

**EGE UNIVERSITY**

**COMPUTER ENGINEERING**

**ONTOLOGY DEVELOPMENT FOR  
HUMAN RESOURCES MANAGEMENT**

**Project Developers**

- ⇒ 05130000216 – Dilşen YILDAR
- ⇒ 05130000275 – Eray HAVAYLAR

June, 2017

Proje için İnsan Kaynakları Yönetimi (İKY) kavramları içeren bir ontoloji geliştirmeyi amaçlıyoruz: İş portallarında staj arama için bir arama motoru.

İKY veya daha özel olarak yetkinlik yönetimi, bir organizasyonun en önemli kaynağı olan personelinin bilgisi ve becerileri nedeniyle bilgi yönetiminin önemli bir fonksiyonudur. İK yönetiminde işe alım süreci önemli bir bölümdür. Biz projemizde şirketlerin kendilerine en uygun aday bulmaları ve staj arayan bir kişinin kendisine en uygun staj yerini bulmasını sağlamaya yönelik bir çalışma yapmayı hedefliyoruz.

**Scope:** Bilişim, Staj, Zorunlu Staj, Programlama Dilleri, Microsoft bünyesindeki diller ve teknolojiler

## Step 1: Competency questions

1. Adayın iş gereksinimleri ve yeterlilik profilleri nelerdir?
2. Adayın kişisel nitelikleri nelerdir?(Behavioral Competencies)
3. Adayın bilgileri ve yetenekleri nelerdir?(Functional Competencies)
4. Aday için staj zorunlu mu?
5. Aday Microsoft bünyesindeki dillerin ne kadarını biliyor?
6. Aday Microsoft programlama dillerinden hangilerini biliyor?
7. Aday Microsoft programlarından hangilerini biliyor?
8. Aday Microsoft IDE ve araçlarından hangilerini biliyor ve kullanıyor?
9. Aday bildiği dillerin yüzde kaçına hakim?(Orta seviye ve üstü olanlar)
10. Bu diller hangi üniversitelerde ders olarak verilir?
11. Aday kaç tane yabancı dil konuşabiliyor?
12. Adayın bildiği yabancı diller ile ilgili sertifikası var mı?
13. Bilişimde hangi sertifika eğitimleri alınır?(Profilini doldurmak isteyen kişinin ne yapmam gerekiyor sorusuna cevap gibi)
14. Benzer Microsoft dillerini kullanan şirketler hangileridir?
15. Aynı iş modelini uygulayan şirketler hangileridir?
16. Aynı bölümden mezun olan kişiler nerelerde çalışıyorlar?
17. Aday şirket çalışanlarına uygun mu?(Alkol, sigara kullanımı gibi)
18. Adayın dahil olduğu projelerde Microsoft dillerinden hangileri kullanılmıştır?
19. Adayın çalışmak istediği departman nedir?
20. Adayın başvurduğu şirket hangi departmanlara sahiptir?

## Step 2: Consider reusing existing ontologies

1. <http://www.ec.tuwien.ac.at/~dorn/Papers/ICKM2007.pdf> : İlk üç soru için bu makaleden faydalananak yol almaya başladık.
2. <http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/99-hrmontology/> : Çok geniş bir IK Ontolojisi var. Burdan faydalananak scope’u daralttık.
3. [http://www.starlab.vub.ac.be/staff/mustafa/ontocontent06/aschmidt\\_HRDOntology\\_OntoContent06.pdf](http://www.starlab.vub.ac.be/staff/mustafa/ontocontent06/aschmidt_HRDOntology_OntoContent06.pdf) : Adayın yabancı dil bilgisi ile ilgili soruları bu kaynaktan faydalananak eklemeyi tercih ettik.
4. [http://xmlns.com/foaf/spec/#term\\_homepage](http://xmlns.com/foaf/spec/#term_homepage) : Adayın kişisel bilgilerini tutmak için faydalanaabileceğimiz FOAF ontolojisini inceledik ve ontolojimize import ettik.
5. [https://msdn.microsoft.com/en-us/library/aa292164\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa292164(v=vs.71).aspx) : Microsoft dilleri hangileri, aday bunların ne kadarını biliyor gibi sorulara cevap verebilmek için Microsoft’un sitesinden faydalandık.
6. [https://my.visualstudio.com/benefits?wt.mc\\_id=o~msft~devprogram~header&amp;campaign=o~msft~devprogram~header](https://my.visualstudio.com/benefits?wt.mc_id=o~msft~devprogram~header&amp;campaign=o~msft~devprogram~header) : Microsoft dillerine ve araçlarını araştırabilmek için bu linkten faydalandık.
7. <https://hockeyapp.net> , <https://azure.microsoft.com/tr-tr/services/application-insights/>
8. <http://protege.stanford.edu/conference/2009/slides/SWRL2009ProtegeConference.pdf> SWRL kurallarımızı yazarken faydalandığımız bir kaynak.
9. <https://www.youtube.com/watch?v=2aJudF8AnBI&list=PLD8uCWff9n-EG4KK2OAiPRSCPgNJXf49j&index=4> ve [http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4\\_v1\\_3.pdf](http://mowl-power.cs.man.ac.uk/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf) ‘dan class hierarchy, object ve data property, individuallarımızı yazarken yararlandık.
10. SWRL Kurallarını yazarken faydalandığımız birkaç kaynak:  
<https://dior.ics.muni.cz/~makub/owl/#swrl> ,  
<https://stackoverflow.com/questions/21540839/swrl-rules-dont-infer-new-object-and-data-property-assertions> , <https://www.w3.org/Submission/SWRL/>  
 , <http://protege.stanford.edu/conference/2009/slides/SWRL2009ProtegeConference.pdf>
11. Sorgularımızı yazarken yararlandığımız kaynaklar : Semantic Web for the Working Ontologist Dean Allemang, James Hendler 2008 ,  
<https://www.w3.org/2009/Talks/0615-qbe/>

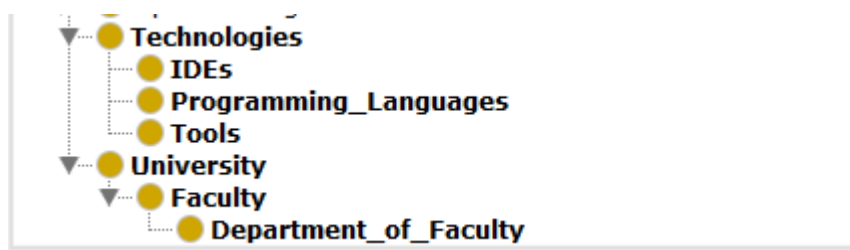
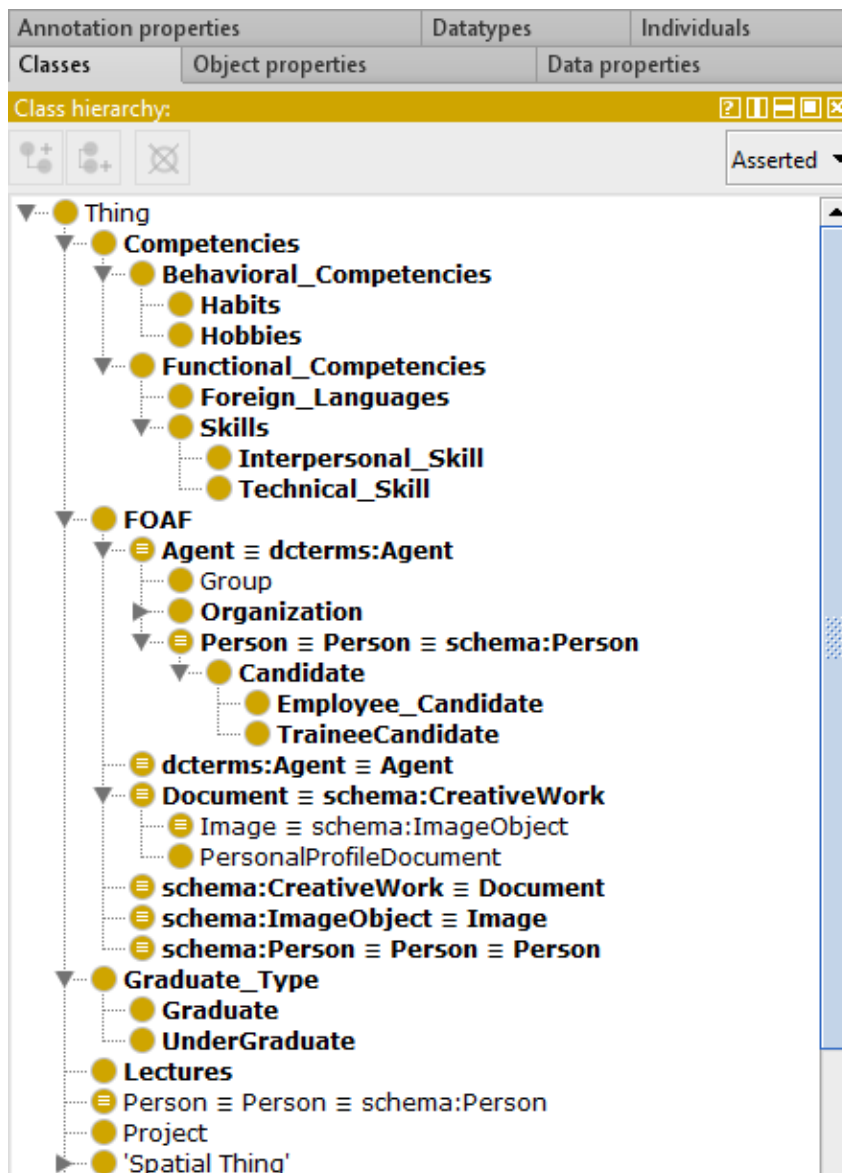
### Step 3 : Enumerate important terms in the ontology

- ↳ Aday
- ↳ İş gereksinimi
- ↳ Şirketler
- ↳ Departmanlar
- ↳ Profil
- ↳ Yeterlilik profilleri
- ↳ Kişisel nitelik
- ↳ Bilgi
- ↳ Yetenek
- ↳ Staj
- ↳ Zorunlu staj
- ↳ Stajyer
- ↳ Teknolojiler
- ↳ Programlama dilleri
- ↳ Microsoft Teknolojileri
- ↳ Microsoft Programlama Dilleri
- ↳ Visual C# .NET
- ↳ ASP.NET
- ↳ Visual C++ .NET
- ↳ Visual Basic .NET
- ↳ Transact-SQL
- ↳ Extensible Markup Language (XML)
- ↳ Scripting Languages
- ↳ VBScript
- ↳ JScript
- ↳ JScript .NET
- ↳ Visual J++
- ↳ Simple Object Access Protocol(SOAP)
- ↳ IDEs
- ↳ Visual Studio Community
- ↳ Visual Studio Code
- ↳ Visual Studio Mac
- ↳ Visual Studio Team Service
- ↳ Azure
- ↳ Xamarin
- ↳ Tools

- ↳ Microsoft R Server
- ↳ Microsoft SQL Server
- ↳ Azure App Service
- ↳ Application Insights
- ↳ Hockey App
- ↳ Universal Windows Platform VM
- ↳ Bilmek
- ↳ Yüzde
- ↳ Üniversite
- ↳ Ders vermek,
- ↳ Yabancı dil bilgisi
- ↳ Konuşmak
- ↳ Sertifikalar
- ↳ Sertifika eğitimi
- ↳ Almak
- ↳ Mezun
- ↳ Bölüm
- ↳ Çalışmak
- ↳ Alışkanlık
- ↳ Hobi
- ↳ Proje
- ↳ Dahil olmak
- ↳ Kullanmak
- ↳ Başvurmak.
- ↳ Çalışmak ister.

#### **Step 4: Define the classes and the class hierarchy**

Sınıflarımızı ve gerekli alt sınıflarımızı oluştururken top-down geliştirme sürecini uyguladık. Daha anlaşılır ve doğru olması açısından bu yöntemi kullanmayı daha uygun bulduk. Aşağıda ontolojimizin bir görüntüsü bulunmaktadır. Varolan FOAF (Friend-of-a-Friend) ontolojisini import ederek kendi ontolojimize uygun sınıflar oluşturup genişlettik.



## Step 5: Define the properties of classes—slots

Sınıflarımız için belirlediğimiz object property'lerimiz ve data property'lerimiz aşağıda görüntülenmektedir. Object property'lerimizin domain ve range'lerine ait birkaç örnek de bulunmaktadır. Object property'lerimizden birkaçını açıklayacak olursak;

**beParticipate:** Candidate beParticipate Project. (Person1 beParticipate .NET\_Project1)

**codeWith:** Project codeWith Programming\_Languages (.NET\_Project1 codeWith ASP.NET)

**code\_with\_language:** beParticipate and codeWith Subproperty of this object property. Candidate beParticipate Project, Project codeWith Programming\_Languages → Candidate code\_with\_language Programming\_Languages (Person1 beParticipate .NET\_Project1. .NET\_Project1 codeWith ASP.NET → Person1 code\_with\_language ASP.NET) \*\*\*(by rule)

**developedBy:** IDEs, Tools, Programming\_Languages developedBy Organization. \*\*\*(inverseOf isDeveloperOf)

bu şekilde gerçekleştirilir.

- owl:topObjectProperty
  - code\_with\_languages
  - account
  - account
  - 'account service homepage'
  - 'based near'
  - beParticipate
  - codeWith
  - contains
  - 'current project'
  - depiction
  - depicts
  - developedBy
    - developedIdeBy
    - developedProgLangBy
    - developedToolBy
  - educatedBy
  - focus
  - 'funded by'
  - has
    - hasBehavioral
    - hasDepartment
    - hasDept
    - hasFunctional
    - hasGraduateType
    - HasSome
    - interest
  - isApply
  - isDeveloperOf
    - isDeveloperIdeOf
    - isDeveloperProgLangOf
    - isDeveloperToolOf
  - isEmpKnow
  - isHired
  - isKnow
  - isUseOf
    - isUseIdeOf
    - isUseProgLangOf
    - isUseToolOf
  - isWorked
  - 'jabber ID'
  - knows
  - logo
  - made
  - maker
  - member
  - nickname
  - page
  - 'past project'
  - 'personal mailbox'
  - phone
  - 'primary topic'
  - publications
  - schoolHomepage
  - 'sha1sum of a personal mailbox URI name'
  - theme
  - thumbnail
  - topic
  - topic\_interest
  - UsedBy
    - UsedIdeBy
    - UsedProgLangBy
    - UsedToolBy
  - 'work info homepage'
  - 'workplace homepage'

Description: beParticipate

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

**Candidate** ? @ x o

Ranges (intersection) +

**Project** ? @ x o

Disjoint With +

SuperProperty Of (Chain) +

Description: isApply

Equivalent To +

SubProperty Of +

Inverse Of +

Domains (intersection) +

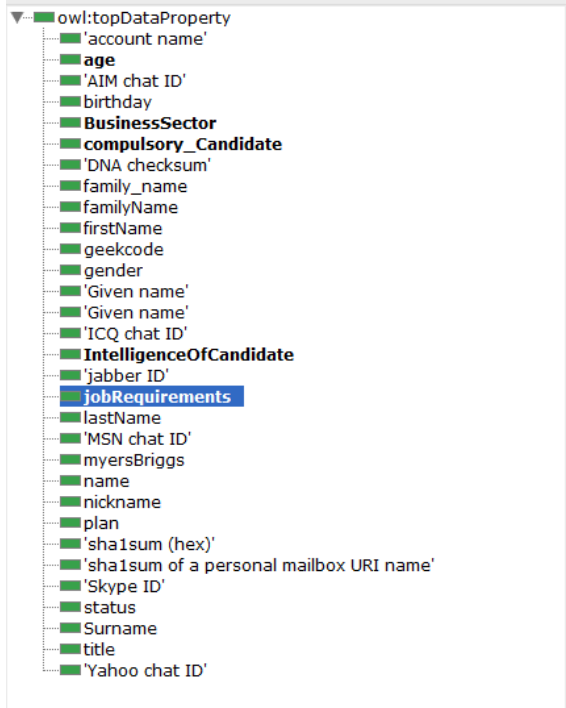
**Candidate** ? @ x o

Ranges (intersection) +

**Organization** ? @ x o

Disjoint With +





#### jobRequirements: Person1

jobRequirements “Computer Engineering” şeklinde kullanıyoruz. İş gereksinimi dediğimiz şey aslında bu. Adayın ihtiyacı olan mesleği bu sayede öğreniyoruz.

**BusinessSector:** Şirketin hangi sektörde olduğunu öğrenmemizi sağlıyor.

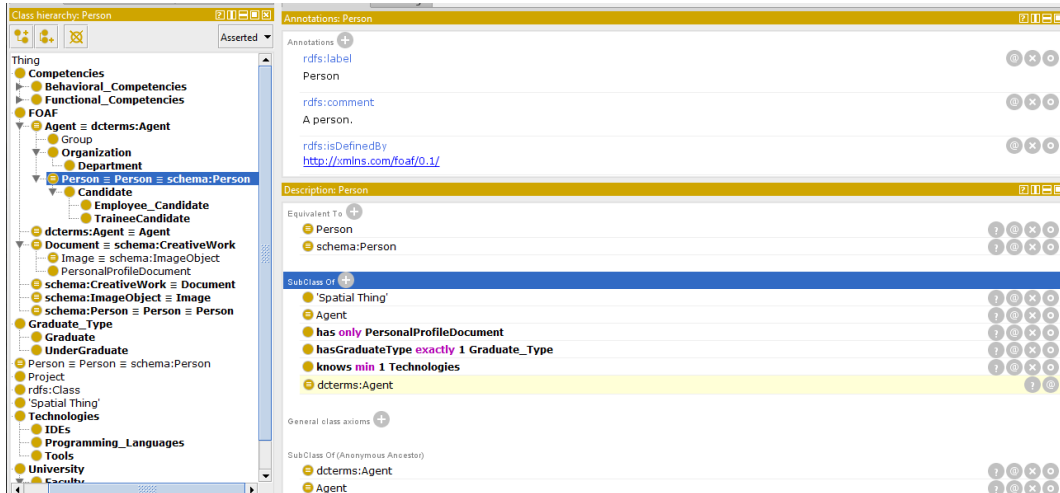
**compulsory\_Candidate:** Stajyer adayın stajının zorunlu olup olmadığını boolean bir değer ile tutmaktayız.

**IntelligenceOfCandidate:** Adayımızın 18 yaşından küçük olduğu ve üniversiteye kabul edildiği istisna durumlarda üstün zekalı olduğunu data property’inde belirtiyoruz.

## Step 6: Define the facets of the slots

Person sınıfımız için bir mezun olup olmama durumu belirlememiz gerekmektedir, çünkü ontolojimiz stajyerlik üzerine olduğu için adayımızın henüz üniversiteden mezun olmamış olması gerekiyor. hasGraduateType object property ile yalnızca 1 graduate\_type olabilir şeklinde belirtiyoruz.

Person’ın en az 1 teknoloji bilmesi durumunun cardinality’ sini belirledik, bir şirkete başvururken herhangi bir programlama dilinin bilinmesi ayrıcalık olabilmektedir. Şirketler kendilerine uygun adayı da bu şekilde daha kolay bulabilir. Örneğin, ontolojimizde Microsoft dillerini bilen stajyerler Java kullanan bir şirkete başvururken Java bilmiyorsa o şirkete girme ihtimali Java bilen adaylara göre daha az olacaktır.



Data property hierarchy: compulsory\_Candidate

owl:topDataProperty

- 'account name'
- age
- 'AIM chat ID'
- birthday
- BusinessSector
- compulsory\_Candidate**
- 'DNA checksum'
- family\_name
- familyName
- firstName
- geekcode
- gender
- 'Given name'
- 'Given name'
- 'ICQ chat ID'
- IntelligenceOfCandidate
- 'jabber ID'
- jobRequirements
- lastName
- 'MSN chat ID'
- myersBriggs
- name
- nickname
- plan
- 'sha1sum (hex)'
- 'sha1sum of a personal mailbox URI name'
- 'Skype ID'
- status
- Surname
- title
- 'Yahoo chat ID'

Annotations: compulsory\_Candidate

Annotations +

Characteristics: compulsory\_Candidate

Functional

Description: compulsory\_Candidate

Equivalent To +

SubProperty Of +

Domains (Intersection) +

- TraineeCandidate

Ranges +

- xsd:boolean

Disjoint With +

Data property hierarchy: jobRequirements

owl:topDataProperty

- 'account name'
- age
- 'AIM chat ID'
- birthday
- BusinessSector
- compulsory\_Candidate
- 'DNA checksum'
- family\_name
- familyName
- firstName
- geekcode
- gender
- 'Given name'
- 'Given name'
- 'ICQ chat ID'
- IntelligenceOfCandidate
- 'jabber ID'
- jobRequirements**
- lastName
- 'MSN chat ID'
- myersBriggs
- name
- nickname
- plan
- 'sha1sum (hex)'
- 'sha1sum of a personal mailbox URI name'
- 'Skype ID'
- status
- Surname
- title
- 'Yahoo chat ID'

Annotations: jobRequirements

Annotations +

Characteristics: jobRequirements

Functional

Description: jobRequirements

Equivalent To +

SubProperty Of +

Domains (Intersection) +

- Person

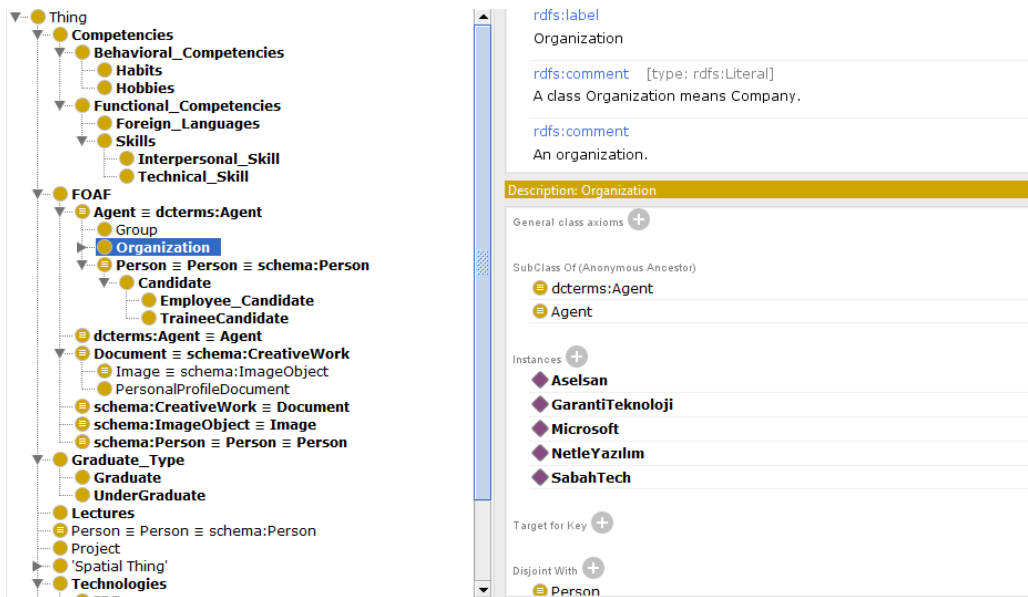
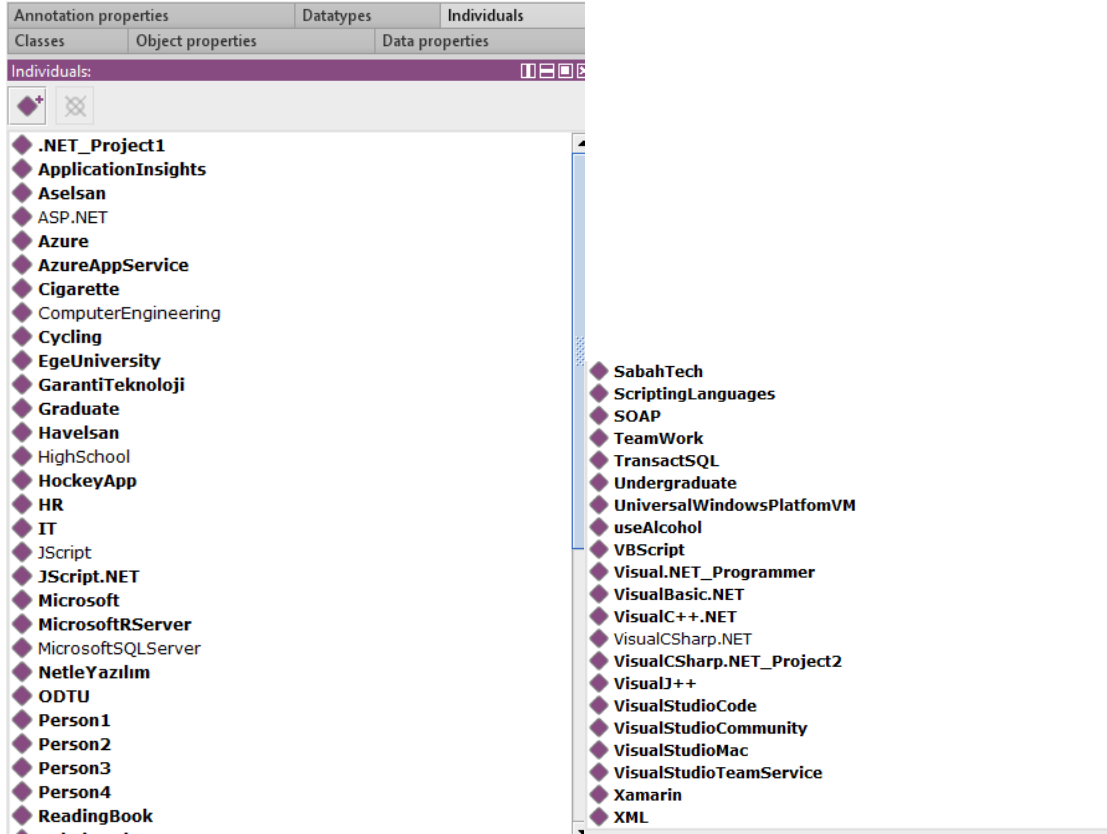
Ranges +

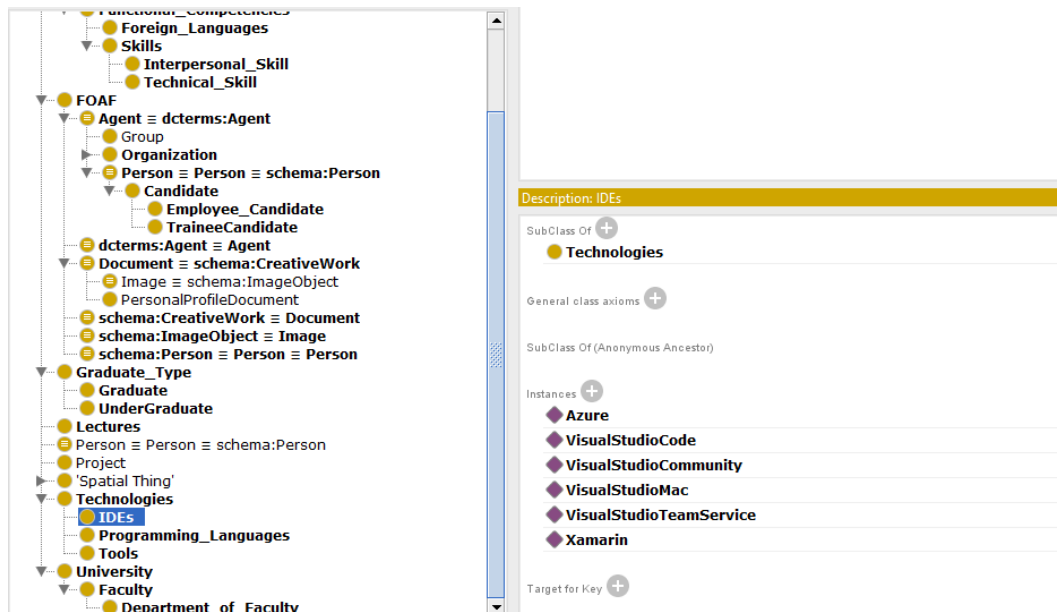
- rdfs:Literal

Disjoint With +

## Step 7: Create instances

Ontolojimize individuals ekleyerek gerekli sınıflara instances olarak entegre ettik.











## Property assertions: Individual


Bazı individuallarımıza eklediğimiz property assertionlarımız aşağıdaki gibidir:




Property assertions: Person1	
Object property assertions +	
hasFunctional	TeamWork
hasBehavioral	Cycling
beParticipate	.NET_Project1
isApply	Havelsan
hasGraduateType	Undergraduate
isKnow	ASP.NET
beParticipate	VisualCSharp.NET_Project2
isKnow	VisualCSharp.NET
educatedBy	EgeUniversity
Data property assertions +	
firstName	"Dilsen"^^rdfs:Literal
jobRequirements	"Computer Engineering"^^rdfs:Literal
age	23
compulsory_Candidate	true

## Property assertions: Person2

Object property assertions 

-  **hasGraduateType** Graduate
-  **hasBehavioral** ReadingBook
-  **beParticipate** .NET\_Project1
-  **isEmpKnow** ASP.NET
-  **educatedBy** EgeUniversity


Data property assertions 






-  **firstName** "Eray"^^**rdfs:Literal**
-  **jobRequirements** "Computer Engineering"^^**rdfs:Literal**
-  **age** 24

## Property assertions: Havelsan

Object property assertions 

-  **hasDepartment** HR    
-  **isHired** Person1    

Data property assertions 

-  **BusinessSector** "Information Technologies"^^**rdfs:Literal**    

Negative object property assertions 

Negative data property assertions 

## Step 8: Rules

Active Ontology	Entities	Individuals by class	DL Query	OntoGraf	SWRLTab	SQWRLTab	SPARQL Query
	Name	Rule					
<input checked="" type="checkbox"/>	Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)					
<input checked="" type="checkbox"/>	Rule2	TraineeCandidate(?trC) ^ beParticipate(?trC, ?project) -> Project(?project)					
<input checked="" type="checkbox"/>	Rule3	Project(?x) ^ codeWith(?x, ?prolang) -> Programming_Languages(?prolang)					
<input checked="" type="checkbox"/>	Rule4	University(?uni) ^ hasDept(?uni, ?deptFac) -> Department_of_Faculty(?deptFac)					
<input checked="" type="checkbox"/>	Rule5	Candidate(?p) ^ isApply(?p, ?c) ^ Organization(?c) ^ isHired(?w, ?p) -> isWorked(?p, ?w)					
<input checked="" type="checkbox"/>	Rule6	Person(?p) ^ age(?p, ?y) ^ swrlb:lessThan(?y, 18) ^ educatedBy(?p, ?uni) ^ University(?uni) -> IntelligenceOfCandidate(?p, "highIntelligence")					

### Rule 1 and its output:

Name	Rule
Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)

Person1 isApply Havelsan, so Havelsan is an Organization.

Property assertions: Person1

Object property assertions +

- hasFunctional TeamWork
- hasBehavioral Cycling
- beParticipate .NET\_Project1
- isApply Havelsan
- hasGraduateType Undergraduate
- isKnow ASP.NET
- beParticipate VisualCSharp.NET\_Project2
- isKnow VisualCSharp.NET
- educatedBy EgeUniversity

Description: Havelsan

Types +

- Organization

Same Individual As +

Different Individuals +

## Rule 2 and its output:

Name	Rule
Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)
Rule2	TraineeCandidate(?trC) ^ beParticipate(?trC, ?project) -> Project(?project)

Person1 beParticipate .NET\_Project, so .NET\_Project is an Project.

The screenshot displays a software interface with two main panels. The left panel, titled 'Property assertions: Person1', lists various object and data property assertions for a person named Person1. The right panel, titled 'Description: .NET\_Project1', shows the types associated with the project.

**Property assertions: Person1**

- Object property assertions:**
  - hasBehavioral: Cycling
  - hasFunctional: TeamWork
  - beParticipate: .NET\_Project1
  - isApply: Havelsan
  - hasGraduateType: Undergraduate
  - isKnow: ASP.NET
  - beParticipate: VisualCSharp.NET\_Project2
  - isKnow: VisualCSharp.NET
  - educatedBy: EgeUniversity
  - code\_with\_languages: VisualCSharp.NET
  - code\_with\_languages: ASP.NET
  - isWorked: Havelsan
- Data property assertions:**
  - firstName: "Dilsen"^^rdfs:Literal
  - jobRequirements: "Computer Engineering"^^rdfs:Literal
  - age: 23
  - compulsory\_Candidate: true

**Description: .NET\_Project1**

- Types:**
  - Project
- Same Individual As:** (empty)
- Different Individuals:** (empty)

## Rule 3 and its output:

Name	Rule
Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)
Rule2	TraineeCandidate(?trC) ^ beParticipate(?trC, ?project) -> Project(?project)
Rule3	Project(?x) ^ codeWith(?x, ?prolang) -> Programming_Languages(?prolang)

.NET\_Project codeWith ASP.NET, so ASP.NET is a Programming\_Languages

The screenshot displays a software interface with two main panels. The left panel, titled 'Property assertions: .NET\_Project1', lists various object and data property assertions for the project .NET\_Project1. The right panel, titled 'Description: ASP.NET', shows the types associated with the programming language ASP.NET.

**Property assertions: .NET\_Project1**

- Object property assertions:**
  - codeWith: ASP.NET
- Data property assertions:** (empty)

**Description: ASP.NET**

- Types:**
  - Programming\_Languages

Rule2 ve Rule3'te yazdığımız `code_with_languages`'in subproperty'lerinden dolayı aşağıda ok ile gösterdiğimiz çıktı da Person1 için gelir.

Property assertions: Person1

Object property assertions +

- hasBehavioral Cycling
- hasFunctional TeamWork
- beParticipate .NET\_Project1
- isApply Havelsan
- hasGraduateType Undergraduate
- isKnow ASP.NET
- beParticipate VisualCSharp.NET\_Project2
- isKnow VisualCSharp.NET
- educatedBy EgeUniversity
- code\_with\_languages VisualCSharp.NET
- code\_with\_languages ASP.NET
- isWorked Havelsan

Data property assertions +

- firstName "Dilsen"^^rdfs:Literal
- jobRequirements "Computer Engineering"^^rdfs:Literal
- age 23
- compulsory\_Candidate true

## Rule 4 and its output:

Name	Rule
Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)
Rule2	TraineeCandidate(?trC) ^ beParticipate(?trC, ?project) -> Project(?project)
Rule3	Project(?x) ^ codeWith(?x, ?prolang) -> Programming_Languages(?prolang)
Rule4	University(?uni) ^ hasDept(?uni, ?deptFac) -> Department_of_Faculty(?deptFac)

EgeUniversity hasDept ComputerEngineering, so ComputerEngineering is a Department\_of\_Faculty.

Property assertions: EgeUniversity

Object property assertions +

- hasDept ComputerEngineering

Description: ComputerEngineering

Types +

- Department\_of\_Faculty



## Rule 5 and its output:

Name	Rule
Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)
Rule2	TraineeCandidate(?trC) ^ beParticipate(?trC, ?project) -> Project(?project)
Rule3	Project(?x) ^ codeWith(?x, ?prolang) -> Programming_Languages(?prolang)
Rule4	University(?uni) ^ hasDept(?uni, ?deptFac) -> Department_of_Faculty(?deptFac)
Rule5	Candidate(?p) ^ isApply(?p, ?c) ^ Organization(?c) ^ isHired(?w, ?p) -> isWorked(?p, ?w)

Person1 isApply Havelsan and Havelsan isHired Person1, so Person1 isWorked Havelsan.

Property assertions: Person1

Object property assertions

- hasBehavioral Cycling
- hasFunctional TeamWork
- beParticipate .NET\_Project1
- isApply Havelsan
- hasGraduateType Undergraduate
- isKnow ASP.NET
- beParticipate VisualCSharp.NET\_Project2
- isKnow VisualCSharp.NET
- educatedBy EgeUniversity
- code\_with\_languages VisualCSharp.NET
- code\_with\_languages ASP.NET
- isWorked Havelsan

Data property assertions

- firstName "Dilsen"^^rdfs:Literal
- jobRequirements "Computer Engineering"^^rdfs:Literal
- age 23
- compulsory\_Candidate true

Property assertions: Havelsan

Object property assertions

- hasDepartment HR
- isHired Person1

Data property assertions

- BusinessSector "Information Technologies"^^rdfs:Literal

Property assertions: Person1

Object property assertions

- isApply Havelsan
- hasGraduateType Undergraduate
- isKnow ASP.NET
- beParticipate VisualCSharp.NET\_Project2
- isKnow VisualCSharp.NET
- educatedBy EgeUniversity
- code\_with\_languages VisualCSharp.NET
- code\_with\_languages ASP.NET
- isWorked Havelsan

Data property assertions

- firstName "Dilsen"^^rdfs:Literal
- jobRequirements "Computer Engineering"^^rdfs:Literal
- age 23
- compulsory\_Candidate true

## Rule 6 and its output:

Name	Rule
Rule1	Candidate(?p) ^ isApply(?p, ?company) -> Organization(?company)
Rule2	TraineeCandidate(?trC) ^ beParticipate(?trC, ?project) -> Project(?project)
Rule3	Project(?x) ^ codeWith(?x, ?prolang) -> Programming_Languages(?prolang)
Rule4	University(?uni) ^ hasDept(?uni, ?deptFac) -> Department_of_Faculty(?deptFac)
Rule5	Candidate(?p) ^ isApply(?p, ?c) ^ Organization(?c) ^ isHired(?w, ?p) -> isWorked(?p, ?w)
Rule6	Person(?p) ^ age(?p, ?y) ^ swrlb:lessThan(?y, 18) ^ educatedBy(?p, ?uni) ^ University(?uni) -> IntelligenceOfCandidate(?p, "highIntelligence")

Person3 hasAges lessThan 18 and goes to University, so Person3 is "highIntelligence"

**Property assertions: Person3**

Object property assertions

**educatedBy** ODTU

Data property assertions

**age** 16

**IntelligenceOfCandidate** "highIntelligence"^^xsd:string

Negative object property assertions

Negative data property assertions

**Property assertions: Person4**

Object property assertions

Data property assertions

**age** 15

Negative object property assertions

**Property assertions: Person1**

Object property assertions

**hasFunctional** TeamWork

**hasBehavioral** Cycling

**beParticipate** .NET\_Project1

**isApply** Havelsan

**hasGraduateType** Undergraduate

**isKnow** ASP.NET

**educatedBy** EgeUniversity

**isWorked** Havelsan

**traniee\_code\_with\_language** ASP.NET

Data property assertions

**firstName** "Dilsen"^^rdf:Literal

**jobRequirements** "Computer Engineering"^^rdf:Literal

**age** 23

**compulsory\_Candidate** true

Negative object property assertions

## Step 9: Queries

PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

### Query 1-

➔ Ege Üniversitesinde eğitim görenlerin isimleri

SELECT ?Candidate ?name

WHERE { ?Candidate hrOnt:educatedBy hrOnt:EgeUniversity

foaf:firstName ?name .

}

SPARQL query:	
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX foaf: <http://xmlns.com/foaf/0.1/> PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>  SELECT ?Candidate ?name WHERE { ?Candidate hrOnt:educatedBy hrOnt:EgeUniversity; foaf:firstName ?name . }	
Candidate	name
Person2	"Eray"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
Person1	"Dilsen"^^<http://www.w3.org/2000/01/rdf-schema#Literal>

➔ ASP.NET bilen stajyer adayları

SELECT ?TraineeCandidate WHERE { ?TraineeCandidate hrOnt:isKnow  
hrOnt:ASP.NET }

SPARQL query:	
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX owl: <http://www.w3.org/2002/07/owl#> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>  SELECT ?TraineeCandidate WHERE { ?TraineeCandidate hrOnt:isKnow hrOnt:ASP.NET }	
TraineeCandidate	
Person1	

## Query 2-

➔ Ege Üniversitesinde okuyan bir stajyer adayının başvurduğu şirketler

```
SELECT ?company WHERE {  
  
    ?traineeCandidate hrOnt:isApply ?company .  
  
    ?traineeCandidate hrOnt:educatedBy hrOnt:EgeUniversity .  
  
}
```

```
PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>
```

```
SELECT ?company WHERE {  
    ?traineeCandidate hrOnt:isApply ?company .  
    ?traineeCandidate hrOnt:educatedBy hrOnt:EgeUniversity .  
}
```

company
Havelsan

➔ Şirketler ve kullanılan tool'lar

```
SELECT ?company ?tool  
  
WHERE { ?tool hrOnt:isUseToolOf ?company }
```

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>  
PREFIX owl: <http://www.w3.org/2002/07/owl#>  
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>  
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  
PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>
```

```
SELECT ?company ?tool  
WHERE { ?tool hrOnt:isUseToolOf ?company }
```

company	tool
MicrosoftSQLServer	GarantiTeknoloji

### Query 3-

➔ Bildiği dillerde kodlama yapılan projelerde bulunan kişiler

SELECT ?traineeCandidate ?Project

WHERE { ?traineeCandidate hrOnt:isKnow ?programmingLanguages .

?traineeCandidate hrOnt:beParticipate ?Project .

?Project hrOnt:codeWith ?programmingLanguages . }

SPARQL query:	
PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>	
SELECT ?traineeCandidate ?Project	
WHERE { ?traineeCandidate hrOnt:isKnow ?programmingLanguages .	
?traineeCandidate hrOnt:beParticipate ?Project .	
?Project hrOnt:codeWith ?programmingLanguages . }	
traineeCandidate	Project
Person1	.NET_Project1

### Query 4-

➔ Person1'in projelerinde kullanılan dil çeşitleri

SELECT DISTINCT ?programmingLanguages

WHERE { hrOnt:Person1 hrOnt:beParticipate ?Project .

?Project hrOnt:codeWith ?programmingLanguages . }

SPARQL query:	
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>	
PREFIX owl: <http://www.w3.org/2002/07/owl#>	
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>	
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>	
PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>	
SELECT DISTINCT ?programmingLanguages	
WHERE { hrOnt:Person1 hrOnt:beParticipate ?Project .	
?Project hrOnt:codeWith ?programmingLanguages . }	
programmingLanguages	
ASP.NET	
VisualCSharp.NET	

## Query 5-

➔ Zorunlu stajı olanlar

SELECT ?traineeCandidate

WHERE { ?traineeCandidate hrOnt:compulsory\_Candidate ?compulsory

FILTER (?compulsory = "true"^^

<http://www.w3.org/2001/XMLSchema#boolean>) . }

SPARQL query:	
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>  PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>  SELECT ?traineeCandidate WHERE { ?traineeCandidate hrOnt:compulsory_Candidate ?compulsory FILTER (?compulsory = "true"^^ <http://www.w3.org/2001/XMLSchema#boolean>) . }	
traineeCandidate	
Person1	

## Query 6-

➔ Proje dil dağılımı

SELECT ?programmingLanguage (COUNT (?programmingLanguage) AS

?howmany)

WHERE { ?Project hrOnt:codeWith ?programmingLanguage }

GROUP BY ?programmingLanguage

SPARQL query:	
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#> PREFIX foaf: <http://xmlns.com/foaf/0.1/>  SELECT ?programmingLanguage (COUNT (?programmingLanguage) AS ?howmany) WHERE { ?Project hrOnt:codeWith ?programmingLanguage } GROUP BY ?programmingLanguage	
programmingLanguage	howmany
VisualCSharp.NET	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
ASP.NET	"1"^^<http://www.w3.org/2001/XMLSchema#integer>

## Query 7:

➔ Canditatelerden en küçük yaşa sahip ve projelerde kodlamış olanın ismi ve yaşı

```
SELECT DISTINCT ?person ?name ?ages
```

```
WHERE { ?person foaf:firstName ?name;
```

```
        foaf:age ?ages ;
```

```
        hrOnt:beParticipate ?Project .
```

```
}
```

```
ORDER BY ?ages
```

```
LIMIT 1
```

SPARQL query:

```
PREFIX hrOnt: <http://www.semanticweb.org/dlsn6/ontologies/2017/4/untitled-ontology-37#>
```

```
SELECT DISTINCT ?person ?name ?ages
```

```
WHERE { ?person foaf:firstName ?name;
```

```
        foaf:age ?ages ;
```

```
        hrOnt:beParticipate ?Project .
```

```
}
```

```
ORDER BY ?ages
```

```
LIMIT 1
```

person	name	ages
Person1	"Dilsen"^^<http://www.w3.org/2000/01/rdf-schema#Literal>	"23"^^<http://www.w3.org/2001/XMLSchema#integer>

## ONTOGRAF:

Ontolojimize ait ontograf'ı Protege 5.2 ile Ontograf tab'ını kullanarak oluşturduk. Sınıflar arasındaki kesikli oklar, domain veya range'i oldukları object property'leri göstermektedir. Örnek olarak; TraineeCandidate isApply Organization veya Organization isHired Candidate şeklinde görülmektedir.

