.2. სისტემის მოდელირება სხვადასხვა წარმოდგენების საფუძველზე

სისტემების და მოდელების მოყვანილი ცნებები გამოიყენება იმ ელემენტების ორგანიზაციისათვის, რომლებიც საჭიროა სისტემის არქიტექტურის ასახვისათვის.

სისტემური არქიტექტურის მოდელირებისას, ხდება იმ გადაწყვეტილებების გაერთიანება, რომლებიც მიღებულია სისტემის მოთხოვნების, მისი ლოგიკური და ფიზიკური ელემენტების მიმართ. მოდელირდება სისტემის სტრუქტურული და ქცევითი ასპექტები, რომლებიც ახდენენ მათი სხვადასხვა წარმოდგენების ფორმირებას. ბოლოს, საჭიროა ყურადღება მიექცეს ქვესისტემების დაკავშირებას და გადაწყვეტილებების ტრასირებას, დაწყებული მოთხოვნების ფორმულირებიდან განლაგების ეტაპამდე. ყველა შემთხვევაში უნდა მოხდეს დიაგრამების დამუშავება ინკრემენტულად (უმატებთ რა თითო ახალ ფრაგმენტს ყოველ ერთ ჯერზე) და იტერაციულად (ვიმეორებთ რა პროექტირების პროცესს ყოველი ახალი გაუმჯობესების შემდეგ).

სისტემის მოდელირებისას სხვადასხვა თვალთახედვით ფაქტიურად ხდება მისი კონსტრუირება რამოდენიმე განზომილებაში. წარმოდგენის ერთობლიობის სწორი არჩევა საშუალებას იძლევა უფრო სწორად განვსაზღვროთ სისტემასთან დაკავშირებული საკითხები, გამოვლინდეს რისკი, რომელიც უნდა გათვალისწინებულ იქნას. თუ სახეები(წარმოდგენები) არჩეულია ცუდად ან ყურადღება მახვილდება მხოლოდ ერთზე სხვების მხედველობაში მიუღებლად, მაშინ იზრდება საშიშროება შეუმჩნეველი დარჩეს ისეთი საკითხები, რომელთა გაუთვალისწინებლობა ადრე თუ გვიან მთელ პროექტს დააყენებს საშიშროების წინაშე.

სისტემის მოდელირება სხვადასხვა წარმოდგენების გამოყენებით მოითხოვს ორი ამოცანის გადაწყვეტას:

- პირველ რიგში უნდა დადგინდეს წარმოდგენის რომელი სახეობები გამოხატავენ ყველაზე უკეთესათ სისტემის არქიტექტურას;
- მეორე - ყოველი ამორჩეული სახეობის მიმართ განისაზღვროს, რომელი დიაგრამებია საჭირო მისი ყველაზე არსებითი დეტალების გამოსახვისათვის.

პირველი ამოცანის გადაწყვეტისას გამოდიან დასამუშავებელი სისტემის მასშტაბებიდან. თუ მოდელირდება უბრალო დანართი, რომელიც სრულდება ერთ კომპიუტერზე, ამ დროს საჭირო იქნება:

- წარმოდგენა გამოყენებით შემთხვევათა თვალთახედვით – პრეცედენტების დიაგრამა;
- წარმოდგენა პროექტირების თვალთახედვით – კლასების დიაგრამა (სტრუქტურული მოდელირებისათვის) და ურთიერთქმედების დიაგრამა (ქცევის მოდელირებისათვის).

დანარჩენი წარმოდგენები მოცემული შემთხვევისათვის საჭირო არ არის.

თუ სისტემა აწყობილია არქიტექტურით "კლიენტი-სერვერი", სასურველი იქნება ჩაირთოს სამუშაოში კომპონენტები და განლაგება რეალიზაციის კონკრეტული ფიზიკური დეტალების მოდელირებისათვის.

რთული განაწილებული სისტემის მოდელირებისას, გამოიყენება არსებული ყველა დიაგრამა. ისინი საშუალებას იძლევიან სრულად გამოიხატოს სისტემის არქიტექტურა და პროექტთან დაკავშირებული ტექნიკური რისკი. ასეთი სისტემისათვის გამოყენებულ უნდა იქნას შემდეგი წარმოდგენები:

- წარმოდგენა პრეცედენტების თვალთახედვით – პრეცედენტების დიაგრამა და აქტიურობის დიაგრამები (ქცევის მოდელირებისათვის);
- წარმოდგენა პროექტირების თვალთახედვით – კლასების დიაგრამა (სტრუქტურული მოდელირებისათვის), ურთიერთქმედების დიაგრამა (ქცევის მოდელირება), მდგომარეობათა დიაგრამა (ქცევის მოდელირება);
- წარმოდგენა პროცესების თვალთახედვით – კლასების დიაგრამა (სტრუქტურული მოდელირებისათვის) და ურთიერთქმედების დიაგრამა (ქცევის მოდელირება).
- წარმოდგენა რეალიზაციის თვალთახედვით – კომპონენტების დიაგრამა;
- წარმოდგენა განლაგების თვალთახედვით – განლაგების დიაგრამა.

სისტემის სირთულის ზრდასთან ერთად აუცილებელი ხდება მისი დეკომპოზიცია უფრო მარტივ ქვესისტემებათ, რომელთაგან თითოეული შესაძლებელია დამუშავდეს დამოუკიდებლად, ხოლო შემდეგ თანდათან მოხდეს მათი გაერთიანება. ქვესისტემის მოდელირებისას ყოველი ქვესისტემისათვის ახდენენ კონტექსტის განსაზღვრას, ისევე როგორც ეს ხდება სისტემისათვის მთლიანად, ხოლო ყოველი ქვესისტემის არქიტექტურის მოდელირება წარმოებს ისევე როგორც ეს ხდება მთელი სისტემისათვის.

5.3. აბსტრაქციის სხვადასხვა დონეები

დამმუშავებლისათვის აუცილებელია განიხილოს სისტემა არა მარტო სხვადასხვა თვალთახედვით, არამედ აბსტრაქციის სხვადასხვა დონეებზე. მაგალითად, თუ დამუშავდა კლასები, რომლებიც მოიცავენ საპრობლემო სფეროს ლექსიკონს, მაშინ პროგრამისტისათვის საჭირო იქნება თითოეული მათგანის შესწავლა შესაბამისი ატრიბუტებით, ოპერაციებითა და მიმართებებით. მეორეს მხრივ, დამპროექტებლისათვის, რომელიც ათანხმებს საბოლოო მომხმარებელთან სისტემის გამოყენების სხვადასხვა ვარიანტებს, იგივე კლასები დასრულებული სისტემის დასჭირდებათ მაქსიმალურად გამარტივებული სახით. მაშასადამე, შეიძლება ითქვას, რომ პროგრამისტი მუშაობს შედარებით დაბალ, ხოლო ანალიტიკოსი და მომხმარებელი შედარებით მაღალ აბსტრაქციის დონეებზე. რამდენადაც დიაგრამები, არსებითად წარმოადგენენ მოდელის შემადგენელი ელემენტების გრაფიკულ წარმოდგენას, შეიძლება დაიხაზოს ერთი და იმავე მოდელის რამოდენიმე დიაგრამა, რომელთაგან თითოეული მალავს ან პირიქით, წარმოაჩენს ზოგიერთ ელემენტებს და შესაბამისად გვიჩვენებს დეტალიზაციის სხვადასხვა დონეს.

არსებობს ორი ძირითადი საშუალება სისტემის აბსტრაქციის სხვადასხვა დონეების მოდელირებისა:

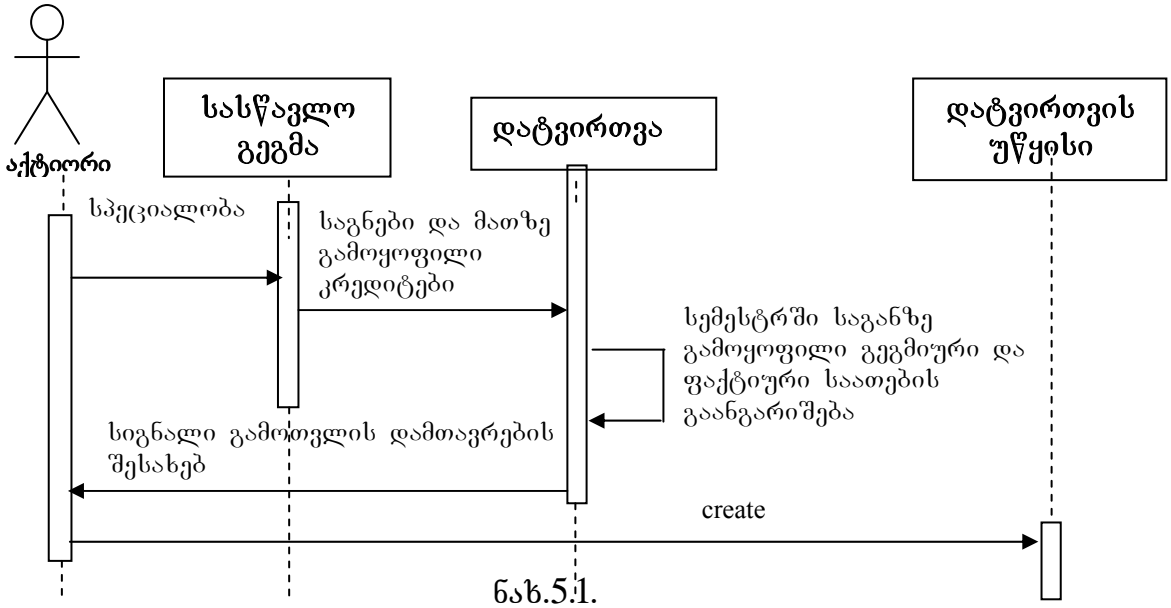
- შეიძლება აიგოს ერთი და იმავე მოდელის დიაგრამები, რომლებიც ხასიათდებიან დეტალიზაციის სხვადასხვა დონით.
- შეიქმნას სხვადასხვა მოდელები აბსტრაქციის სხვადასხვა დონით და ტრასირების დიაგრამები ერთი მოდელიდან მეორეზე.

სისტემის მოდელირება აბსტრაქციის სხვადასხვა დონეებზე დეტალიზაციის სხვადასხვა დონის დიაგრამების გამოყენებით ეყრდნობიან შემდეგ მოსაზრებას:

- თუ შესაქმნელია მოდელი რეალიზაციისათვის, მაშინ დამუშავდება ყველაზე დაბალი დონის დიაგრამები, ხოლო საბოლოო მომხმარებლისათვის ძირითადი კონცეფციების გადასაცემად, მაღალდონიანი დიაგრამები, რომელშიც დეტალების უმრავლესობა დამალულია.
- იმისგან დამოკიდებულებით, თუ რა მოთხოვნებია გასათვალისწინებელი, იქმნება დიაგრამა შესაბამის აბსტრაქციის დონეზე. რისთვისაც მალავენ ან წარმოაჩენენ არსებს, რომლებიც ქმნიან მოდელს. კერძოთ, იყენებენ მხოლოდ იმ სამშენებლო ბლოკებს და მიმართებებს მათ შორის, რომლებიც შეესაბამებიან ამ მოთხოვნებს, დანარჩენებს უგულებელყოფენ. ქცევის დიაგრამებზე უჩვენებენ მხოლოდ იმ შეტყობინებებს და გადასვლებს, რომლებსაც აქვთ მნიშვნელობა არსებული მოთხოვნების გადმოსაცემად.

მაგალითისათვის განვიხილოთ უმაღლეს სასწავლებლებში სასწავლო პროცესის ორგანიზების მოდელირების პროცესი(იხ.§ 2.1.2.). ერთ-ერთ ძირითად პრეცედენტს ასეთ სისტემაში წარმოადგენს სპეციალობის დატვირთვის დამუშავება. დამპროექტებელს ან საბოლოო მომხმარებელს დასჭირდება ურთიერთქმედების დიაგრამა აბსტრაქციის მაღალ დონეზე, რომელიც სქემატურად გამოსახავს ამ პროცესს ისე, როგორც ეს ნახ. 5.1. –ზე არის მოყვანილი.

მეორეს მხრივ, პროგრამისტი, რომელიც პასუხს აგებს ამ სცენარის რეალიზებაზე, უნდა ისარგებლოს უფრო დამუშავებული დიაგრამით, რომელშიც უნდა გაფართოვდეს ზოგიერთი შეტყობინებები და დაემატოს ახალი მოქმედი პირები. ასეთი მაგალითი მოყვანილია ნახ. 2.3.3.-ზე.



ნახ.5.1.

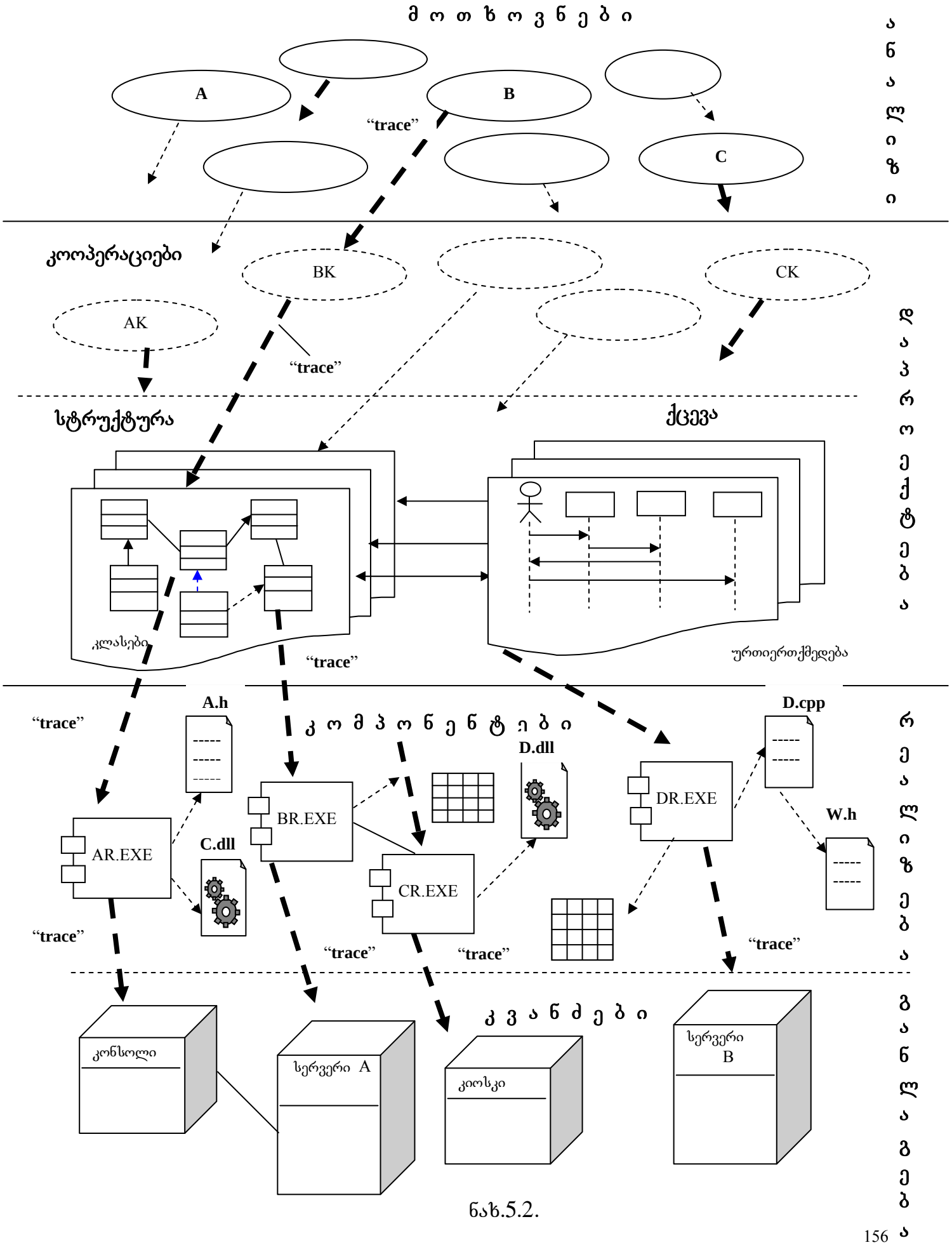
ორივე დიაგრამა მიეკუთვნება ერთი და იმავე მოდელს, მაგრამ ახდენენ სხვადასხვა დეტალიზაციის დონის დემონსტრირებას.

ძირითადი უპირატესობა მოცემული მიდგომისა იმაშია, რომ მოდელირების პროცესში გამოიყენება სემანტიკური ცნებების საერთო ნაკრები. მაგრამ უნდა იცოდეთ, რომ ცვლილებამ დიაგრამაზე აბსტრაქციის ერთ დონეზე, შეიძლება გამოიწვიოს სხვა დონეებზე არსებული დიაგრამების აქტუალობის დაკარგავა.

სისტემის პროექტირებისას სხვადასხვადონიანი მოდელების გამოყენებით ყოველი დონისათვის უნდა დამუშავდეს თავისი მოდელი. მოდელები, რომლებიც უფრო მაღალ დონეზეა, უნდა შეიცავდნენ უფრო განზოგადებულ (უბრალო) აბსტრაქციებს, ხოლო დაბალ დონეებზე – დეტალიზებულს. სხვადასხვა დონის დამაკავშირებელ ელემენტებს შორის დადგინდება ტრასირების მიმართება(იხ.ნახ.5.2.).

აქ შესაძლებელია ოთხი ტიპიური სიტუაცია:

- პრეცედენტები და მათი რეალიზაცია. მოდელში პრეცედენტები ტრასირდებიან მოდელის კოოპერაციებთან.
- კოოპერაციები და მათი რეალიზაცია. კოოპერაციები ტრასირდებიან კლასებთან, რომლებიც ერთობლივად ფუნქციონირებენ მოცემული კოოპერაციის განსახორციელებლად.


მ ო თ ხ ო ვ ნ ე ბ ი
A
B
C
"trace"
კოოპერაციები
BK
CK
AK
"trace"
სტრუქტურა
ქცევა
კლასები
"trace"
ურთიერთქმედება
"trace"
A.h
კ ო მ პ ო ნ ე ნ ტ ე ბ ი
D.dll
D.cpp
BR.EXE
DR.EXE
AR.EXE
C.dll
CR.EXE
W.h
"trace"
"trace"
"trace"
"trace"
კ ვ ა ნ ძ ე ბ ი
კონსოლი
სერვერი A
კიოსკი
სერვერი B

ნახ.5.2.

- კომპონენტები და მათი პროექტირება. მოდელის კომპონენტები ტრასირდებიან პროექტირების მოდელის ელემენტებთან.
- კვანძები და მათი კომპონენტები. კვანძები განლაგების მოდელიდან ტრასირდებიან კომპონენტებთან რეალიზაციის მოდელიდან.

უპირატესობა ასეთი მიდგომისა იმაშია, რომ სხვადასხვა დონის აბსტრაქციის დიაგრამები ნაკლებათ არიან ერთმანეთთან დაკავშირებული. ცვლილება ერთ მოდელში არ მოახდენს ძალიან დიდ გავლენას დანარჩენებზე, მაგრამ საჭირო ხდება დამატებითი ძალისხმევა მოდელების და მათი დიაგრამების სინქრონიზაციაზე. ეს განსაკუთრებით იჩენს თავს თუ მოდელები მიეკუთვნებიან სასიცოცხლო ციკლის სხვადასხვა ფაზებს.

კონცეპტუალური კავშირების მოდელირება ელემენტებს შორის, რომლებიც სხვადასხვა მოდელებში არიან, შესაძლებელია ტრასირების დამოკიდებულებით. ტრასირება წარმოიდგინება სტერეოტიპული დამოკიდებულებით. ხშირად ყურადღებას არ აქცევენ ასეთი დამოკიდებულების მიმართულებას, თუმცა ჩვეულებრივ ისარი მიუთითებს უფრო ადრეულ ობიექტს. უფრო ხშირად ტრასირების მიმართება გამოიყენება იმისათვის, რომ უჩვენონ გზა მოთხოვნიდან რეალიზაციამდე.

§ 5.3. მართვის ავტომატიზებული სისტემის დამუშავების სასიცოცხლო ციკლი

ავტომატიზებული სისტემის დამუშავება დადგენილ ვადებში, რომელიც შეესაბამება მომხმარებლის მოთხოვნებს, მოითხოვს მოწესრიგებულ მიდგომას იმასთან, თუ როგორ უნდა განაწილდნენ სამუშაოები და პასუხისმგებლობები ორგანიზაციაში, რომელიც დაკავებულია დამუშავების პროცესით. პროცესს უწოდებენ ბიჯების ნაწილობრივად მოწესრიგებულ სიმრავლეს, რომლებიც მიმართულია გარკვეული მიზნის მისაღწევად.

მართვის ავტომატიზებული სისტემის დამუშავების პროცესი თავისი ხასიათით უნდა იყოს იტერაციული, რადგან თუ გავითვალისწინებთ თანამედროვე ორგანიზაციული სისტემების სირთულესა და განშლადობას წრფივი მიდგომა დამუშავებისადმი ხდება არარეალური.

იტერაციული მიდგომა გულისხმობს პრობლემის არსში თანდათანობით შეღწევას მიმდევრობითი დაზუსტებითა და სულ უფრო მოცულობითი გადაწყვეტილების მიღებას რამდენიმე ციკლის განმავლობაში. იტერაციული მიდგომისათვის დამახასიათებელია მოქნილობა, რომელიც საშუალებას იძლევა ჩავრთოთ ბიზნეს-მიზნებში ახალი მოთხოვნები ან ტაქტიკური ცვლილებები. მისი გამოყენება შესაძლებლობას იძლევა დამუშავების ადრეულ ეტაპებზე გამოვავლინოთ და გამოვრიცხოთ რისკი, რომელიც დაკავშირებულია პროექტთან.

დამუშავების პროცესი შედგება ხუთი სამუშაო ქვეპროცესისაგან:

- ბიზნეს-მიზნების მოდელირება – აღიწერება ორგანიზაციის სტრუქტურა და დინამიკა;
- მოთხოვნების დამუშავება – აღიწერება მოთხოვნები, რომლებიც რეალიზებულ უნდა იქნას ავტომატიზაციის შედეგად;
- დაპროექტება და ანალიზი – აღიწერება სისტემის სხვადასხვა სახის არქიტექტურა;
- რეალიზაცია – ხდება პროგრამების დამუშავება და ტესტირება;
- განლაგება – მოიცავს სისტემის კონფიგურირებას;

ყოველი სამუშაო პროცესის შიგნით თავმოყრილია ერთმანეთთან დაკავშირებული არტეფაქტები და მოღვაწეობები. **არტეფაქტები** – ეს გარკვეული დოკუმენტი, ანგარიში ან შესრულებადი პროგრამაა, რომლებიც იქმნება, ხოლო შემდეგ გარდაიქმნება ან გამოიყენება. ტერმინით **მოღვაწეობა** აღიწერება ამოცანები – მოფიქრება, შესრულება, პროექტის ანალიზი, რომლებიც გადაწყდება თანამშრომლების მიერ არტეფაქტების შექმნის ან მოდიფიკაციის მიზნით, ასევე რეკომენდაციები და საშუალებები ამ ამოცანების გადასაწყვეტად.

დამუშავების პროცესი შედგება ოთხი ფაზისაგან:

1. დასაწყისი - პროექტის ბიზნეს - მიზნების განსაზღვრა.
2. გამოკვლევა - გეგმის და პროექტის არქიტექტურის დამუშავება.
3. აგება - სისტემის თანდათანობით შექმნა.
4. დანერგვა - საბოლოო მომხმარებლისათვის სისტემის დაყენება.

ფაზა – ეს დროის მონაკვეთია პროცესის ორ მნიშვნელოვან საყრდენ წერტილს შორის, რომელშიც უნდა მიღწეულ იქნას მკაფიოდ გამოხატული მიზნები, მომზადებული იქნას ესა თუ ის არტეფაქტები და მიღებული იქნას გადაწყვეტილება - საჭიროა შემდეგ ფაზაზე გადასვლა თუ არა.

საწყის სტადიაზე განისაზღვრება სისტემის მიზნები და პროექტის საზღვრები. მიზნების ანალიზი მოიცავს წარმატების კრიტერიუმის გამომუშავებას, აუცილებელ რესურსებს და გეგმის შედგენას, რომელშიც გამოხატულია ძირითადი საყრდენი წერტილები.

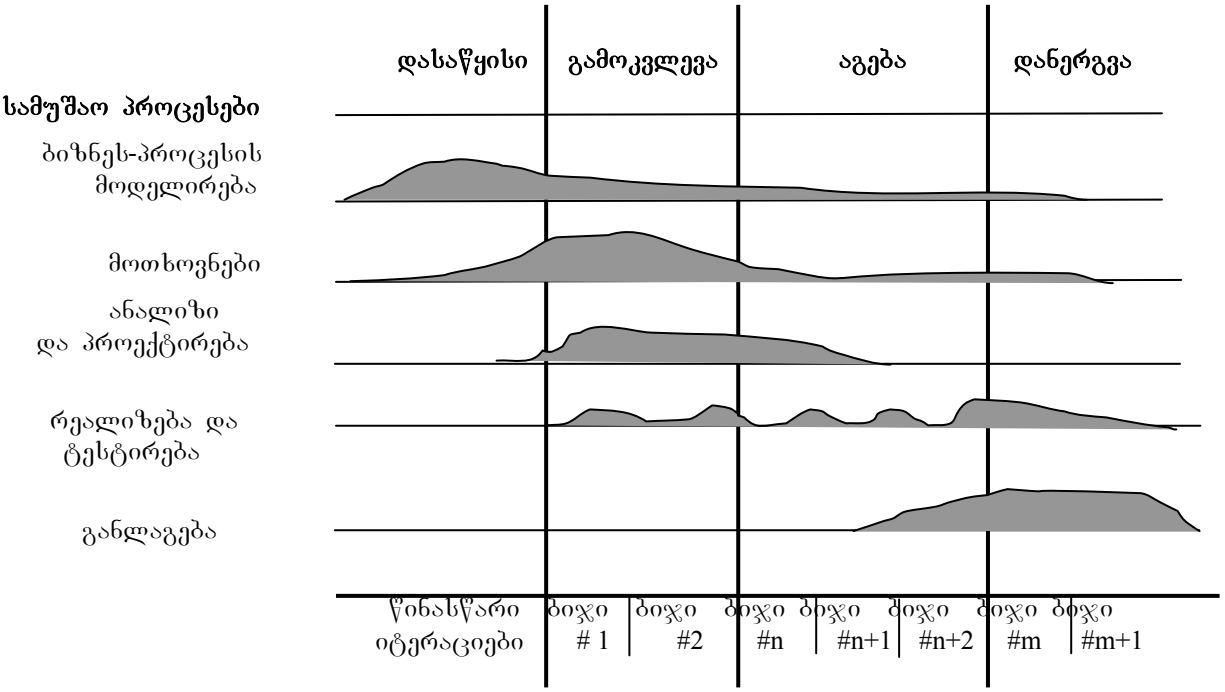
საწყისი ფაზის დასასრულს კიდევ ერთხელ ყურადღებით შეისწავლება პროექტის სასიცოცხლო ციკლი და მიიღება გადაწყვეტილება, ღირს თუ არა დავიწყოთ ფართო მასშტაბიანი დამუშავება.

გამოკვლევის ეტაპზე უნდა გაანალიზდეს საგნობრივი სფერო, დამუშავდეს სისტემის არქიტექტურული საფუძვლები, შედგეს პროექტის გეგმა და გამოირიცხოს ყველაზე საშიში რისკები. არქიტექტურული გადაწყვეტილებები მიღწეულ უნდა იქნას მაშინ, როდესაც გაირკვევა სისტემის სტრუქტურა მთლიანობაში, ე.ი. როდესაც მოთხოვნათა უდიდესი ნაწილი უკვე ფორმულირებულია.

გამოკვლევის ფაზის დასასრულს დეტალურად შეისწავლება პროექტის მიზნები, მისი საზღვრები, არჩეული არქიტექტურა და ძირითადი რისკების მართვის მეთოდები, რომლის შემდეგაც მიიღება გადაწყვეტილება იმის შესახებ, დავიწყოთ აგება თუ არა.

აგების ფაზაში თანდათან ან იტერაციულად მუშავდება პროდუქტი, რომელიც შესაძლებელი იქნება დაინერგოს. მოცემულ ეტაპზე აღიწერება დარჩენილი მოთხოვნები და მიღების კრიტერიუმები, მთავრდება დამუშავება და პროგრამული კომპლექსის ტესტირება.

აგების ფაზის დასასრულს მიიღება გადაწყვეტილება პროგრამების და მომხმარებლების დანერგვისათვის მზადყოფნის შესახებ.



ნახ.5.3. ავტომატიზებული სისტემის დამუშავების სასიცოცხლო ციკლი

დანერგვის ფაზაში პროგრამული უზრუნველყოფა გადაეცემა მომხმარებელს. ამ დროს ხშირად წარმოიშვება დამატებითი მოთხოვნის დამუშავების საკითხები სისტემის გაწყობასთან დაკავშირებით, შეცდომების გასასწორებლად, რომლებიც მანამდე არ იყო შემჩნეული და რიგი ფუნქციების საბოლოოდ გასაფორმებლად, რომელთა რეალიზებაც გადადებული იყო.

დანერგვის ფაზის ბოლოს დგინდება მიღწეულია თუ არა პროექტის მიზნები და საჭიროა თუ არა დავიწყოთ დამუშავების ახალი ციკლი.

დამუშავების პროცესის ყოველი ფაზა შეიძლება დაიყოს იტერაციებათ. იტერაცია ეს დასრულებული ეტაპია, რომლის შედეგად გამომუშავდება პროექტის ვერსია, რომელიც ახდენს დაგეგმილი ფუნქციების ნაწილის რეალიზაციას. შემდეგ ეს ვერსია იტერაციიდან იტერაციამდე ფართოვდება მზა პროდუქციის მიღებამდე. ყოველი იტერაციისას სრულდება განსაკუთრებული სამუშაო პროცესები, თუმცა სხვადასხვა ფაზებში ძირითადი დაწოლა ხდება განსხვავებულ სამუშაოებზე. საწყის ფაზაში მთავარ ამოცანას წარმოადგენს მოთხოვნათა გამომუშავება, ხოლო დანერგვის ფაზაში – განლაგება.

ოთხივე ძირითად ფაზაზე გასვლას უწოდებენ დამუშავების ციკლს. ყოველი ციკლი მთავრდება სისტემის ვერსიის გენერაციით. პირველ გასვლას ყველა ოთხივე ფაზაზე უწოდებენ დამუშავების საწყის ციკლს. თუ ამის შემდეგ მუშაობა პროექტზე არ წყდება, მაშინ მიღებული პროექტის განვითარება გრძელდება და ხელახლა გაივლის იმავე ფაზებს: საწყის, გამოკვლევის, აგების და დანერგვის. რის შედეგადაც სისტემა განიცდის ევოლუციას, ამიტომ ყოველ ციკლს, რომელიც მოსდევს საწყისს, უწოდებენ ევოლუციურს.

ლიტერატურა

1. Booch G.,Rumbaugh J.,Jacobson I., The Unified Modeling Language User Guide. Addison Wesley Longman, 1999. Addison Wesley Londman, 1999.
2. Буч Г., Рамбо Д., Джекобсон А. Язык **UML**. Руководство пользователя.// Серия "Объектно- ориентированные технологии в программировании". Москва, 2004.
3. **Леоненков. Самоучитель UML. UML** Teach Yourself. **...** Особенности реализации языка **UML** в CASE-инструментарии Rational Rose 98/2000 · Заключение · Каталог · Индекс раздела. khpi-iip.mipk.kharkiv.edu/library/case/leon/index.html - 3k -
4. სასამართლო საქმეთა წარმოების ქსელური კომპიუტერული სისტემის დამუშავების პროექტი. მ. ახობაძე, გ. გოგიჩაიშვილი, გ. სურგულაძე, თ. სუხიაშვილი, გ. ღვინეფაძე // სტუ- ს შრომები, 2004, № 1(451) გვ.100- 104.
5. სუხიაშვიული თ. სასამართლო სამოქალაქო სამართლის საქმეთა მდგომარეობათა ანალიზი სამართალწარმოების ავტომატიზებულ სისტემაში. პერიოდული სამეცნიერო ჟურნალი „ინტელექტი", №1 (18) , 2004, გვ. 43 - 47.
6. გოგიჩაიშვილი გ., სუხიაშვილი თ. სამოქალაქო სამართალწარმოების ავტომატიზებული სისტემის დამუშავება. საერთაშორისო სამეცნიერო კონფერენცია "მართვისა და ენერგეტიკის პრობლემები" **PCPE**- 2004 თბილისი. მოხსენებათა კრებული.
7. სუხიაშვილი თ. ავტომატიზებული მართვის თეორიული საფუძვლები. დამტკიცებულია სტუ-ს სამეცნიერო-ტექნიკური საბჭოს მიერ. გამომცემლობა "ტექნიკური უნივერსიტეტი", 2005, 210 გვ.
8. Сухиашвили Т. Моделирование систем на разных уровнях абстракции. МУЖДУНАРОДНАЯ ИНЖЕНЕРНАЯ АКАДЕМИЯ "Georgian Injeneering News". 2005, № 3(456), 47 – 50.

9. Sukhiashvili T.The ascertaining of classes during the object-oriented projecting . BULLETIN OF THE GEORGIAN ACADEMY OF SCIENCES. VOLUME 172 NUMBER 2 SEPTEMBER-OCTOMBER 2005, 219-221 .
10. Sukhiashvili T. Simulation of Life Cycles of Objects in Automated Control Systems. BULLETIN OF THE GEORGIAN ACADEMY OF SCIENCES. VOLUME 172 NUMBER 3 NOVEMBER-DECEMBER 2005, 407-408
11. სუხიაშვიული თ. სისტემისადმი მოთხოვნების მოდელირება ობიექტ-ორიენტირებული დაპროექტებისას. საქართველოს მეცნიერებათა აკადემიის ყოველთვიური სამეცნიერო-რეფერირებული ჟურნალი ,,მეცნიერება და ტექნოლოგიები“, №3 (18) , 2006, გვ. 43 - 47.
12. სუხიაშვიული თ. სურგულაძე გ. ორგანიზაციულ-ადმინისტრაციული მართვის განაწილებული სისტემების არქიტექტურა. პერიოდული სამეცნიერო ჟურნალი ,,ინტელექტი“, №2 (18) , 2006, გვ. 43 - 47
13. სუხიაშვილი თ., განაწილებული სისტემების მოდელირება პროცესების თვალთახედვით. სტუ-ს შრომები, 2006, № 4(437), გვ. 206-207.
14. სუხიაშვილი თ., ორგანიზაციულ-ადმინისტრაციული მართვის განაწილებული სისტემების ავტომატიზაცია. სტუ-ს შრომები, 2006, № 4(437), გვ. 206-207.
15. სუხიაშვილი თ., სამუშაო ადგილების მოდელირება მართვის განაწილებული სისტემების პროექტირებისას. სტუ-ს შრომები, 2007, № 1(2), გვ. 206-207.
16. სუხიაშვილი თ., რეალიზების მექანიზმების მოდელირება ობიექტ-ორიენტირებული პროექტირებისას. სტუ-ს შრომები, 2007, № 1(2), გვ. 115-118.
17. სუხიაშვილი თ., უმაღლეს სასწავლებლებში სასწავლო პროცესის ორგანიზების მართვის ავტომატიზებული სისტემის დამუშავება. სტუ-ს შრომები, 2007, № 2(3), გვ. 114-120.
18. სუხიაშვილი თ. სისტემების ობიექტ-ორიენტირებული ანალიზი. დამხმარე სახელმძღვანელო საკურსო პროექტის შესადგენად. დამტკიცებულია სტუ-ს

19. სუხიაშვილი თ., ავტომატიზებული მართვის თეორიული საფუძვლები., დამტკიცებულია სტუ-ს სამეცნიერო-ტექნიკური საბჭოს მიერ. გამომცემლობა "ტექნიკური უნივერსიტეტი", 2005. 210 გვ.