საინფორმაციო სისტემის დაპროექტება ობიექტ-როლური მოდელირების და სერვის-ორიენტირებული არქიტექტურის ბაზაზე

გია სურგულაძე, ნინო თოფურია, კობა ბაკურია, მაკა ლომიძე
საქართველოს ტექნიკური უნივერსიტეტი

რეზიუმე

გადმოცემულია უნივერსიტეტის საინფორმაციო სისტემის განაწილებული ბაზის და მომხმარებელთა ინტერფეისების ავტომატიზებული დაპროექტების და პროგრამული რეალიზაციის პროცესის აღწერა ჰიბრიდული ტექნოლოგიების და სერვის-ორიენტირებული არქიტექტურის საფუძველზე. შემოთავაზებულია „Facts /ORM / ERM / DDL" კომპონენტების თანამიმდევრული გარდაქმნით MsSQL Server ბაზის აგების პროცესის შედეგები. აგრეთე მომხმარებელთა ინტერფეისების Web-სერვისების დამუშავება Workflow და Windows Communication Foundation ტექნოლოგიებით.

საკვანძო სიტყვები: უნივესიტეტი. საინფორმაციო სისტემა. მონაცემთა განაწილებული ბაზა. ORM-დიაგრამა. ER–მოდელი. Web-სერვისი. SOA. დაპროგრამების ჰიბრიდული ტექნოლოგია.

1. შესავალი

საუნივერსიტეტო მართვის საინფორმაციო სისტემის შექმნა ან არსებული სისტემის მოდიფიკაცია ინტეგრაციის პრინციპების საფუძველზე მოითხოვს ამ საპრობლემო სფეროს ობიექტ-ორიენტირებული, პროცეს-ორიენტირებული და სერვის-ორიენტირებული მიდგომების კომპლექსურ გამოყენებას [1,2]. სისტემის შესაბამისი ინფრასტრუქტურის დასამუშავებლად კი აუცილებელია დღეისათვის არსებული ისეთი სტანდარტებისა და მეთოდოლოგიების გამოყენება, როგორიცაა BSI, ITIL, COBIT [3-5], რაც საბოლოო ჯამში უზრუნველყოფს უსაფრთხო განაწილებული ინფორმაციული სისტემის შექმნას, მის შემდგომ მასშტაბირებას და განვითარებას. მეორეს მხრივ, ინტეგრაციის პროცესში სისტემის ცალკეული კომპონენტების დასამუშავებლად დროითი პარამეტრების და პროგრამული პროდუქტის ხარისხის გასაუმჯობესებლად აუცილებელი ხდება თანამედროვე CASE ტექნოლოგიების გამოყენება, როგორიცაა მაგალითად, Enterprise Architect (UML-ის ინსტრუმენტული საშუალება), Natural Object Role Modeling Architect (NORMA) და სხვ., რომლებიც მაიკროსოფთის Visual Studio .NET Framework პაკეტის სამუშაო გარემოს თავსებადია [6-8].

წინამდებარე სტატიაში გადმოცემულია საუნივერსიტეტო მართვის საინფორმაციო სისტემის საპილოტო ვერსიის მაგალითზე მონაცემთა განაწილებული ბაზის და მომხმარებელთა ინტერფეისების დაპროექტების და პროგრამული რეალიზაციის პროცედურების დეტალური აღწერა ზემოაღნიშნული ახალი ტექნოლოგიებისა და სერვის-ორიენტირებული არქიტექტურის საფუძველზე. პროგრამული უზრუნველყოფის სასიცოცხლო ციკლის ეტაპების შესაბამისად, საუნივერსიტეტო მართვის საინფორმაციო სისტემის შექმნა, UML-ტექნოლოგიით, ითვალისწინებს სისტემის ფუნქციონალური და არაფუნქციონალური მოთხოვნილებების განსაზღვრას, ობიექტ-ორიენტირებული ანალიზის და დაპროექტების განხორციელებას, შესაბამისად სისტემასთან მომხმარებელთა მუშაობის ინტერაქტიული სცენარების, კლასთა-ასოსიაციების და მდგომარეობათა დიაგრამების აგებით სხვადასხვა შემთხვევით მოვლენათა შესაბამისად.

ესაა ცოდნა სამართავი ობიექტის შესახებ, მისი სტატიკური (მდგომარეობათა სიმრავლე) და დინამიკური (ქცევათა სიმრავლე) მოდელებით. თუ ობიექტ-ორიენტირებული მოდელირების ტერმინებით ვისარგებლებთ, დასმული ამოცანის გადაწყვეტის „გასაღებს" კლასების, ობიექტების, კლასთაშორისი კავშირების, ობიექტ-როლური და არსთა-დამოკიდებულების მოდელებისა და სხვა

სახის დიაგრამების აგება წარმოადგენს. ხოლო შემდეგ, კლასთა-ასოციაციებისა და არსთა-დამოკიდებულების დიაგრამათა საფუძველზე განხორციელდება მიზნობრივი სისტემის პროგრამული კოდების რეალიზაციის ავტომატიზებული პროცესი [9,10].

2. ობიექტ-როლური მოდელის დაპროექტება

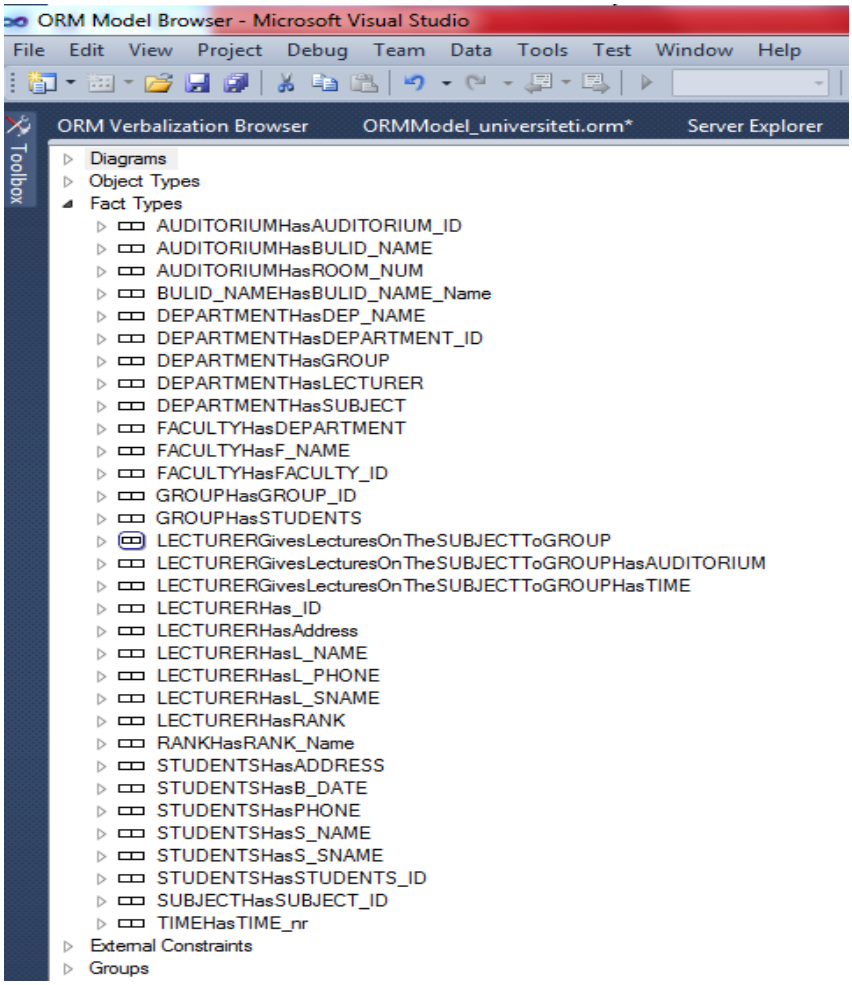
განვიხილოთ ზოგადად „უნივერსიტეტის“ კლასის ობიექტისათვის მონაცემთა განაწილებული ბაზის დაპროექტების ამოცანა. უნივერსიტეტის საპრობლემო სფეროს არაფორმალიზებული აღწერის ობიექტებია (ტერმინთა ლექსიკონი): ფაკულტეტი, დეპარტამენტი, სტუდენტი, ლექტორი, საგანი (აკადემიური დისციპლინები, რომლებიც იკითხება შესაბამისი დეპარტამენტებისა და სპეციალობების მიხედვით), სასწავლო გეგმები, სილაბუსები (პროგრამები), ლექციები, პრაქტიკული და ლაბორატორიული სამუშაოები, აუდიტორიები, გამოცდები, ტესტირება და ა.შ.

ობიექტ-როლური მოდელირების თეორიის წესების თანახმად საჭიროა აიგოს უნივერსიტეტის სასწავლო პროცესის შესაბამისი ORM-დიაგრამა. ამისათვის **ფაქტ-შეზღუდვების** ერთობლიობით (რომელსაც ადგენს სისტემური ანალიტიკოსი და საბოლოო მომხმარებელი), რომლებშიც ასახულია საპრობლემო სფეროს შესახებ ცოდნა (კლასებისა და ობიექტების ძირითადი ტერმინები და ქცევის წესები, დებულებით დადგენილი კანონზომიერებები და სხვ.), გადაიტანება Ms_Visual Studio.NET Framework სამუშაო გარემოში, ობიექტ-როლური მოდელის, NORMA-პაკეტის (Natural ORM Architect) ინტერფეისზე [7]. მაგალითად, ასეთი ფაქტები შეიძლება იყოს:

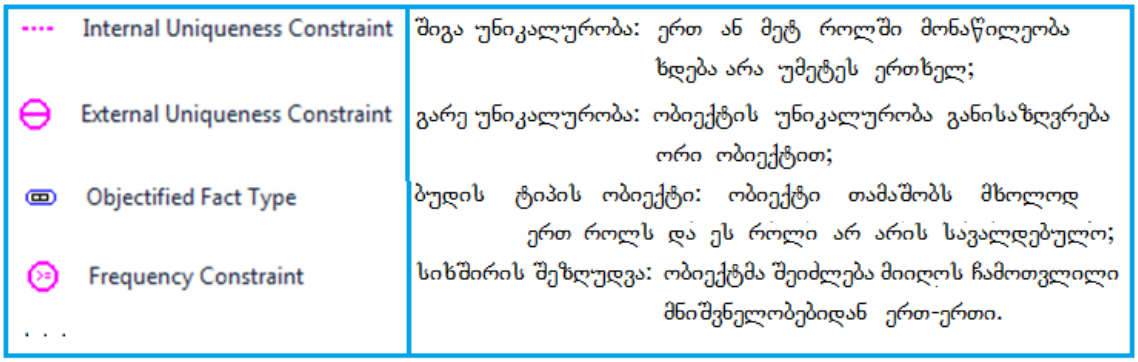
f1 : ლექტორს აქვს გვარი
f2 : ლექტორი მუშაობს დეპარტამენტში
f3 : ლექტორს აქვს თანამდებობა
f4 : ლექტორს აქვს ტელეფონი
f5 : ლექტორს აქვს ელ-ფოსტა
f6 : ლექტორს აქვს ხარისხი
f7 : ლექტორი კითხულობს საგანს
f8 : ლექტორი ასწავლის №-ჯგუფს
f8 : სტუდენტი არის №-ჯგუფში
. . .
f50 : ლექტორი მუშაობს №-დეპარტამენტში
f51 : დეპარტამენტი ეკუთვნის №-ფაკულტეტს
f52 : №-ჯგუფი ეკუთვნის №-დეპარტამენტს
f53 : ფაკულტეტს აქვს დასახელება
. . .
f100 : სტუდენტი არ შეიძლება იყოს ერთზე მეტ ჯგუფში
f101 : ლექტორი არ შეიძლება იყოს სრულ შტატზე ერთზე მეტ დეპარტამენტში
f102 : ლექტორი დროის ერთ მომენტში არ შეიძლება იყოს ორ სხვადასხვა აუდიტორიაში
და ა.შ.

1-ელ ნახაზზე მოცემულია Visual Studio.NET სამუშაო გარემოში ORM ინსტრუმენტის გამოყენებით მიღებული შედეგები „უნივერსიტეტის“ ფაქტების შეტანის საფუძველზე.

მე-2 ნახაზზე წარმოდგენილია ობიექტ-როლური მოდელირების თეორიაში არსებული ზოგიერთი შეზღუდვის მაგალითი, რომლებიც გამოყენებულია ჩვენ მაგალითში. შეზღუდვების აღწერა ხდება კატეგორიალური მიდგომის საფუძველზე, რომელიც აერთიანებს სალაპარაკო ენის ფორმალური გრამატიკის და ლოგიკურ-ალგებრულ წესებს. შედეგად მიიღება პრედიკატები, რომლებიც შეიძლება იყოს ერთ-, ორ- ან n-ადგილიანი [10].

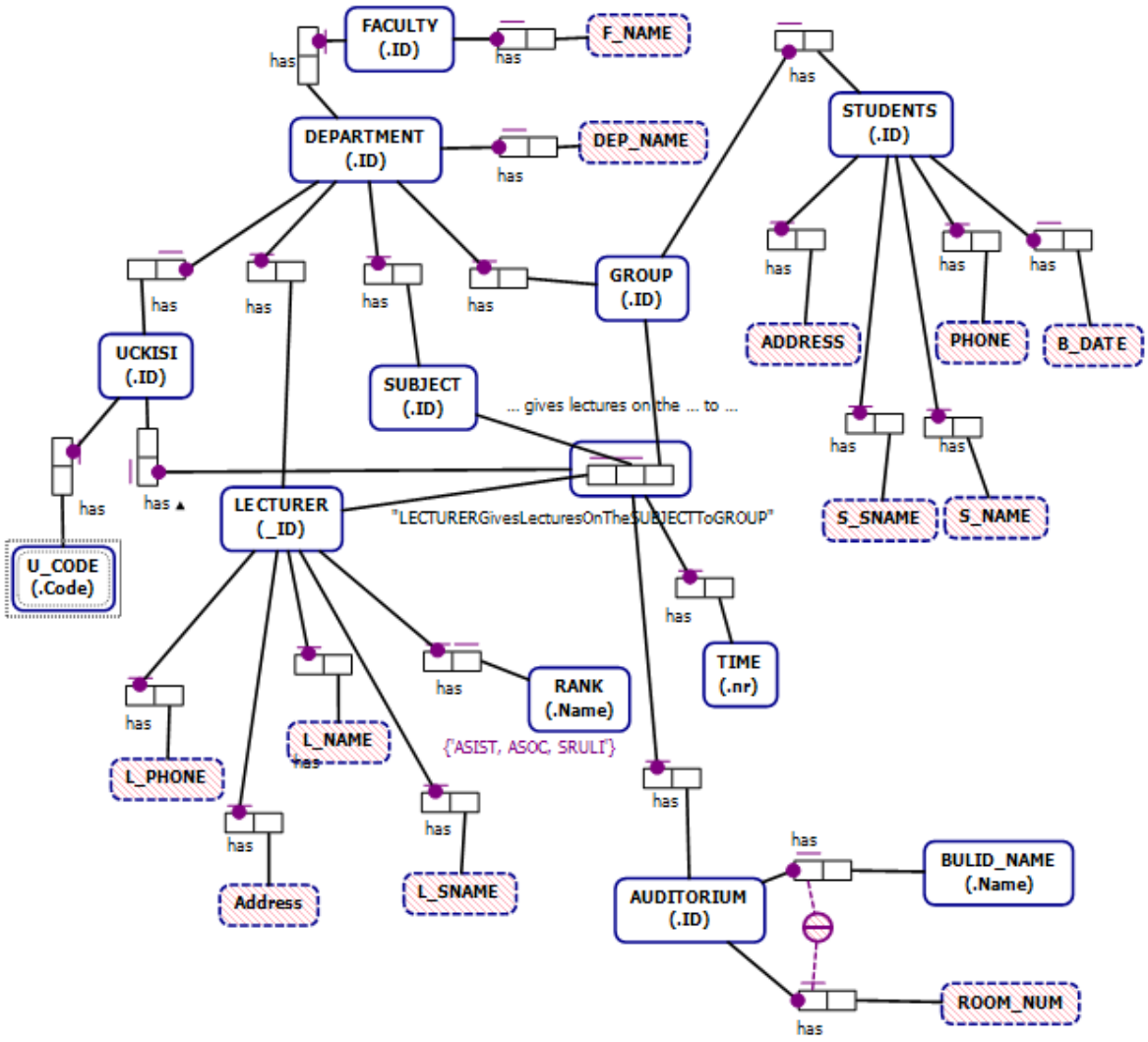


ნახ.1. „უნივერსიტეტის“ ფაქტების აღწერის ფრაგმენტი



ნახ.2. შეზღუდვების აღწერის მაგალითები ORM-დიაგრამაზე

ზემოჩამოთვლილი ფაქტებიდან NORMA ინსტრუმენტი გვაძლევს შემდეგი სახის ORM-დიაგრამას (ნახ.3). აქ შესაძლებელია ახალი ფაქტის დამატება, არსებულის მოდიფიკაცია / წაშლა.

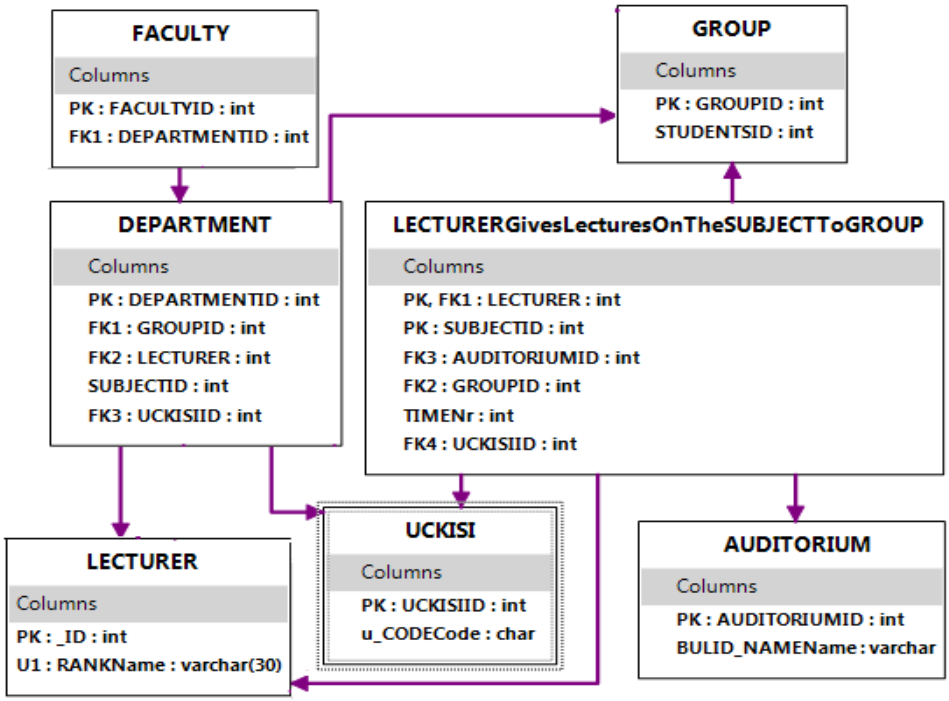


ნახ.3. „უნივერსიტეტის“ ORM დიაგრამის ფრაგმენტი (საპრობლემო სფეროს კონცეპტუალური მოდელი-1)

3. არსთა-დამოკიდებულების ER მოდელის დაპროექტება

მომდევნო ეტაპზე განხორციელდება ობიექტ-როლური მოდელის ავტომატიზებული გადაყვანა არსთა-დამოკიდებულების მოდელში. სისტემის დამპროექტებელი გააქტიურებს NORMA პაკეტის მენიუდან გენერაციის პროცედურას. ORM-დიაგრამიდან გენერირებული ERM-დიაგრამა მოცემულია მე-4 ნახაზზე. აქ შესაბამისად გამოკვეთილია შვიდი ობიექტი (არსი - Entity): ფაკულტეტი (Faculty), დეპარტამენტი (Department), ჯგუფი (Group), სტუდენტი (Students), ლექტორი (Lecturer), საგანი (Subject), აუდიტორია (Auditorium), საგამოცდო_უწყისი (Uckisi).

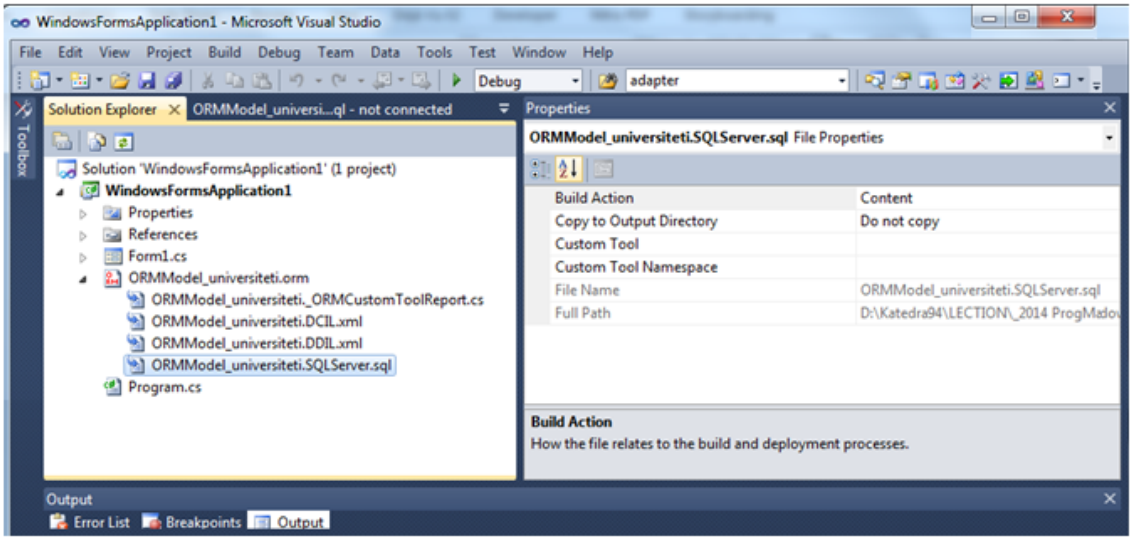
დიალოგში შესაძლებელია ER-მოდელის ობიექტების (Tables) განლაგების შეცვლა, რათა ვიზუალურად უფრო მოხერხებული მდებარეობა მიიღოს თითოეულმა, ობიექტთაშორისი კავშირების რაც შეიძლება ნაკლები გადაკვეთებით. ცხრილებში ჩანს ობიექტის სახელი და ატრიბუტთა დასახელებები, ტიპების მითითებით. აგრეთვე ასახულია პირველადი გასაღებური (PK), მეორეული გასაღები (FK) და ა.შ.



ნახ.4. ORM დიაგრამიდან გენერირებული ER-მოდელი (საპრობლემო სფეროს კონცეპტუალური მოდელი-2)

4. მონაცემთა ბაზის სერვერზე განთავსება

მონაცემთა ბაზის კონცეპტუალური ERM სქემის აგების შემდეგ საჭიროა მის საფუძველზე სისტემის მიერ დაიწეროს შუალედური ტექსტური ტიპის DLL–ფაილი, რომელიც მომავალში SQL Server მონაცემთა ბაზების მართვის სისტემამ უნდა გამოიყენოს. ამგვარად, ER-დიაგრამიდან, ცხრილებითა და ატრიბუტებით, ავტომატურად გენერირდება .DDL ფაილები, რომლებიც შემდგომ სტრუქტურულად მოთავსდება Ms SQL Server მონაცემთა განაწილებულ ბაზაში. მე-5 ნახაზზე ნაჩვენებია ამ ეტაპის პროცესის ინიცირების დიალოგური სქემა.



ნახ.5. DDL ფაილის გენერაცია ER -მოდელიდან

1-ელ ლისტინგში მოცემულია ავტომატურად გენერირებული DDL-ფაილის ტექსტის ფრაგმენტი.

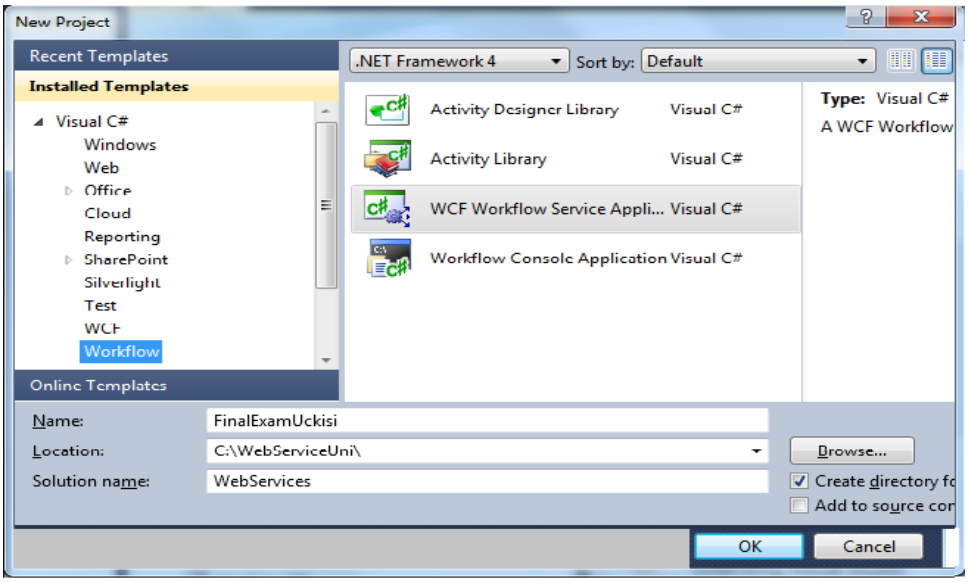
```
-- Listing_1 -----DDL file -------------------------
CREATE SCHEMA ORMModel1
GO
GO
CREATE TABLE ORMModel1.FACULTY
(  FACULTYID INTEGER IDENTITY (1, 1) NOT NULL,
   DEPARTMENTID INTEGER NOT NULL,
   CONSTRAINT FACULTY_PK PRIMARY KEY(FACULTYID)  )
GO
CREATE TABLE ORMModel1.DEPARTMENT
(  DEPARTMENTID INTEGER IDENTITY (1, 1) NOT NULL,
   GROUPID INTEGER NOT NULL,
   LECTURER INTEGER NOT NULL,
   SUBJECTID INTEGER IDENTITY (1, 1) NOT NULL,
   CONSTRAINT DEPARTMENT_PK PRIMARY KEY(DEPARTMENTID)   )
GO
CREATE TABLE ORMModel1."GROUP"
(  GROUPID INTEGER IDENTITY (1, 1) NOT NULL,
   STUDENTSID INTEGER IDENTITY (1, 1) NOT NULL,
   CONSTRAINT GROUP_PK PRIMARY KEY(GROUPID)       )
GO
CREATE TABLE ORMModel1.LECTURER
(  "_ID" INTEGER IDENTITY (1, 1) NOT NULL,
   RANKName NATIONAL CHARACTER VARYING(30) NOT NULL,
   CONSTRAINT LECTURER_PK PRIMARY KEY("_ID"),
   CONSTRAINT LECTURER_UC UNIQUE(RANKName),
   CONSTRAINT LECTURER_RANKName_RoleValueConstraint1 CHECK (RANKName IN (N'ASIST, ASOC, SRULI'))  )
GO
CREATE TABLE ORMModel1.LECTURERGivesLecturesOnTheSUBJECTToGROUP
(  LECTURER INTEGER NOT NULL,
   SUBJECTID INTEGER IDENTITY (1, 1) NOT NULL,
   AUDITORIUMID INTEGER NOT NULL,
   GROUPID INTEGER NOT NULL,
   TIMENr INTEGER NOT NULL,
   CONSTRAINT LECTURERGivesLecturesOnTheSUBJECTToGROUP_PK PRIMARY KEY(LECTURER, SUBJECTID)     )
GO
CREATE TABLE ORMModel1.AUDITORIUM
(  AUDITORIUMID INTEGER IDENTITY (1, 1) NOT NULL,
   BULID_NAMEName NATIONAL CHARACTER VARYING(MAX) NOT NULL,
   CONSTRAINT AUDITORIUM_PK PRIMARY KEY(AUDITORIUMID)          )
GO
ALTER TABLE ORMModel1.FACULTY ADD CONSTRAINT FACULTY_FK FOREIGN KEY (DEPARTMENTID) REFERENCES
ORMModel1.DEPARTMENT (DEPARTMENTID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
ALTER TABLE ORMModel1.DEPARTMENT ADD CONSTRAINT DEPARTMENT_FK1 FOREIGN KEY (GROUPID) REFERENCES
ORMModel1."GROUP" (GROUPID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
ALTER TABLE ORMModel1.DEPARTMENT ADD CONSTRAINT DEPARTMENT_FK2 FOREIGN KEY (LECTURER) REFERENCES
ORMModel1.LECTURER ("_ID") ON DELETE NO ACTION ON UPDATE NO ACTION
GO
ALTER TABLE ORMModel1.LECTURERGivesLecturesOnTheSUBJECTToGROUP ADD CONSTRAINT
LECTURERGivesLecturesOnTheSUBJECTToGROUP_FK1 FOREIGN KEY (LECTURER) REFERENCES ORMModel1.LECTURER
("_ID") ON DELETE NO ACTION ON UPDATE NO ACTION
GO
ALTER TABLE ORMModel1.LECTURERGivesLecturesOnTheSUBJECTToGROUP ADD CONSTRAINT
LECTURERGivesLecturesOnTheSUBJECTToGROUP_FK2 FOREIGN KEY (GROUPID) REFERENCES ORMModel1."GROUP"
(GROUPID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
ALTER TABLE ORMModel1.LECTURERGivesLecturesOnTheSUBJECTToGROUP ADD CONSTRAINT
LECTURERGivesLecturesOnTheSUBJECTToGROUP_FK3 FOREIGN KEY (AUDITORIUMID) REFERENCES ORMModel1.AUDITORIUM
(AUDITORIUMID) ON DELETE NO ACTION ON UPDATE NO ACTION
GO
GO
```

შეიძლება ჩვთვალოთ, რომ ამ DDL ფაილის კოპირებით Ms SQL Server –ში შეიქმნება შესაბამისი ბაზა, ცხრილებით, ატრიბუტებით და კავშირებით. რა თქმა უნდა, შესაძლებელია აქაც დამპროექტებლის ჩარევა ბაზის სტრუქტურის ზოგიერთი კომპონენტის შესასწორებლად, საჭიროების შემთხვევაში.

5. ბიზნეს-პროცესის სერვისის შექმნა

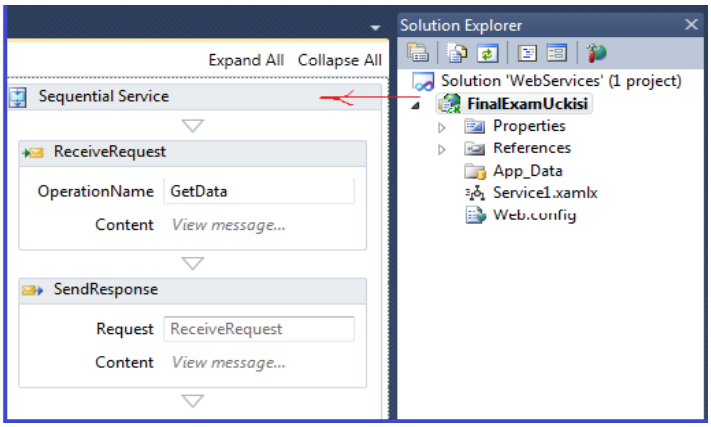
როგორც ცნობილია, ბიზნეს-პროცესი შეიძლება განთავსდეს ვებ-სერვისში, რომელიც უზრუნველყოფს ბიზნეს-პროცესის გადაწყვეტილების (შედეგის) მიწოდებას კლიენტებისთვის (ვებ-აპლიკაციებისთვის). ვებ-სერვისი იღებს მოთხოვნას კლიენტისგან, ასრულებს მის დამუშავებას და უბრუნებს პასუხს. ეს პროცედურები სრულდება Receive და Send ქმედებებით. განვიხილოთ ჩვენი მაგალითის რეალიზაცია ჰიბრიდული ტექნოლოგიების, WF (Workflow Foundation) და WCF ბაზაზე (Windows Communication Foundation) [11]. ისევე, როგორც WPF (Windows Presentation Foundation) ტექნოლოგია, Visual Studio .NET Framework 4.0/4.5 გარემოში ქმნის მომხმარებელზე ორიენტირებულ მაღალი ხარისხის დიზაინის აპლიკაციებს C#.NET (ლოგიკის ნაწილი) და XAML (დიზაინის ნაწილი) ენების საფუძველზე [12,13].

ავამუშავოთ Visual Studio 2010 და შევქმნათ ახალი პროექტი WCF Workflow Service Application template გამოყენებთ. შევიტანოთ პროექტის სახელი FinalExamUckisi და solution-ის სახელი WebServices.



ნახ.6. Visual Studio.NET –ში WCF-პროექტის შექმნა

შეიქმნება საინიციალიზაციო workflow Sequence ბლოკი, რომელიც შეიცავს Receive და SendReply ქმედებებს, როგორც მე-7 ნახაზზეა ნაჩვენები.



ნახ.7. FinalExamUckisi პროექტის workflow Sequence ბლოკი

თავიდან საჭიროა ამ ქმედებების კონფიგურაცია სერვისის კონტრაქტის განსაზღვრის მიზნით, რომელსაც ისინი დააკმაყოფილებს. შემდეგ დავამატოთ დამუშავების ბიზნეს-პროცესი, რომელიც განხორციელდება Receive და SendReply ქმედებებს შორის. შაბლონი შექმნის საწყის ბიზნესპროცესს ფაილში, სახელით Service1.xamlx. შევცვალოთ Solution Explorer-ში ეს სახელი FinalExamUckisi.xamlx -ით.

სერვისი, რომელიც მაგალითის სახით უნდა შევქმნათ, ,,საფინალო გამოცდისთვის ძებნის შესაბამის უწყისებს მითითებული აკადემიურ ჯგუფის, ლექტორის და აკადემიური საგნის მიხედვით. ელექტრონული საგამოცდო უწყისები ეკუთვნის ფაკულტეტის დეკანატს, ხოლო მათი დამუშავება ხდება დეპარტამენტთა ლექტორების მიერ''.

6. სერვისის კონტრაქტის განსაზღვრა

WCF სისტემებში იყენებენ კონტრაქტების სამ დონეს: მონაცემთა კონტრაქტი, შეტყობინებათა კონტრაქტი და სერვისის კონტრაქტი [14].

მონაცემთა კონტრაქტის დანიშნულებაა შეთანხმება კლიენტსა და სერვისს შორის ერთმანეთთან გასაცვლელ მონაცემებზე (ითვალისწინებენ მონაცემთა სტრუქტურებს, პარამეტრებს, მოწესრიგებას და ა.შ.).

შეტყობინებათა კონტრაქტი უზრუნველყოფს SOAP (Simple Object Access Protocol) შეტყობინებების კონტროლს, რომელიც გამოიყენება ქსელში სხვადასხვა შეტყობინების გასაცვლელად XML ფორმატში. იგი უზრუნველყოფს აგრეთვე ინფორმაციის გაცვლის უსაფრთხოებას შეტყობინებების დონეზე.

სერვისის კონტრაქტი (ანუ კონტრაქტი მომსახურებისთვის) განსაზღვრავს ოპერაციათა სახეებს, რომლებსაც უზრუნველყოფს სერვისი. იგი კლიენტს აწვდის ასევე ინფორმაციას: შეტყობინებაში მონაცემთა ტიპების შესახებ, ოპერაციათა ადგილმდებარეობის შესახებ, ინფორმირების პროტოკოლისა და სერიალიზაციის ფორმატის შესახებ, შეტყობინებათა გაცვლის შაბლონების შესახებ (ცალმხრივი, ორმხრივი ან კითხვა/პასუხის ტიპებით).

ჩვენი პროექტისთვის სერვისის კონტრაქტის ასაგებად უნდა შევქმნათ საგამოცდო უწყისის ინფორმაციის კლასი C# ენაზე - UckisiInfo.cs. ამისთვის Solution Explorer-ში მარჯვენა ღილაკით FinalExamUckisi პროექტზე ვირჩევთ Add->Class სახელით UckisiInfo.cs, რომლის ტექსტი მოცემულია 1-პირველ ლისტინგში.

```
// ----- ლისტინგი 1 – Service Contract ---
using System;
using System.Collections.Generic;
using System.Runtime.Serialization;
using System.ServiceModel;

namespace FinalExamUckisi
{  // ---- სერვისის კონტრაქტის განსაზღვრა  IfinalExamUckisi,  რომელიც
   // --- შედგება ერთი მეთოდისგან  -  LookupUckisi () ------
     [ServiceContract]
     public interface IFinalExamUckisi
     {   [OperationContract]
         UckisiInfoList LookupUckisi(UckisiSearch request);     }
     //--- მოთხოვნის ეტყობინების განსაზღვრა , UckisiSearch ----
     [MessageContract(IsWrapped = false)]
     public class UckisiSearch
      {  private String _JGUPI;
         private String _Sagani;
         private String _Lectori;
         public UckisiSearch() {  }
```

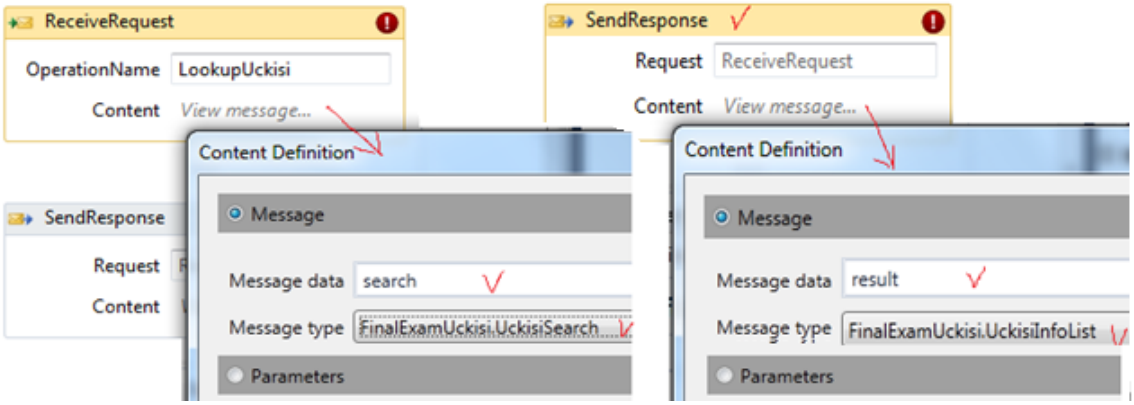
```
        public UckisiSearch(String sagani, String lectori, String jgupi)
        {
            _Sagani  = sagani;
            _Lectori = lectori;
            _JGUPI   = jgupi;       }
        #region Public Properties
        [MessageBodyMember]
        public String Sagani
        {   get { return _Sagani; }
            set { _Sagani = value; }    }
        [MessageBodyMember]
        public String Lectori
        {   get { return _Lectori; }
            set { _Lectori = value; }   }
        [MessageBodyMember]
        public String JGUPI
        {   get { return _JGUPI; }
            set { _JGUPI = value; }     }
        #endregion Public Properties
    }

    // --- UckisiInfo კლასის განსაზღვრება -----
        [MessageContract(IsWrapped = false)]
    public class UckisiInfo
    {   private Guid _ExamUckisiID;
        private String _JGUPI;
        private String _Sagani;
        private String _Lectori;
        private String _Status;
        public UckisiInfo() {   }
        public UckisiInfo(String sagani, String lectori, String jgupi, String status)
        {   _Sagani = sagani;
            _Lectori = lectori;
            _JGUPI = jgupi;
            _Status = status;
            _ExamUckisiID = Guid.NewGuid();     }
        #region Public Properties
        [MessageBodyMember]
        public Guid ExamUckisiID
        {   get { return _ExamUckisiID; }
            set { _ExamUckisiID = value; }      }
        [MessageBodyMember]
        public String Sagani
        {   get { return _Sagani; }
            set { _Sagani = value; }            }
        [MessageBodyMember]
        public String Lectori
        {   get { return _Lectori; }
            set { _Lectori = value; }           }
        [MessageBodyMember]
        public String JGUPI
        {   get { return _JGUPI; }
            set { _JGUPI = value; }             }
        [MessageBodyMember]
        public String status
        {   get { return _Status; }
            set { _Status = value; }            }
        #endregion Public Properties
    }
    // ------ საპასუხო შეტყობინების განსაზღვრა --- UckisiInfoList ---
       [MessageContract(IsWrapped = false)]
    public class UckisiInfoList
```

```
    {   private List<UckisiInfo> _UckisiList;
        public UckisiInfoList()
        {  _UckisiList = new List<UckisiInfo>();        }
        [MessageBodyMember]
        public List<UckisiInfo> UckisiList
        {    get { return _UckisiList; }          }
    }
}
```

სერვისის კონტრაქტი IFinalExamUckisi შეიცავს ერთადერთ მეთოდს LookupUckisi(). იგი მონაცემებს გადასცემს UckisiSearch კლასს, რომელსაც აქვს სახადასხვა თვისებები, საჭირო საგამოცდო უწყისის მოსაძებნად, მაგალითად, ლექტორი, საგანი, ჯგუფი. ის აბრუნებს უკან UckisiInfoList კლასს, რომელიც შეიცავს UckisiInfo კლასების კოლექციას. F6 ამოქმედებით აიგება გადაწყვეტა (solution). ამგვარად, ჩვენ განვსაზღვრეთ შეტყობინებები, რომლებიც გადაიცემა სერვის–მეთოდების მიერ პარამეტრების სახით.

7. Receive და SendReply კონფიგურირება

შემდეგ ეტაპზე FinalExamUckisi.xamlx –ში ReceiveRequest ქმედებისთვის OperationName თვისებაში ვათავსებთ LookupUckisi სახელს. WorkflowService-დიზაინერში შევქმნით ორ ახალ ცვლადს (Variables): search ცვლადი, შემომავალი შეტყობინების შესანახად და result ცვლადი, გამოსატანი შედეგისთვის (ნახ.8). Message ბუტონი უნდა იყოს ჩართული და ტიპებიც არჩეული.



ნახ. 8. შემავალი მოთხოვნის შეტყობინების და გამომავალი შედეგის ცვლადების განსაზღვრა

8. PerformLookup ქმედების აგება (ძებნის შესრულება)

ძებნის განსახორციელებლად (მონაცემთა ბაზებთან მიმართვის მიზნითაც) ვქმნით ახალ კლასს PerformLookup.cs, რომელიც Solution Explorer-ში პროექტისთვის FinalExamUckisi აირჩევა Add➤NewItem, Workflow-კატეგორიაში, Code Activity-ით. ამ ახალი კლასის ტექსტი მოცემულია მე-2 ლისტინგში.

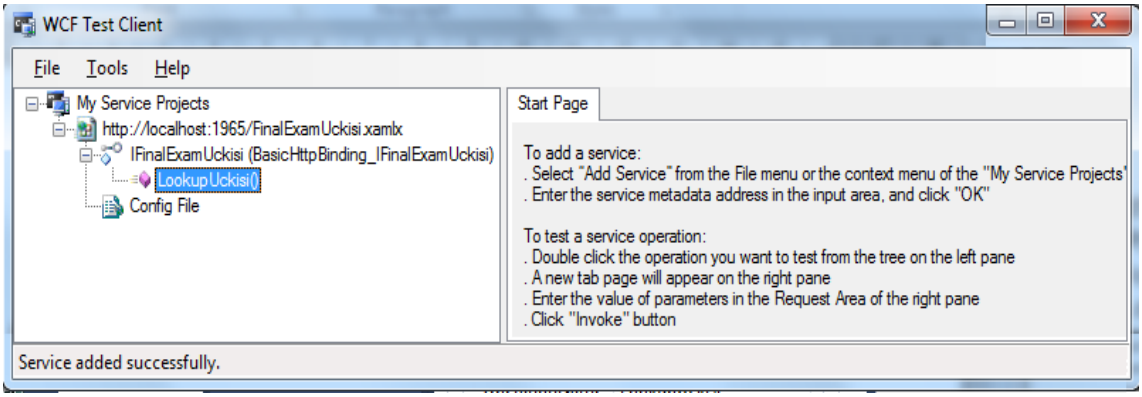
```
// ---- ლისტინგი 2 ----- PerformLookup -----------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Activities;

namespace FinalExamUckisi
{
  public sealed class PerformLookup : CodeActivity
   {
      public InArgument<UckisiSearch> Search { get; set; }
```

```
        public OutArgument<UckisiInfoList> UckisiList { get; set; }
        protected override void Execute(CodeActivityContext context)
        {   string lectori = Search.Get(context).Lectori;
            string sagani = Search.Get(context).Sagani;
            string jgupi = Search.Get(context).JGUPI;

            UckisiInfoList l = new UckisiInfoList();

            l.UckisiList.Add(new UckisiInfo(sagani, lectori, jgupi, "Available"));
            l.UckisiList.Add(new UckisiInfo(sagani, lectori, jgupi, "CheckedOut"));
            l.UckisiList.Add(new UckisiInfo(sagani, lectori, jgupi, "Missing"));
            l.UckisiList.Add(new UckisiInfo(sagani, lectori, jgupi, "Available"));
            UckisiList.Set(context, l);        }
    }
}
```

PerformLookup ქმედება ჩაემატება ინსტრუმენტების პანელზე. იგი უნდა გადმოვიტანოთ „ReceiveRequest" და „SendResponse" ქმედებებს შორის შესასრულებლად. ამასთანავე მისი Search თვისებისთვის ჩავწერეთ search, ხოლო UckisiList თვისებისთვის კი - result.

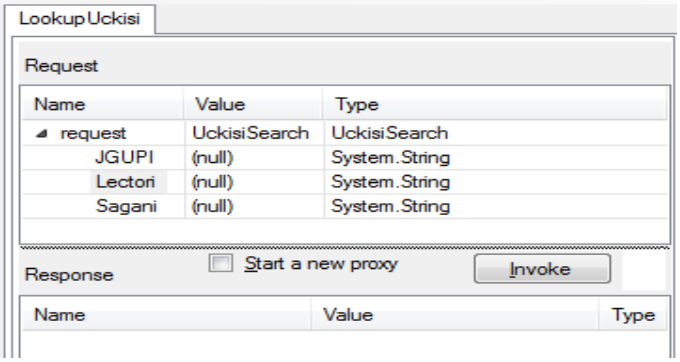
9. სერვისის ტესტირება

აპლიკაციის საბოლოო გამართვამდე უნდა მოვახდინოთ აგებული სერვისების ტესტირება. F5-ით ავამუშავოთ სერვისის გამართვის პროცედურა (debug). ვინაიდან ეს Web-სერვისია, Visual Studio ავტომატურად აამუშავებს WCF Test Client-ს [14]. ეს მეტად მოსახერხებელი უტილიტაა. იგი ჩატვირთავს Web-სერვისებს და აღმოაჩენს მეთოდებს, რომლებიც გათვალისწინებულია. ეს შედეგი ჩანს მე-9 ნახაზის მარცხენა პანელზე.



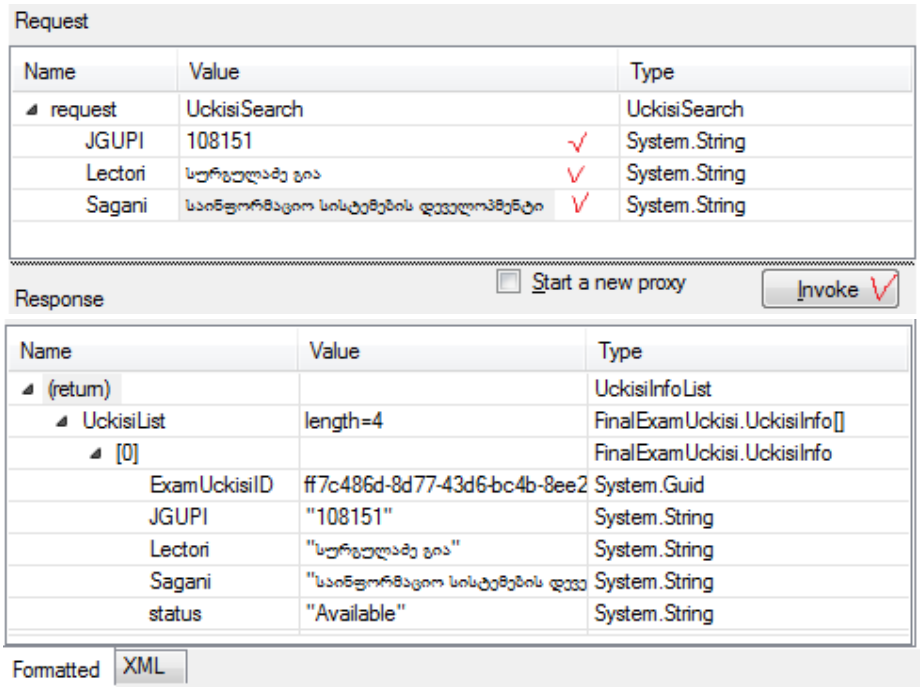
ნახ.9. ტესტირების პროცედურა

LookupUckisi() მეთოდზე 2-ჯერ დაჭერით მარჯვენა პანელის ზედა ნაწილში გამოიყოფა ადგილი შემოსული შეტყობინების განსათავსებლად (ნახ.10).



ნახ.10. საწყისი მონაცემების შესატანი ფანჯარა

შევიტანოთ კონკრეტული მნიშვნელობები Lectori, JGUPI, Sagani და ავამოქმედოთ Invoke ღილაკი. ცხრილებში გამოჩნდება შედეგები (ნახ.11).



ნახ.11. WCF Client Test უტილიტით სერვისის ტესტირების შედეგების ნახვა

მე-12 ნახაზზე ნაჩვენებია Request და Response ცხრილების შესაბამისი ფაილები XML ფორმატში.

ნახ.11. ტესტირების შედეგების XML ტექსტები

10. დასკვნა

ბიზნეს-პროცესები ხშირად განაწილებულია სხვადასხვა აპლიკაციებში და სხვადსხვა სერვერებზეც კი. ამიტომაც კომუნიკაცია არის ბიზნეს-პროცესის დიზაინის მნიშვნელოვანი ნაწილი. პროექტში, მაგალითად, ფაკულტეტები, დეპარტამენტები ან სხვა სტრუქტურული განყოფილებები ურთიერთქმედებს ერთმანეთთან, რათა გაცვალონ მონაცემები და შეტყობინებები, რომლებიც ექვემდებარება გადაცემას. Send და Receive ქმედებები უზრუნველყოფს შეტყობინებათა მოსახერხებელი ფორმით გადაცემას და მიღებას. ეს ქმედებები ეყრდნობა WCF-ს შეტყობინებების გადასაცემად და შეუძლია გამოიყენოს რიგი პროტოკოლებისა, როგორიცაა HTTP ან TCP.

მიუხედავად იმისა, რომ ბიზნეს-პროცესის ქმედებებს არ აქვს მომხმარებლის ინტერფეისი, ისინი ხშირად საჭიროებს ურთიერთობას ჰოსტ-აპლიკაციასთან, ან აპლიკაციის განახლებას, ან მოითხოვოს მონაცემები, რომლებიც შეიტანება მომხმარებელთა მიერ.

Web-სერვისების გამოყენება ხდება დაპროექტებისთვის სულ უფრო პოპულარული მიდგომა. საქმე შეიძლება დაწყებულ იქნას მომსახურების კონტრაქტის მიღებიდან, ან უბრალოდ განისაზღვროს შემავალი და გამომავალი პარამეტრები, ბიზნეს-პროცესების კონსტრუქტორის გამოყენებით. მუშა პროცესი შეიძლება გამოყენებულ იქნას როგორც სერვისის შესაქმნელად და მათ გამოსაყენებლად. მეთოდები, რომლებიც უზრუნველყოფილია Web-სერვისებით, ხდება ტრადიციული, სტანდარტული ქმედებები, რომელთა გამოყენება შესაძლებელია ბიზნეს-პროცესებში.

ლიტერატურა:

1. სურგულაძე გ., ბულია ი. კორპორაციულ Web–აპლიკაციათა ინტეგრაცია და დაპროექტება. მონოგრ., სტუ. თბ., 2012

2. ბაკურია კ. სასწავლო პროცესის ინფორმაციული მხარდაჭერის ავტომატიზებული სისტემის ძირითადი ელემენტები. სტუ-ს შრ.კრ. №1(491), საგამომც. სახლი ,,ტექნიკური უნივერსიტეტი", თბ., 2014. გვ.83-88

3. BSI-Standard 100-1: Managementsysteme für Informationssicherheit (ISMS). Bundesamt für Sicherheit in der Informationstechnik, (BSI) Godesberger Allee 185-189, 53175 Bonn, 2008/2013.

4. ITIL moving towards Enterprise Architecture. http://blogs.msdn.com/b/mikewalker /archive/2007/07/06/itil-moving-towards-enterprise-architecture.aspx?Redirected=true

5. COBIT: Framework for IT Governance and Control. http://www.isaca.org/knowledge-center/cobit/Pages/Overview.aspx

6. Booch G., Jacobson I., Rumbaugh J. Unified Modeling Language for Object-Oriented Development. Rational Software Corporation, Santa Clara, 2006

7. Halpin T. ORM-2 Graphical Notation. Neumont Univer., 2005. http://www.orm.net/ pdf/ORM2_TechReport1.pdf

8. თურქია ე. ბიზნეს–პროექტების მართვის ტექნოლოგი-ური პროცესების ავტომატიზაცია. მონოგრ., სტუ. თბ., 2010

9. Surguladze G., Turkia E., Topuria N., Lominadze T., Giutashvili M. Automation of Business-Processes of an Election System. IV-Intern.Conf. "Problems of Cybernetics and Informatics" (PCI' 2012). Baku, Azerbaijan, 2012. pp. 16-19

10.ვედეკინდი ჰ., სურგულაძე გ., თოფურია ნ. განაწილებული ოფის-სისტემების მონაცემთა ბაზების დაპროექტება და რეალიზაცია UML-ტექნოლოგიით. მონოგრ., სტუ, თბ., 2006

11. Collins M.J. Beginning WF: Windows Workflow in .NET 4.0. ISBN-13 (pbk): 978-1-4302-2485-3 Copyright © 2010. USA. http://www.ebooks-it.net/ebook/beginning-wf

12. Уотсон К.,Нейгел К., Педерсен Я., ХаммерР., Джон Д., Скиннер М., Уайт Э. Visual C# 2008: базовый курс. : Пер. с англ. - М. : ООО "И.Д. Вильямс", 2009

13. Petzold Ch. Applications=Code+Markup. A Guide to the MicroSoft Windows Presentation Foundation. St-Petersburg. 2008

14. Anurag S. WCF: From a Beginner's perspective & a Tutorial. V, 4 Apr 2013. http://www.codeproject.com/Articles/566691/WCF-From-a-Beginners-perspective-a-Tutorial

DESIGN OF INFORMATION SYSTEM ON BASED OF OBJECT-ROLE MODELLING AND SERVICE-ORIENTED ARCHITECTURE

Surguladze Gia, Topuria Nino, Bakuria Koba, Lomidze Maka

Georgian Technical University

Summary

In this work is considered issues of design and software implementation of a distributed information system of the University, and user interfaces based on hybrid technology and service-oriented architecture. Proposed the results of the processes of constructing database MsSQL Server via the serial automated conversion components “Facts / ORM / ERM / DDL" study the problem area. Also the development of Web-services user interfaces using Workflow and Windows Communication Foundation technologies.

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ НА БАЗЕ ОБЪЕКТНО-РОЛЕВОГО МОДЕЛИРОВАНИЯ И СЕРВИС-ОРИЕНТИРОВАННОЙ АРХИТЕКТУРЫ

Сургуладзе Г., Топурия Н., Бакурия К., Ломидзе М.

ГрузинскийТехническийУниверситет

Резюме

Рассматриваются вопросы проектирования и программной реализации распределенной информационной системы Университета и пользовательских интерфейсов на основе гибридных технологий и сервис-ориентированной архитектуры. Предлагаются результаты процессов построения базы данных MsSQL Server посредством последовательного автоматизированного преобразования компонентов „Facts /ORM / ERM / DDL” исследуемой проблемной области. Также предложена разработка Web-сервисов пользовательских интерфейсов с помощью Workflow и Windows Communication Foundation технологий.