**ინტერნეტ დაპროგრამება**

**ლექცია №1**

**შესავალი**. ბოლო წლებში იმდენად გრანდიოზული ცვლილებები მოხდა კომპი­უტერულ მეცნიერებებში, მათ შორის პროგრამულ უზრუნველყოფაში, რომ ბევრ ცნობილ ინფორმატიკოსსაც უჭირდა იმის გააზრება, თუ რაშია იმ ცვლილებათა არსი, რომლებიც ეხება ინტერნეტ-სტანდარტებს. ძველი თაობის პროგრამული უზრუნველყოფის აღიარებულ სპეციალისტებს, როგორიცაა **როჯერ პრესმანი, უოტს ჰამფი** და სხვები მყარად მიიჩნევდნენ იმას, რომ ინტ. დაპროგრამებაშიც ასევე შეუცვლელად უნდა ხორციელდებოდეეს პრინციპი - პარადიგმა - ყოველი სისტემის შექმნის წინაპირობაა მისი პროექტირება, პრობლემის ღრმა შესწავლა, ამ პრობლემის გადაწყვეტის კონსტრუირება, მისი რელიზაცია და შემოწმება - დახვე­წა, რომლის დროსაც უნდა გქონდეთ კონტროლის ქვეშ შესატანი ცვლილებები და სხვა.

ახალი თაობის პროგრამული უზრუნველყოფის სპეციალისტებს, როგორიცაა **ტედ ლუისი, ბედ ადიდა** და სხვებს მიაჩნიათ, რომ ინტერნეტ სივრცეში სისტემა უნდა შეიქმნას უმცირეს დროში და მისი (ამ სისტემის, პროდუქტის) გამოყენების დრო არის მცირე, ამიტომაც არ არის საჭირო საფუძვლიანი პროექტების წამოწყება, მაშინ როცა ახალი ტექნოლოგიები, რომელიც შეიძლება ერთ თვეში შეიქმნას ამ სისტემას ისტორიის სანაგვეზე მოისვრის. ამიტომაც აქ საჭიროა სწრაფი ”ჰაკინგი” - პროგრა­მული უზრუნველყოფის შექმნა საჭიროა წინასწარი დამუშავების გარეშე და იმ პროგრამაში ოპერატიული ცვლილებების მოხდენა, რომლსაც დოკუმენტაციაა არ გააჩნია. ესეც ახალი პარადიგმა - **სწრაფი ”ჰაკინგი”**

დღესდღეობით რთულია წარმოიდგინო იზოლირებული კომპიუტერი, ის როგორც წესი ჩართულია ქსელში. ამდენად სულ უფრო და უფრო ფართო გამოყენება აქვს *განაწილებულ აპლიკაციებს*, რომელთა ნაწილები იმუშავებს სრულიად სხვადასხვა კომპიუტერებზე, ჰეტეროგენურ ქსელებში, სხვადასხვა ოპერაციულ სისტემებში სხვადასხვა ენების, ტექნოლოგიებისა და მეხსიერების გამოყენებით. ესეც პროგრა­მული უზრუნველყოფის კიდევ ერთი პარადიგმა **-*განაწილებული აპლიკაციები***.

კომპიუტერები თანდათან კარგავენ თავის ჩვეულ ფორმა, ისინი სულ მოულედ­ნელი სახით შეიძლება წარმოგვიდგნენ - კედელში ჩაშენებილი ბლოკიდან ჯიბის ვიდეოტელეფონამდე. ამ კომპიუტერების პროგრამული უზრუნველყოფაში საჭირო იქნება *მკვეთრი განცალკავება ინტერფეისების (რა გამოვიტანოთ) ალგორითმების­გან* (როგორ). ეს გამოყოფა ***მიღწეულ იქნება კროს-პლათმორმობით, კროს-ენებით, კროს-პროექტირებით***.

**მონოლითური აპლიკაციები** - ეს ისეთი აპლიკაციაა, რომელშიც მონაცემთა წვდო­მა, ბიზნეს-ლოგიკა და კომპიუტერის მონიტორზე მონაცემთა გამოტანა ”ჩაკერებუ­ლია” ერთ პროგრამაში. ასეთი აპლიკაციები უკვე მოძველებულია. ამ პრიმიტიული, ერთდონიანი, მაგრამ რესურსტევადი აპლიკაციებიათვის გამოიყენებოდა მეინფრე­იმი, რომელშიც ერთდროულად ფუნქციონირებდა ასეთ ამოცანათა მრავალი ასლე­ბი, რომლებიც სრულდებოდა მრავალამოცანიან გარემოში, სადაც გამოთვლითი პროცესი რთულად იყო ორგანიზებული და არსებობდა ამოცანათა მართვის სპეცი­ალური ენა.

**კლიენტ-სერვერის აპლიპაცია -** დღესდღეობით უმეტესი აპლიკაცია სწორედ ამ ტექნოლოგიის ბაზაზე მუშაობა. ეს ტექნოლოგია დროთა განმავლობაში დაიხვეწა: NetWare ფაილური სერვერებიდან მოხდა გადასვლა კროსპლატფორმულ მასშტაბი­რებად აპლიკაციების სერვერებზე, რომელზეც განლაგებულია Oracle, IBM, Sybase პროგრამული უზრუნველყოფა. ფაილური სერვერები მაინც დარჩნენ გამოყენებაში არქივებისა და შუალედური რეზულტატების შესანახად; მემკვიდრეობითი სისტე­მების შესაბამისობის უზრუნველყოფისათვის. ნახატი 1.



**ნახატი 1. კლიენტ-სერვერი**

**განაწილებული აპლიკაციები.** ობიექტზე- ორიენტირებული დაპროგრამების სა­შუალებით შესაძლებელია უფრო საიმედო, ნებისმიერ პირობებისადმი ადაპტირე­ბადი, იაფი, სწრაფად ცვლილებადი შესაძლებლობის კროსპლატფორმული პროგ­რამები, რომელნიც ადვილი შესაქმნელია და ექცპლუატაციაშიც მარტივია. ამჟამად პროგრამული უზრუნველყოფის ბაზარი ისეთია, რომ დეველოპერებმა (შემმუშავე­ლები) ზუსტად არ იციან როგორ იქნება ამოცანა განსაზღვრული და სპეციფირებუ­ლი დეტალებში, რადგანაც თვით შემკვეთებს არ შეუძლიათ მკაფიოდ ჩამოაყალი­ბონ მოთხოვნები, ამასაც შეიძლება არსებითი მნიშვნელობა არ ჰქონდეს, რადგანაც, როცა სისტემა იქნება დასრულებული,, ის შეიძლება უკვე მოძველდეს და საჭირო­ებდეს მოდიფიკაციას და ეს მოდიფიცირება გახდეს პერმანენტული დინა­მიურად ცვლადი ბაზრის კონიუქტურის პირობებში. იდეალში, პროგრამები უნდა გახდნენ ”ცოცხალი” და თვითსწავლებადი, მათ უნდა შეეძლოთ შეეგუონ გარედან წამოსულ შეშფოთებებს, და იმუშაონ არა მონაცემებზე, არამედ მეტამონაცემებზე, ანუ მათ აღ­წერასთან, რომელშიც შესაძლებელია მუდმივი ცვლილებები. ახალი დაგეგმარების საშუალებებში კასკადური მოდელი აღარ მუშაობს, სპირალური მოდელი უმეტეს შემთხვევაში უშედეგოა და აქცენტი აღებულია ტიპების დინამიურ განსაზღვრაზე (”პროტოტირება”). ობიექტზე - ორიენტირებული მიდგომა იძლევა საშუალებას ძველი და ახალი ტექნოლოგიების შერწყმას, ინკაფსულაციის და შეფუთვის (wrap-ping) მეშვეობით. მაგალითად, შეიძლება განვიხილოთ მოდულები, რომლებიც დაწერილია პროცედურულ ენებზე, როგორც ობიექტი და გამოვიყენოთ ისინი. ინკაფსულაცია იმითაა კარგი, რომ ის შეიძლება გამოვიყენოთ მემკვიდრეობითი სისტემების ინტეგრაციისა და რეინჟინირებისთვის, ძველი და გამოცდილი ფუნქციონალების შენარჩუნებით. კარგ მაგალითად შეიძლება მივიჩნიოთ IBM Intelligent Miner -ინტელექტუალური საძიებო მანქანის შემუშავება. აქ შემუშავებულია ინტერფეისის ბლოკი JavaGUI, ის გამოიყენებს C-API-ის ინტერფეისების ნაკრებს ბირთვთან მიმართვისთვის, რომელიც დაწერილია C++-ზე. მომხმარებელს შეუძლია მუშაობა გრაფიკულ Java - ინტერფეისის ახალ ნაკრებთან, რომელსაც შეუძლია მიმართოს ბირთვს C Wrapper-ის საშუალებით. მომხმარებელი ამასთან ვერ გრძნობს, რადგანაც საქმე აქვს მხოლოდ Java-სთან. სხვა მნიშვნელოვანი უპირატესობა ობიექტზე - ორიენტირებული მიდგომაში არის განმეორებით გამოყენებადი კოდი. განაწილებული ობიექტებს შეუძლიათ იმუშაონ როგორც დამოუკიდებლად, ასევე კონტეინერში. ყოველ მოდულს ან კომპონენტს შეუძლია ჰქონდეს ურთიერთობა სხვა კომპონენთან. დეველოპერები არ ქმნიან ახალ აპლიკაციას ნულიდან, ამის სანაცვლო ისინი ქმნიან უფრო მვირე და სრულიად გასაგებ ობიექტებს, ხოლო აპლიკაცია შედგება ამ ობიექტებისგან და აქტიურად იყენებს არსებულ კოდს. რადგანაც განაწილებული აპლიკაციის კოდის დიდი ნაწილი მუშაობს სერვერებზე, მისი მოდიფიკაციის შემთხვევაში, აღარაა საჭირო აპლიკაციის კლიენტის ნაწილის მასობრივი განახლება, რადგანაც ის ”გადმოქაჩავს” შეცვლილ კოდა და ამით თავის თავის მოდიფიცირებას ახდენს.

ობიექტზე - ორიენტირებული მიდგომა კარგია დინამიური WEB-აპლიკაციების შექმნისათვის, რომელთაც ყოველდღიურად რამდენიმე ათასი მომხმარებელი მი­მართავს სხვადასხვა მოთხოვნებით. ეს შეიძლება იყოს WEB-მაღაზია, WEB-ბანკი, საძიებო სისტემა და სხვა. მნიშვნელოვანია არა მარტო მისი ინფრასტრუქტუ­რის, არამედ მისი ინფორმაციული შიგთავსის მოდიფიკაციის პროცესების დაგეგმარება და აგება. ინფრასტრუქტუ­რის აგება - ეს იგივე პროექტირებაა პროგრამული უზ­რუნველყოფის, მხოლოდ ახალი ინსტრუმენტალური საშუალებებით. ამასთან შიგთავსის მოდიფიკაციის პროცესების აგება წააგავს მსხვილი ინფორმაციული ყოველდღიურის მომზადებას და გამოშვებას. ამ პროცესში უნდა ჩავრთოთ ისეთი ქმედებები, როგორიცაა მონაცემთა შეყვანა, მონაცემთა სისწორის შემოწმებას და ა. შ. ასეთი ამოცანების გადაწყვეტისთვის საჭიროა ამ ოპერატორთა დიდი რაოდენო­ბა, ან პროცესის მნიშვნელოვანი ავტომატიზაცია



**ნახატი 2 განაწილებული აპლიკაცია**

ორივე ეს ამოცანა მკვეთრად განსხვავდება გამოყენებადი მეთოდების რაოდენობით და თვისებებით. ამასთან გასათვალისწინებელია, რომ ინტერნეტ-პროექტირებისას ხშირად აღარ მუშაობს ისეთი ტრადიციული მიდგომები, როგორიცაა ეტაპობრივი პროექტირება, ვერსიების გამოშვების დროის დაგეგმარება, ტესტირება და სხვა. შეიძლება ამის გამოც, ზოგიერთი დეველოპერი და კონსულტანტი ხშირად საუბ­რობენ პროგრამული უზრუნველყოფისთვის ”ახალ ბიზნეს-მოდელზე”, რომელშიც WEB-სისტემა აღარაა პროდუქტი, რომელსაც ყიდულობ და რომელსაც ფლობ, არა­მედ ხდება საარენდო ხასიათის საათობრივი ანაზღაურებით.

**ჰოსტინგი და ჰაკინგი. სპეციალისტთა პრობლემა.**

ბიზნესი და კომერცია, რომელიც ახალ ინფორმაციულ ტექნოლოგიებზეა დაფუძ­ნებული თავის მოთხოვნებს აყენებს. ინფორმაციულ ტექნოლოგიებზე სამუშაოს მოცულობა იზრდება და მას უკვე ვეღარ გაწვდებიან კომპანიაში არსებული მაღალ­კვალიფიცირებული პროგრამისტები და დამპროექტებელი, ამიტომ კომპანია ნაკ­ლებ კვალიფიცირებულ სპეციალისტებსაც ასაქმებს. ამიტომ სისტემები WEB ბა­ზაზე უნდა იქნეს დაგეგმილი და დანერგილი უმოკლეს დროში. ამიტომ წარმოიშვა ტერმინი - ჰაკინგი - ზესწრაფი შემუშავება არადოკუმენტალური პროგრამებისა, რომლებიც შექმნილია წინასწარი გეგმის გარეშე. ცხადია, რომ ასეთი ტემპებისათ­ვის საჭიროა სულ სხვა ინსტრუმენტალური საშუალებები. და ისინიც შეიქმნენ. მაგალითად Lotus Domino designer -ის საშუალებით შეიძლება შეიქმნას მონაცემთა ბაზები, WEB საიტი Domino-სერვერისთვის, სულ რაღაც რამდენიმე დღეში, თანაც ეს საიტი ძალიან პროფესიონალური იქნება და მას რამდენიმე ასეული მომხმარებ­ლის მომსახურება შეუძლია. იგივე ითქმის Oracle Tools-ზე Oracle Web Application Server-სთვის, და Microsoft-ის პროდუქტების მთელ გამაზე და ასევე IBM WEB Studio-ზე WebSphere -სთვის და სხვა პროდუქტეზე. ზოგჯერ მსხვილი ინტერნეტ-პროვაიდერი აყალიბებს პროგრამისტებისაგან და დამგეგმარებისგან სპეციალურ გუნდს, რომლებიც ასრულებენ **ჰოსტინგს** ანუ, მსხვილი შემკვეთისთვის სისტემის შემუშავებას და შემდომში მის ექსპლუატაციას უშუალოდ თავის სერვერზე.

**განმარტებები**

**ბიზნეს ლოგიკა -** ინფორმაციული სისტემების დამუშავებაში - საგნობრივ სფეროში, არეში ( ადამიანის მოღვაწეობის სფერო რომელსის მხარდაჭერასაც ახორციელებს ინფორმაციული სისტემა) ობიექტების წესების, პრინციპების, მოქმედებათა ურთი­ერთკავშირების ერთობლიობა. ანუ ბიზნეს-ლოგიკა ეს არის ავტომატიზირებადი ოპერაციების წესებისა და შეზღუდვების რეალიზაცია. სინონიმია ”საგნობრივი სფეროს ლოგიკისა” – domain logic

ამდენად, ბიზნეს-ლოგიკა ეს არის საგნობრივი სფეროს რეალიზაცია ინფორმაცი­ულ სისტემებში. მას განეკუთვნება მაგალითად, სესხების დაფარვის ყოველთვიური გადასახადების გამოთვლის ფორმულები - *ფინანსურ ინდუსტრიაში*; პროექტის ხელმძღვანელისადმი შეტყობინებების ავტომატიზირებული გაგზავნა, როცა ყველა ქვეშემდგომი დაასრულებს დავალებათა ნაწილებს - *პროექტების მართვის სისტემა-ში*; სასტუმროზე უარის თქმა, როვა ავიარეისი გადაიდება - *ტურისტულ ბიზნესში* და სხვა.

**კასკადური (ჩანჩქერის) მოდელი (Waterfall model)** - პროგრამული უზრუნველყო­ფის შემუშავების პროცესის მოდელი, რომელშიც შემუშავების პროცესი წარმოად­გენს ნაკადს თანამიმდევრულად ცვლად ფაზებს - მოთხოვნათა ანალიზი, პროექ­ტირება, პეალიზაცია, ტესტირება, ინტეგრაცია და მხარდაჭერა. ნახატი 2.

# სპირალური მოდელი (Spiral model) - პროგრამული უზრუნველყოფის შემუ­შავების პროცესის მოდელი, რომელშიც შემუშავების პროცესში ერთდროულად ხდება როგორც პროექტირება ასევე სტადიური პროტოტირება (სწრაფი რეალი­ზაცია). ნახაზი 3.



**ნახატი 3 კასკადური მოდელი**



**ნახატი 4. სპირალური მოდელი**

**ინკაფსულაცია (encapsulation) -** მექანიზმი, რომელიც აერთიანებს მონაცემებს და კოდს, რომელიც ახდენს ამ მონაცემებზე მანიპულაციას, ეს მექანიზმი ასევე იცავს მონაცემებს და კოდს გარე ზემოქმედებისგან და არასწორი გამოყენებისგან.

 состение программ без предварительной разработки и оперативное внесение исправлений в программы, не имеющие документации

составление программ без предварительной разработки и оперативное внесение исправлений в программы, не имеющие составление программ без предварительной разработки и оперативное внесение исправлений в программы, не имеющие документации - Прим.пер.).

составление программ без предварительной разработки и оперативное внесение исправлений в программы, не имеющие документации -

|  |  |  |
| --- | --- | --- |
|  |  |  |

**ინტერნეტ დაპროგრამირება -** დაპროგრამების ნაწილია, რომელიც ორიენტიტებუ­ლია ვებ აპლიკაციების შესამუშავებლად. ვებ-აპლიკაციები - პროგრამები WWW-ში დინამიური საიტების ფუნქციონირების უზრუნველსაყოფად.

ინტერნეტ დაპროგრამირების ენები - ეს ის ენებია, რომლებიც გათვალისწინებუ­ლია ვებ-ტექნოლოგიებთან სამუშაოდ. ინტერნეტ დაპროგრამირების ენები შეიძ­ლება დაცყოთ ორ ჯგუფად, რომლებიც იკვეთებიან: ***კლიენტური*** და ***სერვერული***.

**კლიენტური ენები,** პროგრამები კლიენტურ ენაზე მუშავდება მომხარებლის მხარეს, როგორც წესი მათ ასრულებს ბრაუზერი. წორედ ეს ქმნის მთავარ პრობლემას კლი­ენტური ენებისა - პროგრამის (სპრიპტის) შესრულების რეზულტატი დამოკიდებუ­ლია მომხმარებლის ბრაუზერზე. ანუ, თუ მომხმარებელი აკრძალავს კლიენტურ პროგრამის შესრულებას, მიუხედავად პროგრამისტის სურვილისა, იგი არ შეს­რულდება. ასევე შესაძლებელია, რომ ერთი და იგივე სკრიპტი სხვადასხვა ბრა­უზერებისთვის, ანდა თუნდაც ერთი და იგივე ბრაუზერის სხვადასხვა ვერსი­ისთვის იმუშავებს სხვადასხვანაირად. მეორეს მხრივ თუ პროგრამისტი იმედს ამყარებს სერვრტულ პროგრამებზე, მაშინ მას შეუძლია მათი მუშაობის გამარ­ტივება და შეამციროს დატვირთვა სერვერზე იმ პროგრამების ხარჯზე, რომლებიც კლიენტის მხარეს მუშაობენ, რადგანა ისინი ყოველთვის არ ითხოვენ გვერდის გადატვირთვას (გენერაციას).

ყველაზე გავრცელებული კლიენტური დაპროგრამების ენებია:

* [**HTML**](http://ru.wikipedia.org/wiki/HTML)
* [**CSS**](http://ru.wikipedia.org/wiki/CSS)
* [**JavaScript**](http://ru.wikipedia.org/wiki/JavaScript)
* [**VBScript**](http://ru.wikipedia.org/wiki/VBScript)
* [**ActionScript**](http://ru.wikipedia.org/wiki/ActionScript)
* [**Java**](http://ru.wikipedia.org/wiki/Java)

**სერვერული ენები,** როცა მომხმარებელი აფორმებს მოთხოვნას ამა თუ იმ ვებ-გვერზე ( გადადის მასზე ბმულით ან თავისი ბრაუზერის სამისამართო სტრიქონში ჩაწერს მისამართს), მაშინ გამოძახებული გვერდი თავიდან დამუშავდება სერვერზე, ანუ შესრულდება ყველა ის პროგრამა, რომელიც დაკავშირებულია გვერდზე და ამის შემდეგ გადაეცემა მომხმარებელს ქსელის საშუალებით ფაილის სახით. ამ ფაილს შეიძლება ჰქონდეს გაფართოება: [**HTML**](http://ru.wikipedia.org/wiki/HTML)**,** [**PHP**](http://ru.wikipedia.org/wiki/PHP)**,** [**ASP**](http://ru.wikipedia.org/wiki/Active_Server_Pages)**,** [**ASPX**](http://ru.wikipedia.org/wiki/Asp.net)**,** [**Perl**](http://ru.wikipedia.org/wiki/Perl)**,** [**SSI**](http://ru.wikipedia.org/wiki/SSI_%28%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%29)**,** [**XML**](http://ru.wikipedia.org/wiki/XML)**,** [**DHTML**](http://ru.wikipedia.org/wiki/DHTML)**,** [**XHTML**](http://ru.wikipedia.org/wiki/XHTML)**.**

პროგრამის მუშაობა უკვე სრულადაა დამოკიდებული სერვერზე, რომელზეც განლაგე­ბულია საიტი და იმ ენის ესათუ ის ვერსია, რომელის მხარდაჭერაა აქვს სერვერს. დაპ­როგრამირების სერვერული ენებია: [**PHP**](http://ru.wikipedia.org/wiki/PHP)**,** [**Perl**](http://ru.wikipedia.org/wiki/Perl)**,** [**Python**](http://ru.wikipedia.org/wiki/Python)**,** [**Ruby**](http://ru.wikipedia.org/wiki/Ruby), და ნებისმიერი [.**NET**  დაპ­როგრამირების ენა](http://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_.NET-%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2) (ტექნოლოგია [**ASP.NET**](http://ru.wikipedia.org/wiki/ASP.NET)), [**Java**](http://ru.wikipedia.org/wiki/Java)**,** [**Groovy**](http://ru.wikipedia.org/wiki/Groovy).

სერვერული ენების მნიშვნელოვანი შესაძლებლობაა უშუალო ურთიერთობის ორგანიზება მონაცემთა ბაზების დამუშავების სისტემებთან (**მბდს**) - იმ სერვერთან, რომელზეც ორგა­ნიზებულად ინახება ინფორმაცია და რომელიც ნებისმიერ დროს შეიძლება იქნეს გამოთ­ხოვილი. მონაცემთა ბაზების დამუშავების სისტემებია **მბდს** ( შედარებით პოპულარუ­ლები)

* [Firebird](http://ru.wikipedia.org/wiki/Firebird_%28database_server%29)
* [IBM DB2](http://ru.wikipedia.org/wiki/IBM_DB2)
* [IBM DB2 Express-C](http://ru.wikipedia.org/w/index.php?title=IBM_DB2_Express-C&action=edit&redlink=1)
* [Microsoft SQL Server](http://ru.wikipedia.org/wiki/Microsoft_SQL_Server)
* [Microsoft SQL Server Express](http://ru.wikipedia.org/wiki/Microsoft_SQL_Server_Express)
* [mSQL](http://ru.wikipedia.org/wiki/MSQL)
* [MySQL](http://ru.wikipedia.org/wiki/MySQL)
* [Oracle](http://ru.wikipedia.org/wiki/Oracle_%28%D0%A1%D0%A3%D0%91%D0%94%29)
* [PostgreSQL](http://ru.wikipedia.org/wiki/PostgreSQL)
* [SQLite](http://ru.wikipedia.org/wiki/SQLite)
* [Sybase Adaptive Server Enterprise](http://ru.wikipedia.org/wiki/Sybase)
* [MongoDB](http://ru.wikipedia.org/wiki/MongoDB)