PAC არქიტექტურული მოდელის გამოყენება სერვერულ აპლიკაციებში

იოსებ ძმანაშვილი
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

თანამედროვე სერვერული აპლიკაციების დინამიზმი მოითხოვს შესაბამის სტრუქტურულ და არქიტექტურულ გადაწყვეტილებებს აპლიკაციების კომპლექსურობის ეფექტურად სამართავად. კომპლექსური სამომხმარებლო ინტერფეისების მხარდაჭერა, სერვისებისა და ინტეგრაციის ფენების უზრუნველყოფა განსაზღვრავს დამატებით მოთხოვნებს კონკრეტული რეალიზაციების მიმართ, რაც ბადებს მდგრადი და განვრცობადი არქიტექტურული მოდელების შემუშავების აუცილებლობას. სტატიაში განხილულია მულტიაგენტური არქიტექტურული მოდელი *წარმოდგენა-აბსტრაქცია-მართვა* (Presentation-Abstraction-Control) და მისი გამოყენების შესაძლო გზები კომპლექსურ სერვერულ აპლიკაციებში.

საკვანძო სიტყვები: არქიტექტურული მოდელები. ინტერაქტიული სისტემა. მონაცემთა სტრუქტურები. პროგრამული აპლიკაციები. მულტიაგენტური მოდელი.

1. შესავალი

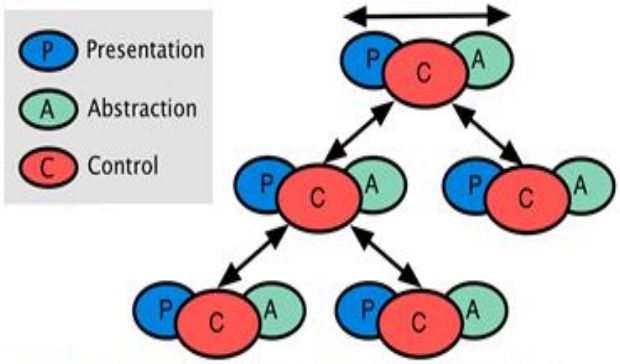
წარმოდგენა-აბსტრაქცია-მართვა (მოგვიანებით უბრალოდ PAC) განეკუთვნება არქიტექტურული მოდელების იმ კატეგორიას რომლებიც როგორც წესი გამოიყენება კომპლექსური ინტერაქტიული სისტემების ასაგებად. არსებობს ამ არქიტექტურული მოდელის რამდენიმე ვარიცია, თუმცა, ზოგადად იგი შესაძლებელია გავნმარტოთ როგორც: მოდელი რომელიც განსაზღვრავს კოოპერაციული აგენტების იერარქიულ სტრუქტურას ინტერაქტიული სისტემებისთვის. ყოველი აგენტი პასუხისმგებელია აპლიკაციის ფუნქციონალობის კონკრეტულ ასპექტზე და შედგება სამი კომპონენტისგან: წარმოდგენა, აბსტრაქცია და მართვა. აღნიშნული დაყოფა განაცალკევებს აგენტის ადამიანი-კომპიუტერი ურთიერთქმედების ასპექტებს მისი ფუნქციონალური ბირთვისა და სხვა აგენტებთან კომუნიკაციის დეტალებისგან [2].

მოცემული კონტექსტში, აგენტი აღნიშნავს ინფორმაციის დამამუშავებელ ერთეულს რომელიც შეიცავს ხდომილებათა მიმღებებსა და გადამცემებს მონაცმთა სტრუქტურებს მდგომარეობის სამართავად და პროცესორებს შემომავალ ხდომილებათა დასამუშავებლად. აგენტს გააჩნია როგორც საკუთარი მდგომარეობის განახლების, ასევე ახალ ხდომილებათა გამოწვევის უნარი. ცალკეული აგენტი შესაძლებელია წარმოდგენილი იყოს როგორც ერთი ობიექტი ან როგორც კომპლექსური და სრულყოფილი ქვესისტემა.

2. აგენტური მოდელები

აგენტურ მოდელზე დაფუძნებული ინტერაქტიული სისტემა წარმოადგენს სპეციალიზებული გამოთვლითი ერთეულების სიმრავლეს რომლებსაც ეწოდება აგენტები. ცალკეულ აგენტს გააჩნია მდგომარეობა, სპეციალიზაცია და ხდომილებების გამოწვევისა და მათზე რეაგირების უნარი. აგენტური მოდელები ეფუძნება მოდულარულ ორგანიზებას და ურთიერთქმედების მდგომარეობის განაწილებას კოოპერაციულ ერთეულებს შორის [1].

ნებისმიერი აგენტური სტილი (მაგალითად, MVC[2], PAC[2] და ა.შ.) ეყრდნობა ე.წ. პასუხისმგებლობათა გაყოფის პრინციპს [3]. აგენტურ მოდელებში განზოგადებულია განსხვავება კონცეფციასა და წარმოდგენის მექანიზმებს შორის, პასუხისმგებლობათა გაყოფის პრინციპის, აბსტრაქციისა და დეტალიზაციის ყველა დონეზე გამოყენების გზით. უფრო კონკრეტულად კი, პასუხისმგებლობათა გაყოფის პრინციპი განაწილებულია ცალკეულ კოოპერაციულ აგენტებს შორის. მაგალითად, PAC მოდელში, აგენტის კომპონენტები გამოიყენება ნებისმიერი ლოგიკური ერთეულის, განსხვავებული, მაგრამ ერთმანეთთან მჭიდრო კავშირში მყოფი ფუნქციონალური ელემენტების აღსაწერად. 1-ელ ნახაზზე მოცემულია PAC ინტერაქტიული სისტემა აგენტების იერარქიის სახით. იერარქია განსაზღვრულია აგენტებს შორის კომუნიკაციის სქემის მიხედვით, სადაც, ჰორიზონტალური ისარი განსაზღვრავს კომუნიკაციას აგენტის კომპონენტებს, ხოლო ირიბი ისარი აგენტებს შორის.



ნახ.1: PAC ინტერაქტიული სისტემა წარმოდგენილი აგენტების იერარქიის სახით

აგენტურ მოდელებს შორის არსებული მსგავსებებისა თუ განსხვავებების მიუხედავად, ნებისმიერი მათგანი განსაზღვრავს ინტერაქტიული სისტემების ჰომოგენური ტიპის მოდელირების შესაძლებლობას, სადაც, სისტემის ყველა ფუნქციონალური ასპექტი გამოსახულია ერთგვაროვანი გზით. მიუხედავად ამისა, ჰეტეროგენულობის პრობლემა ნებისმიერ შემთხვევაში გარდაუვალია რაც საჭიროებს დამატებით ანალიზს და სპეციფიური გადაწყვეტების მისადაგებას კონკრეტული ამოცანის მოთხოვნების გათვალისწინებით.

2.1. წარმოდგენა-აბსტრაქცია-მართვა

PAC არქიტექტურის ფარგლებში ინტერაქტიული სისტემა შესაძლებელია განვიხილოთ როგორც კოოპერაციული აგენეტების ერთობლიობა. მაგალითად, ადამიანისა და კომპიუტერის ურთიერთქმედებაზე სპეციალიზებული აგენტი პასუხისმგებელია მომხმარებლის მიერი შეტანილი ინფორმაციის მიღებასა და მონაცემების წარმოდგენაზე. სხვა აგენტები შესაძლებელია სპეციალიზებული იყვნენ სისტემის მონაცმეთა მოდელის დამუშავებასა და ამ მონაცემების სამართავი ფუნქციონალობის უზრუნველყოფაზე. ზოგიერთი აგენტი შესაძლებელია განკუთვნილი იყოს მხოლოდ შეცდომების დამუშავებისა ან სხვა ქვედა დონის ამოცანების გადასაჭრელად და ა.შ.

აღწერილი ტიპის კოოპერაციული აგენტების არქიტექტურაში, ყოველი აგენტი სპეციალიზებულია კონკრეტული ამოცანაზე, ხოლო, აგენტების ერთობლიობა წარმოადგენს სისტემის ერთიან ფუნქციონალობას. მოცემული არქიტექტურის ფარგლებში დასაშვებია როგორც ჰორიზონტალური, ასევე ვერტიკალური დეკომპოზიცია. PAC აგენტის შემადგენელი კომპონენტებიდან:

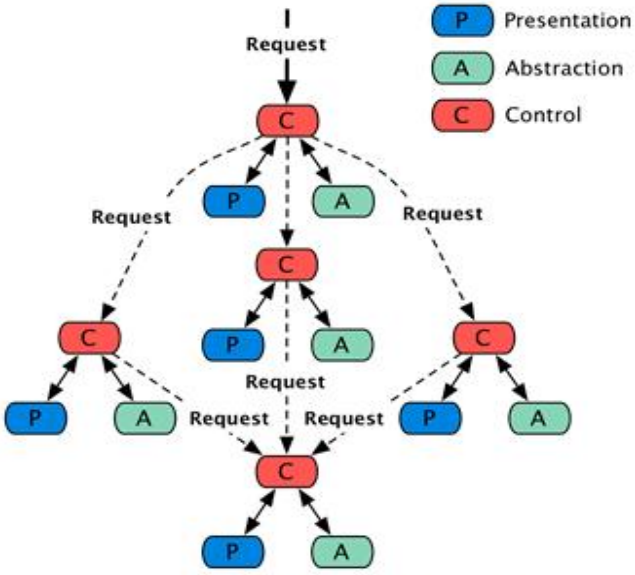
- წარმოდგენის კომპონენტი უზრუნველყოფს PAC აგენტის ვიზუალურ ქცევას;
- აბსტრაქციის კომპონენტი უზრუნველყოფს აგენტის უკან მდგომი მონაცემების მოდელის მართვასა და ამ მონაცემების ოპერირებისთვის განკუთვნილ ფუნქციონალურ მხარეს;
- მართვის კომპონენტი უზრუნველყოფს როგორც კავშირს წარმოდგენისა და აბსტრაქციის კომპონენტებს შორის, ასევე სხვა PAC აგენტებთან კომუნიკაციისთვის აუცილებელ ფუნქციონალურ მხარეს. გარდა იმისა რომ მართვის კომპონენტი პასუხისმგებლია სხვა აგენტებთან კომუნიკაციის

უზრუნველყოფაზე. განსხვავებით წარმოდგენისა და აბსტრაქციის კომპონენტებისგან იგი არის ერთადერთი სავალდებულო კომპონენტი. შესაბამისად აგენტი შესაძლებელია წარმოდგენილი იყოს მხოლოდ მართვის კომპონენტის სახით.

კონცეპტუალურად, PAC არქიტექტურაში აპლიკაცია ორგანიზებულია როგორც ქვესისტემების იერარქია და არა როგორც პასუხისმგებელი ფენები (მაგალითად, წარმოდგენის ფენა, მონაცემების ფენა, რესურსებზე წვდომის ფენა და ა.შ.). საკუთარი ფუნქციის რეალიზაციისთვის აპლიკაციის რომელიმე ერთი კონკრეტული ქვესისტემა შესაძლებელია დამოკიდებული იყოს ერთ ან რამდენიმე სხვა ქვესისტემაზე. ყოველი ქვესისტემა წარმოდგენილია როგორც ერთიანი სისტემის ერთი სპეციალიზებული ასპექტი, სადაც ურთიერთქმედება მომხმარებელსა და აპლიკაციას შორის უზრუნველყოფილია PAC აგენტის წარმოდგენის კომპონენტის მიერ, ხოლო, კომუნიკაცია PAC აგენტებს შორის ხორციელდება მხოლოდ და მხოლოდ ტრიადების მართვის კომპონენტის მეშვეობით. მართვის კომპონენტს გააჩნია სრულფყოფილი ცოდნა აგენტის აბსტრაქციისა და წარმოდგენის კომპონენტების შესახებ და უზრუნველყოფს მათ შორის კომუნიკაციას რადგან მათ ერთმანეთთან არ გააჩნია პირდაპირი კავშირი. მართვის კომპონენტის პასუხისმგებლობაში შედის როგორც ვიზუალური წარმოდგენის განახლება, ასევე მონაცემებზე წვდომა აბსტრაქციის კომპონენტის მეშვეობით, მდგომარეობის მართვა, აპლიკაციის ლოგიკის რეალიზაცია და კომუნიკაცია მშობელ და შვილობილ აგენტებთან.

მე-2 ნახაზზე წარმოდგენილია PAC აგენტების იერარქია, რომელიც აღწერს შემომავალი მოთხოვნის მიღებისა და მისი სხვა აგენტებისთვის გადამისამართების ზოგად სქემას, სადაც შემომავალი მოთხოვნა შესაძლებელია განვიხილოთ როგორც სისტემასთან ურთიერთქმედების ამსახველი მდგომარეობა.

როგორც უკვე აღვნიშნეთ, კომუნიკაცია აგენტებს შორის PAC არქიტექტურული მოდელის განუყოფელი შემადგენელი ნაწილია. PAC აგენტებს შორის მოქნილი კომუნიკაციის მექანიზმის უზრუნველსაყოფად შესაძლებელია რეგისტრაციის მექანიზმის გამოყენება, ისე როგორც განსაზღვრულია გამომქვეყნებელი-ხელმომწერი მოდელში [2]. აღნიშნული მოდელის მეშვეობით მიიღწევა კოოპერაციული კომპონენტების მდგომარეობის სინქრონიზაცია. მოდელი განსაზღვრავს ცვლილებათა გავრცელების ისეთ სქემას, სადაც ერთი გამომქვეყნებელი ატყობინებს ნებისმიერი რაოდენობის ხელმომწერს მისი მდგომარეობის ცვლილების შესახებ.



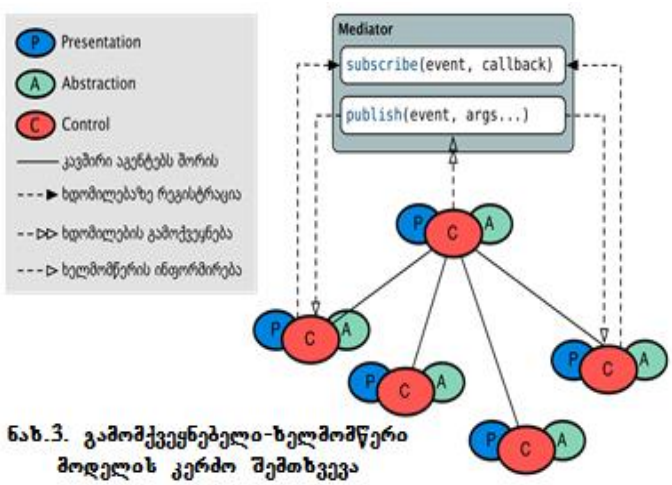
ნახ.2. PAC არქიტექტურის დიაგრამა

უფრო მეტი სიცხადისთვის შესაძლებელია განვიხილოთ შემთხვევა როდესაც აგენტი დამოკიდებულია სხვა აგენტების მიერ მართულ მონაცემებზე. ასეთ შემთხვევაში, აუცილებელია რომ სისტემას გააჩნდეს ცენტრალიზებული ცვლილებათა გავრცელების მექანიზმი, რომელიც მოიცავს სისტემის ყველა აქტიურ აგენტს იერარქიის ნებისმიერ დონეზე.

მაგალითად, აგენტის შიდა მდგომარეობის ცვლილებისას აბსტრაქციის კომპონენტეს შეუძლია გამოიწვიოს ხდომილება ინფორმაციის ცვლილების შესახებ, ხოლო, აგენტის მართვის კომპონენტმა გადაამისამართოს ეს შეტყობინება. გადამისამართების შედეგად მოხდება ყველა სხვა დამოკიდებული აგენტის(მიმდინარე აგენტის ჩათვლით) ინფორმირება გამოწვეულ ცვლილებასთან დაკავშირებით. აღსანიშნავია, რომ, ასეთი ტიპის შეტყობინებებზე რეაგირებისას, დაინტერესებულმა აგენტება შესაძლებელია განახორციელონ არა მხოლოდ მათი შიდა მდგომარეობისა ან სამომხმარებლო ინტერფეისის ელემენტების განახლება, ასევე სხვა დამატებითი ხდომილებების გამოწვევა.

მნიშვნელოვანია გავითვალისწინოთ, რომ, კომუნიკაციის ინტერფეისი უნდა იყოს ერთი და იგივე ყველა PAC აგენტისთვის რადაგან ასეთ შემთხვევაში გაცილებით მარტივია როგორც უკვე არსებული აგენტების მოდიფიცირება, ასევე, აპლიკაციისთვის სხვა ახალი აგენტების დამატება მისი განვრცობის მიზნით.

მე-3 ნახაზზე ნაჩვენებია გამომქვეყნებელი-ხელმომწერი მოდელის კერძო შემთხვევა სადაც ზედა დონის აგენტი არის გამომქვეყნებელი ხოლო ერთი საფეხურით ქვემოთ მდგომი აგენტები ხელმომწერები. იმ შემთხვევაში თუ ზედა დონის აგენტი გამოიწვევს ხდომილებას რომელსაც გააჩნია ხელმომწერები, მედიატორი კომპონენტის მიერ მომენტალურად მოხდება მათი ინფორმირება შესაბამისი უკუგამოძახების ფუნქციების შესრულებისა და აუცილებელი მონაცემების გადაცემის გზით.



ნახ.3. გამომქვეყნებელი-ხელმომწერი მოდელის კერძო შემთხვევა

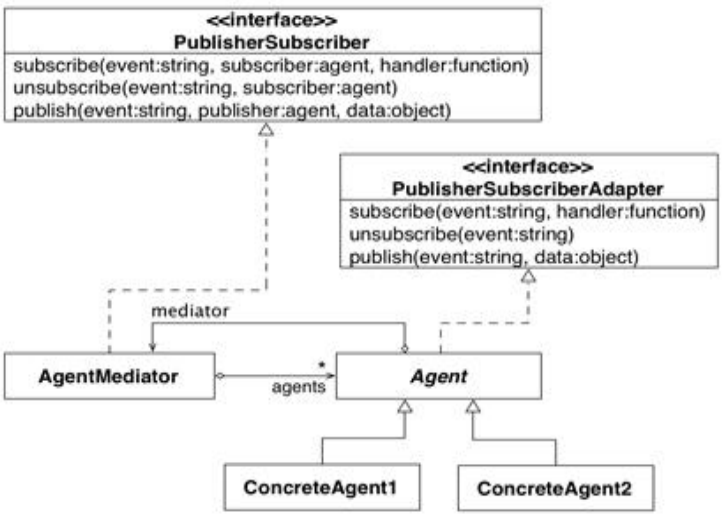
ნაჩვენები მოდელის ფარგლებში საინტერესოა, რომ ნებისმიერი აგენტი შესაძლებელია იყოს როგორც გამომქვეყნებელი, ასევე ხელმომწერი. გამომქვეყნებელ-ხელმომწერ მიდგომის სარეალიზაციოდ შესაძლებელია გამოვიყენოთ მედიატორი და დამკვირვებელი /დაქვემდებარებული (მოგვიანებით უბრალოდ დამკვირვებელი) მოდელები [4]. აღნიშნული მოდელების თვისებების გათვალისწინებით და მათი გაერთიანებით შესაძლებელია მივიღოთ ისეთი მოდელი, რომელიც სრულყოფილად უზრუნველყოფს კომუნიკაციას აგენტებს შორის. უშუალო რეალიზაციის სქემის აღწერამდე მნიშვნელოვანია განვიხილოთ ამ მოდელების ის თვისებები და მახასიათებლები რომელთა გაერთიანებითაც მივიღებთ საბოლოო სასურველ შედეგს:

- **მედიატორი** განსაზღვრავს ობიექტებს (მოგვიანებით კოლეგები) შორის ურთიერთქმედების ისეთ მექანიზმს, სადაც, კოლეგებს არ გააჩნიათ ინფორმაცია ერთმანეთის არსებობის შესახებ. მიდგომა უზრუნველყოფს კოლეგებს შორის არამჭირდრო კავშირს მათ შორის პირდაპირი კომუნიკაციის აუცილებლობის გარეშე. ამ შემთხვევაში იგულისხმება, რომ ურთიერთქმედება ხორციელდება კოლეგებსა და მედიატორს შორის, ხოლო, კოლეგების სხვა კოლეგებთან კომუნიკაციის დეტალები რეალიზებულია მედიატორის შიგნით. როგორც წესი, მედიატორი განსაზღვრავს კოლეგების რეგისტრაციის შესაძლებლობას და მას გააჩნია სრულყოფილი ცოდნა ყველა წევრისა და მათ შორის კომუნიკაციის დეტალების შესახებ;

- **დამკვირვებელი** განსაზღვრავს ერთი-მრავალთან დამოკიდებულებას ობიექტებს შორის, ისე რომ, ობიექტის მდგომარეობის ცვლილებისას ხდება ყველა სხვა დამოკიდებული ობიექტის ინფორმირება აღნიშნული მდგომარეობის ცვლილების შესახებ. ამ მოდელში განსაზღვრულია ორი როლი – დამკვირვებელი (ან ხელმომწერი) და დაქვემდებარებული (ან გამომქვეყნებელი), სადაც დამკვირვებელი არის დაინტერესებული ობიექტი ხოლო დაქვემდებარებული ობიექტი, რომელიც ვალდებულია შეატყობინოს დამკვირებელ ობიექტებს მისი მდგომარეობის ნებისმიერი ცვლილების შესახებ.

ამოცანის სპეციფიკიდან გამომდინარე მნიშვნელოვანია მოვახდინოთ აგენტებს შორის კომუნიკაციის ცენტრალიზაცია მათ შორის პირდაპირი კავშირისა და კომუნიკაციის ლოგიკის მინიმიზაციის მიზნით. ამ მოთხოვნის გათვალისწინებით უნდა შევნიშნოთ, რომ, დამკვირვებელი/დაქვემდებარებული მოდელის გამოყენება პირდაპირი გზით მიუღებელია, რადგან, მოდელი ავალდებულებს დამკვირვებელ ობიექტს დაქვემდებარებული ობიექტების არსებობის შესახებ ცოდნას. მედიატორის შემთხვევაში კონკრეტული მედიატორი ვალდებულია გააჩნდეს კოლეგებს შორის კომუნიკაციის ლოგიკის დეტალების შესახებ ცოდნა. როგორც წესი აღნიშნული დეტალი არის მედიატორის ერთერთი სუსტი მხარე რომელიც აუცილებლად უნდა იქნას გათვალისწინებული უშუალო რეალიზაციისას.

აღნიშნული პრობლემების გადაწყვეტა მარტივად არის შესაძლებელი, თუ მოვახდენთ მოდელების შესაბამის მოდიფიცირებას. მაგალითად, მედიატორსა და დამკვირვებლის გაერთიანების შემთხვევაში მივიღებთ ისეთ ცენტრალიზებულ ობიექტს, რომელიც პასუხისმგებელი იქნება როგორც დამკვირვებელი ობიექტების ინფორმირებაზე (ამ შემთხვევაში მედიატორი არის დაქვემდებარებული), ასევე, სხვა ობიექტებისგან (კოლეგებისგან) შეტყობინებათა მიღებაზე მათი შემდგომი გადამისამართების მიზნით. ჩვენი ამოცანის კონტექსტში საკმარისი იქნება ერთი ცენტრალიზებული მედიატორის არსებობა, რომელიც, როგორც უკვე აღვნიშნეთ, მის ძირითად თვისებასთან ერთად შეითავსებს დამკვირვებლის თვისებას, ხოლო აგენტი უნდა აღვჭურვოთ როგორც კოლეგისა და დაქვემდებარებულის (მედიატორთან კომუნიკაციისთვის), ასევე დამკვირვლების ფუნქციით.



ნახ.4. გამომქვეყნებელი-ხელმომწერი მოდელის სრული დიაგრამა

აღწერილი მოდელი მოცემულია მე-4 ნახაზზე UML დიაგრამის სახით, სადაც დამკვირვებელი გაერთიანებულია PublisherSubscriber ინტერფეისში. ინტერფეისი აერთიანებს როგორც დამკვირვებლის, ასევე დაქვემდებარებულის თვისებებს, რადგან მედიატორს ესაჭიროება ყველა აღნიშნული თვისება. აგენტების მედიატორთან ურთიერთქმედების სიმარტივისა და სტრუქტურული კომპლექსურობის შესამცირებლად, გამოყენებულია PublisherSubscriberAdapter ინტერფეისი მედიატორთან ურთიერთქმედების ადაპტირებისთვის, რომელიც რეალიზებულია საბაზისო Agent აბსტრაქტული კლასის მიერ [4].

3. დასკვნა

წარმოდგენილი დაიაგრამის შესაბამისი კოდის რეალიზაციის შემთხვევაში მივიღებთ მარტივ და ამავდროულად უკიდურესად მოქნილ კომუნიკაციის მოდელს, რომელიც არ არის შეზღუდული რომელიმე კონკრეტული ტიპის ცვლილებისა თუ სხვა სახის შეტყობინებების გავრცელების ფორმატით. სისტემის ყველა აქტიურ აგენტს მინიჭებული აქვს სრული თავისუფლება, როგორც სასურველი ტიპის ხდომილებათა გამოწვევის, ასევე სხვა აგენტების მიერ გამოწვეულ ხდომილებებზე რეგისტრაციისა და მათზე რეაგირების თვალსაზრისით.

ლიტერატურა:

1. Calvary G., Coutaz J., Nigay L. From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW. http://iihm.imag.fr/publs/1997/CHI97_PAC.pdf

2. Buschmann F., Meunier R., Rohnert H., Sommerlad P., Stal M. Pattern-Oriented Software Architecture Vol.1: A System of Patterns. Wiley; Vol.1 edition (August 8, 1996)

3. Dijkstra E. W Archive: On the role of scientific thought (EWD447). http://www.cs.utexas.edu /users/EWD/transcriptions/EWD04xx/EWD447.html

4. Gamma E., Helm R., Johnson R., Vlissides J.M. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional (November 10, 1994).

APPLYING PAC ARCHITECTURAL PATTERN IN SERVER APPLICATIONS

Dzmanashvili Ioseb

Georgian Technical University

Summary

Dynamic nature of modern server applications impose additional requirements on structural and architectural requirements for effectively managing application complexities. Complex user interfaces, services and integration layers require existence of robust and extensible architectural models. In this paper we discuss Presentation-Abstraction-Control (PAC) architectural model and its possible uses in complex server applications.

ПРИМЕНЕНИЕ PAC-АРХИТЕКТУРНОЙ МОДЕЛИ В СЕРВЕРНЫХ ПРИЛОЖЕНИЯХ

Дзманашвили И.

Грузинский Технический Университет

Резюме

Динамический характер современных серверных приложений требует устанавливать соответствующие структурные и архитектурные решения с целью эффективного управления комплексными приложениями. Поддержка сложных пользовательских интерфейсов, обеспечение слоев услуг и интеграции определяют допольнительные требования к конкретным реализациям, что, в свою очередь, требует необходимую разработку устойчивых и расширяемых архитектурных моделей. В предлагаемой статье обсуждается Presentation-Abstraction-Control (PAC) архитектурная модель и пути ее возможных применений в сложных серверных приложениях.