GBIF Data Access and Database Interoperability

분산데이터 검색을 위한 통합 프로토콜

Markus DÄoring & Renato De Giovanni

2004년 9월

문서의 버전 정보

시 간	버전	저자	목적
2005-11-20	0.1	안성수 (ssahn@kisti.re.kr)	처음 번역

목차

1 요약

이 보고서는 세계생물다양성정보기구(Global Biodiversity Information Facility, GBIF)의 장려로 생물다양성데이터 검색에 사용되는 2개의 주요 프로토콜을 통합하기 위해 수행된 연구 결과이다. 현재 DiGIR와 BioCASe는 생물학적 수집물과 관찰 데이터 소유주의 분산되고 이질적인 데이터베이스의 약 4,000만 건의 레코드를 서비스하는데 사용되고 있다.

DiGIR와 BioCASe의 통합은 기존 네트워크간 더욱 큰 상호운용성을 가져와 도구개발 가속화, 새로운 데이터 제공자의 참여자극, 생물다양성데이터의 접근을 증가시킬 것이다. 이러한 노력은 생물다양성분야에서 하나의 표준 프로토콜이 사용될 수 있게 하기위한 중요한 단계로 여겨지고 있다.

이 문서는 두개의 프로토콜을 검토하고, 새로운 제안을 위해 잠재적인 소스로 이용할 수 있는 다른 사양(specifications)과 기술의 대략적인 연구와 짧은 기간에 기존의 도구로 구현할 수 있는 통합 제안을 포함한다. 마지막으로 몇몇의 추가 권고내용을 포함하고 있다.

2. 감사

보고서의 저자들은 특별히 이 연구의 여러 분야에 기여하고 참여한 모든 개인과 기관에 감사를 표시하고자 한다.

Anton GÄuntsch, Dave Vieglais, Donald Hobern, Javier de la Torre, John Wieczorek, Stan Blum - 전체 연구기간동안 끊임없는 기여와 도움을 주었다

Gregor Hagedorn, Mauro Mu~noz, Ronald Bourret, Wolfgang Lipp - 많은 제안과 활기찬 토론을 진행하였다.

세계생물다양성정보기구(GBIF) – 이 연구를 장려하고 지원하였다.

University of Kansas Natural History Museum and Biodiversity Research Center, California Academy of Sciences, and Museum of Vertebrate Zoology (MVZ) in Berkeley – DiGIR를 만들고 발전시켰다.

Botanischer Garten und Botanisches Museum Berlin-Dahlem (BGBM) – BioCASE를 만들고 발전시켰고 이 연구수행에 모든 지원을 하였다.

Centro de Ref^erencia em Informa»c~ao Ambiental (CRIA) – DiGIR 기여와 이 연구수행동안 모든 지원을 하였다.

3 개정 기록

날짜	저자	참고사항
2004년 9월 23일	Markus DÄoring & Renato De Giovanni	초기 버전
2004년 11월 16일	Renato De Giovanni	약간의 수정

4 소개

2004년 오하카(Oaxaca)에서 개최된 GBIF의 데이터접근 및 데이터 상호운용성 소위원회 모임동안, DiGIR와 BioCASe 통합의 중대성은 충분히 인식되었고 중요 우선순위 사항으로 간주되었다. 이 때 이후로 GBIF는 두개 프로토콜의 대표자들이 참여하는 연구를 장려하였다.

생물다양성 공동체(community)의 전문가와 기존의 생물다양성 네트워크의 대표자들에게 그들이 사용하는 프로토콜에 대한 의견과 제안사항을 제공할 수 있도록 연락하였다. 많은 사람들이 참여하였고 모든 연구는 모임, 전자 메시지, 와 공개된 위키 웹사이트[1]를 통해 이루어졌다.

두개의 원본 프로토콜을 검토 및 문서화하였고 이 연구는 XQuery, OGC 사양과 SOAP와 같은 기존의 다른 표준을 채택하는 것을 고려하였다.

이 연구의 결과(새로운 통합 프로토콜을 위한 완전한 제안(a complete proposal)을 포함)를 이 보고서에 문서화하였다.

[1] http://ww3.bgbm.org/protocolwiki

5 주요 프로토콜 검토

5.1 DiGIR 프로토콜

5.1.1 간단한 역사

DiGIR[2]는 당시 사용되던 Z39.50[3] 프로토콜의 대체물로 종 분석자 네트워크(the Species Analyst Network, TSA)[4]에서 구상되었다. TSA는 120개 이상 연결된 데이터 소스와 분산 자연사 수집물 및 관찰 데이터베이스에 접근제공을 목적으로 하는 연구프로젝트의 일부로 만들어졌다.

Z39.50 프로토콜은 개발자에게 가파른 학습곡선을 요구했고 네트워크 관리자가 수용하기에 어려운 점 등 프로토콜 자체가 너무 복잡하게 인식되었다. 다른 기술적인 요인으로는 당시에 개념 스키마(conceptual schemas)를 정의하기 위한 형식 언어(formal language)의 부족과 XML 및 유니코드의 제한적인 지원 을 포함할 수 있다.

DiGIR 프로토콜 사양, DarwinCore[5]로 알려진 이것의 "기본(default)" 개념 스키마, 첫번째 관련 소프트웨어 개발에 초기에 3개 기관이 관여하였다: University of Kansas Natural History Museum and Biodiversity Research Center[6], California Academy of Sciences[7], and Museum of Vertebrate Zoology[8] in Berkeley. 프로젝트의 총괄적인 전략은 다음과 같다.

- 오픈 프로토콜과 표준을 사용 (HTTP, XML 그리고 UDDI)
- 프로토콜, 소프트웨어 그리고 의미(semantics)를 명확히 분리
- 소프트웨어 설치와 가능한 데이터제공자의 설정을 쉽게 하기

소프트웨어 도구들은 "오픈 소스"모델을 따라 협업 환경[9]에서 개발되었고 그 산출물은 공개 라이센스에 따라 이용할 수 있게 하였다.

---

[2] http://digir.net
[3] http://www.loc.gov/z3950/agency/
[4] http://speciesanalyst.net
[5] http://speciesanalyst.net/docs/dwc/index.html
[6] http://www.nhm.ku.edu
[7] http://www.calacademy.org
[8] http://www.mip.berkeley.edu/mvz/
[9] http://sourceforge.net/projects/digir

TSA 데이터제공자는 점차 DiGIR로 이전하고 있고 Manis[10], HerpNet[11], FishNet[12], ORNIS[13]과 같은 분리된 주제별 네트워크(thematic network)를 만들고 있다. 세계의 다른 네트워크 또한 메인 프로토콜(speciesLink[14], OBIS[15])로 DiGIR 또는 지원 프로토콜(GBIF[16])의 하나로 채택하였다.

5.1.2 설명

DiGIR는 독립적이고 이질적인 데이터베이스의 데이터를 검색하기 위한 HTTP상의 XML기반 프로토콜이다. 네트워크 참여자의 동일한 가상 뷰(a uniform virtual view)를 이루기 위해, 각 DiGIR 자원은 자신의 데이터베이스의 필드를 사용하고 맵핑할 수 있는 하나 또는 여러 개의 개념 스키마를 필요로 한다. 개념 스키마는 연합 데이터베이스 네트워크의 참조 역할을 하는 공통 데이터 모델이다.

각 네트워크는 서로 다른 개념 스키마를 정의할 수 있어 분야에 관계없이 이 프로토콜은 어느 공동체에서든 사용될 수 있다. DiGIR에서 개념 스키마는 평면 리스트로 특정 형식을 따르고 일련의 항목(개념)을 정의하는 XML 스키마로 표현된다. 이러한 스키마의 예는 DarwinCore[17]로 최초 DiGIR 네트워크에서 생물학데이터 소스가 사용할 수 있도록 제안되었다.

최초 프로젝트에서 개발된 구성요소를 사용하는 전형적인 DiGIR 네트워크는 3개의 다른 소프트웨어를 필요로 한다. 첫번째 것은 “표현 계층(presentation layer)”으로 알려진 클라이언트 소프트웨어이다. 클라이언트 소프트웨어는 몇몇 인터페이스를 통해 사용자와 상호동작하고 다른 구성 요소인 “포탈 엔진(portal engine)”과 통신한다. 포탈 엔진은 질의 발송자이다. 이것은 수동 설정 또는 UDDI 레지스트리와 상호작용을 통해 여러 데이터 제공자의 주소를 알고 있다. 질의는 선택된 데이터 제공자에 분산되고, 이 데이터 제공자는 DiGIR 메시지를 관련 데이터베이스가 사용할 수 있는 질의 언어로 해석하고 요청된 데이터를 가져오는 책임이 있다.

모니터링 서비스, 색인 서비스, 그리고 다른 포탈위에 포탈 등을 포함하는 다른 형태의 상호작용과 아키텍쳐가 가능하다.

데이터 제공자는 3~4개의 “자원(resources)”의 데이터를 서비스할 수 있다. DiGIR 자원은 이것의 구조를 하나의 개념 스키마에 맵핑한 데이터 소스로 간주되고, 개념 스키마는 다른 스키마를 확장할 수 있다. 이러한 스키마는 이질적인 자원에 대해

[10] http://elib.cs.berkeley.edu/manis/
[11] http://herpnet.org/
[12] http://habanero.nhm.ku.edu/fishnet/
[13] http://www.specifysoftware.org/Informatics/informaticsornis/
[14] http://splink.cria.org.br/
[15] http://iobis.org/
[16] http://www.gbif.net/
[17] http://digir.net/schema/conceptual/darwin/2003/1.0/darwin2.xsd

동일한 뷰를 제공한다. DiGIR 자원은 이것의 자료구조를 하나이상의 개념 스키마에 맵핑할 수 있다고 알려져 있다. 즉, 완전히 독립적이고 단위화된 스키마(modularized schemas)가 존재할 수 있다. 예를 들어 각 네트워크는 자신의 DiGIR 자원을 다른 네트워크간 공통의 개념 스키마에 맵핑할 수 있고 또한 자신의 특정 분야에 맞는 여분의 개념 스미카에 맵핑할 수 있다.

DiGIR 프로토콜 스키마[18]의 최초 범위는 제공자들간에 교환되는 메시지를 검증하는 것이다. 자원과의 통신은 제공자만을 통해서 가능하다.

5.1.3 기술적인 세부사항

개괄적인 메시지 형태

DiGIR 메시지는 <request> 또는 <response> 루트 항목(root element)에 3개의 주요 섹션(sections)을 가지고 있다. <header> 섹션은 모든 메시지에 나타나고 메시지를 만든 소프트웨어 버전 정보, 시간기록(time stamp), 메시지가 발송된 URL 또는 IP 주소, 메시지가 전송될 URL 또는 IP 주소, 요청의 종류(type)를 포함한다. 몇몇 종류의 요청은 헤더 목적지 항목(header destination element)에 DiGIR 자원을 가리키는 추가 속성(attribute)을 가질 수 있다.

<header> 이후에 메시지의 종류에 따라 내용 섹션(content section)이 있을 수 있다. 요청 메시지는 필터 조건과 반환될 개념에 대한 사양을 가질 수 있다. 응답 메시지는 반환되는 구조화된 내용을 포함한다.

응답메시지는 발생할 수 있는 에러, 경고, 또는 질의에 부합하는 레코드의 개수 등의 간단한 몇몇 정보를 포함하는 <diagnostics> 섹션을 가진다.

DiGIR 제공자는 근접 자원(local resources)에 대한 질의 발송자로 간주될 수 있기 때문에, 제공자는 헤더에 하나 이상의 근접자원과 관련된 다수의 목적지 항목을 수용할 수 있어야 한다. 이 경우에 자원으로부터 응답은 <responseWrapper> 루트 항목에 덧붙여지고 포함된다.

다음 페이지는 3~4가지의 DiGIR 메시지 예제와 함께 모든 종류의 요청과 응답 메시지를 망라한다.

[18] http://digir.net/schema/protocol/2003/1.0/digir.xsd

요청 및 응답 타입

DiGIR 서비스에서 메시지를 교환하기 위해 사용할 수 있는 3개의 요청 및 응답 타입이 있다.

DiGIR 메타데이터 연산(metadata operation)

메타데이터 요청은 제공자의 서비스와 자원에 관한 정보를 요청하는 것이다. 이것은 인자(parameters) 없이 제공자의 URL에 접근할 때 발생하는 기본 요청이다. 메타데이터 요청은 헤더 섹션만을 포함하고 목적지 항목은 제공자의 접근점(access point)이어야 한다.

XML 메시지를 사용하는 간단한 DiGIR 메타데이터 요청:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://digir.net/schema/protocol/2003/1.0">
    <header>
        <version>1.0.0</version>
        <sendTime>2004-07-20T08:51:50-0500</sendTime>
        <source>127.0.0.1</source>
        <destination>
            http://example.net/provider/DiGIR.php
            </destination>
            <type>metadata</type>
        </header>
</request>
```

메타데이터 응답은 제공자(이름과 접근점), 서비스를 운영하는 실체(이음, 웹사이트 주소, 연락 정보, 선택사항으로 요약), 그리고 각각 이용 가능한 자원에 관한 일반적인 정보를 포함한다 자원은 또한 이름, 코드(요청할 때 자원을 가리키기 위해 사용됨)와 같은 일반적인 정보, 이용 가능한 레코드의 수, 최종으로 갱신된 레코드의 시간기록, 요약, 키워드와 레코드의 단위 등의 내용정보, 가져올 수 있는 최대 레코드 수, <like>연산에 사용될 수 있는 최소 문자 수와 같은 기술정보, 인용정보, 데이터를 가져올 때의 제약사항과 같은 기타 항목을 포함한다. 자원 메타데이터의 중요한 부분은 개념 스키마가 어떠한 것에 맵핑되었는가이다.

DiGIR 메타데이터 응답 견본(sample):

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://digir.net/schema/protocol/2003/1.0">
    <header>
        <version>$Revision: 1.96 $</version>
        <sendTime>2004-07-20T08:51:55-0500</sendTime>
        <source>http://example.net:80/provider/DiGIR.php</source>
        <destination>127.0.0.1</destination>
        <type>metadata</type>
    </header>
    <content>
        <metadata>
            <provider>
                <name>Example Biodiversity Center</name>
                <accessPoint>
                    http://example.net:80/provider/DiGIR.php
                </accessPoint>
                <implementation>$Revision: 1.96 $</implementation>
                <host>
                    <name>Some Hosting Institution</name>
                    <code>SHI</code>
                    <relatedInformation>
                        http://example.net/
                    </relatedInformation>
                    <contact type="technical">
                        <name>Person A</name>
                        <title>Support specialist</title>
                        <emailAddress>person_a@example.net</emailAddress
>
                        <phone>+ 11 22 333333</phone>
                    </contact>
                    <abstract>Default hosting institution to be used
                        in examples.
                    </abstract>
                </host>
                <resource>
                    <name>Herbarium dataset</name>
                    <code>HDS</code>
                    <relatedInformation>
                        http://example.net/herbarium
                    </relatedInformation>
```

```
                    <contact type="administrative">
                        <name>Person B</name>
                        <title>Curator</title>
                        <emailAddress>person_b@example.net</emailAddress
>
                        <phone>+ 11 22 334444</phone>
                    </contact>
                    <contact type="technical">
                        <name>Person C</name>
                        <title>Systems Analyst</title>
                        <emailAddress>person_c@example.net</emailAddress
>
                        <phone>+ 11 22 334445</phone>
                    </contact>
                    <abstract>
                        Plant specimen dataset from some region.
                    </abstract>
                    <keywords>plant, specimen</keywords>
                    <citation>Herbarium dataset DiGIR provider.
                        Retrieved on (date accessed).
                        http://example.net:80/provider/DiGIR.php
                    </citation>
                    <useRestrictions>Users may not distribute, modify,
                        transmit, reuse, repost, transfer, or use any
                        content or information from this service for
                        commercial purposes without prior written
                        permission.
                    </useRestrictions>
                    <conceptualSchema schemaLocation=
                        "http://example.net/schema/darwin2.xsd">
                        http://digir.net/schema/conceptual/darwin/2003/1.0
                    </conceptualSchema>
                    <recordIdentifier>HDS</recordIdentifier>
                    <recordBasis>specimen</recordBasis>
                    <numberOfRecords>7887</numberOfRecords>
                    <dateLastUpdated>
                        2004-05-06T13:56:00-0500
                    </dateLastUpdated>
                    <minQueryTermLength>3</minQueryTermLength>
                    <maxSearchResponseRecords>
                        100
                    </maxSearchResponseRecords>
```

```
                    <maxInventoryResponseRecords>
                        1000
                    </maxInventoryResponseRecords>
                </resource>
            </provider>
        </metadata>
    </content>
    <diagnostics>
        <diagnostic code="STATUS_INTERVAL" severity="info">
            3600
        </diagnostic>
        <diagnostic code="STATUS_DATA" severity="info">1,0,0
        </diagnostic>
    </diagnostics>
</response>
```

DiGIR 목록 연산(inventory operation)

목록 요청은 특정 개념의 유일한 값을 요청한다. 선택적으로 필터를 사용할 수 있고 맵핑된 개념 스키마의 중의 하나의 개념을 지정해야 한다. 목록 요청은 모든 유일한 값의 총계를 셀 수 있는 추가 항목을 수용할 있다.

DiGIR 목록 요청 견본:

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://digir.net/schema/protocol/2003/1.0"
        xmlns:darwin="http://digir.net/schema/conceptual/darwin/2003/1.0">
    <header>
        <version>1.0.0</version>
        <sendTime>2004-07-20T09:15:30-0500</sendTime>
        <source>127.0.0.1</source>
        <destination resource="HDS">
            http://example.net/provider/DiGIR.php
            </destination>
        <type>inventory</type>
    </header>
    <inventory>
        <filter>
            <equals>
                <darwin:Collector>Thomas, J.</darwin:Collector>
            </equals>
```

```
        </filter>
        <darwin:Genus/>
        <count>true</count>
    </inventory>
</request>
```

목록 응답은 개념의 유일한 값을 포함하는 레코드 항목을 포함한다. 각 유일한 값의 발생횟수를 나타내는 숫자(count)가 제공되고 각 유일한 값의 발생 횟수를 나타내는 총계 숫자를 요청할 경우 진단(diagnostics) 섹션에 제공된다. 프로토콜 스키마에 따르면 목록 요청에서 페이징(paging)은 가능하지 않지만 몇몇 제공자 구현 시스템은 이 특징을 제공한다.

DiGIR 목록 응답 견본:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://digir.net/schema/protocol/2003/1.0">
    <header>
        <version>$Revision: 1.96 $</version>
        <sendTime>2004-07-20T09:15:32-0500</sendTime>
        <source resource="HDS">
            http://example.net:80/provider/DiGIR.php
        </source>
        <destination>127.0.0.1</destination>
        <type>inventory</type>
    </header>
    <content xmlns:darwin=
        "http://digir.net/schema/conceptual/darwin/2003/1.0"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <record>
            <darwin:Genus count="4">Agave</darwin:Genus>
        </record>
        <record>
            <darwin:Genus count="8">Passiflora</darwin:Genus>
        </record>
        <record>
            <darwin:Genus count="26">Rubus</darwin:Genus>
        </record>
        <record>
            <darwin:Genus count="3">Solanum</darwin:Genus>
        </record>
    </content>
```

```
    <diagnostics>
        <diagnostic code="STATUS_INTERVAL" severity="info">
            3600
        </diagnostic>
        <diagnostic code="STATUS_DATA" severity="info">
            1,0,1
        </diagnostic>
        <diagnostic code="MATCH_COUNT" severity="info">
            4
        </diagnostic>
        <diagnostic code="RECORD_COUNT" severity="info">
            4
        </diagnostic>
        <diagnostic code="END_OF_RECORDS" severity="info">
            true
        </diagnostic>
    </diagnostics>
</response>
```

DiGIR 검색 연산(search operation)

검색 요청은 필수적인 <filter> 항목과 응답메시지에 어떠한 것이 반환되어야 하는가와 어떻게 레코드가 구조화 될 것인지를 지정하는 선택적인 <records> 항목을 가지고 있다. 몇몇 요청은 단지 필터 조건을 만족시키는 레코드의 총계만을 요청할 수 있기 때문에 레코드 구조는 선택적이다. 그러나 이러한 것이 필요할 때 다음 예제와 같이 전체 구조를 직접 명시할 수 있고 또는 <structure> 항목내에서 선택적인 “schemaLocation” 속성을 통해 참조하게 할 수 있다. 검색 요청은 또한 일치된 레코드의 총계를 셀 수 있도록 추가 항목을 수용할 수 있다. 페이징은 “limit”와 “start” 속성을 사용해서 가능하다.

DiGIR 검색 요청 견본:

```
<?xml version="1.0" encoding="UTF-8"?>
<request
    xmlns=http://digir.net/schema/protocol/2003/1.0
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:darwin="http://digir.net/schema/conceptual/darwin/2003/1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <header>
        <version>1.0.0</version>
        <sendTime>2004-07-20T09:56:00-0500</sendTime>
        <source>127.0.0.1</source>
```

```
        <destination resource="HDS">
            http://example.net/provider/DiGIR.php
        </destination>
        <type>search</type>
    </header>
    <search>
        <filter>
            <and>
                <equals>
                    <darwin:Collector>Thomas, J.</darwin:Collector>
                </equals>
                <equals>
                    <darwin:Genus>Rubus</darwin:Genus>
                </equals>
            </and>
        </filter>
        <records limit="3" start="0">
            <structure>
                <xsd:element name="record"
                             minOccurs="0"
                             maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element ref="darwin:InstitutionCode"/>
                            <xsd:element ref="darwin:CollectionCode"/>
                            <xsd:element ref="darwin:CatalogNumber"/>
                            <xsd:element ref="darwin:ScientificName"/>
                            <xsd:element ref="darwin:Latitude"/>
                            <xsd:element ref="darwin:Longitude"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </structure>
        </records>
        <count>true</count>
    </search>
</request>
```

검색 응답은 질의와 일치한 요청된 레코드를 포함한다. 이것은 주어진 레코드 구조에 맞는 형식을 갖는다.

DiGIR 검색 응답 견본:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://digir.net/schema/protocol/2003/1.0">
    <header>
        <version>$Revision: 1.96 $</version>
        <sendTime>2004-07-20T09:56:03-0500</sendTime>
        <source resource="HDS">
            http://example.net:80/provider/DiGIR.php
        </source>
        <destination>127.0.0.1</destination>
        <type>search</type>
    </header>
    <content xmlns:darwin=
        http://digir.net/schema/conceptual/darwin/2003/1.0
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <record>
            <darwin:InstitutionCode>INS</darwin:InstitutionCode>
            <darwin:CollectionCode>COL</darwin:CollectionCode>
            <darwin:CatalogNumber>27694</darwin:CatalogNumber>
            <darwin:ScientificName>
                Rubus brasiliensis
            </darwin:ScientificName>
            <darwin:Latitude>-23.6625</darwin:Latitude>
            <darwin:Longitude>-46.7738</darwin:Longitude>
        </record>
        <record>
            <darwin:InstitutionCode>INS</darwin:InstitutionCode>
            <darwin:CollectionCode>COL</darwin:CollectionCode>
            <darwin:CatalogNumber>27907</darwin:CatalogNumber>
            <darwin:ScientificName>
                Rubus brasiliensis
            </darwin:ScientificName>
            <darwin:Latitude>-23.412</darwin:Latitude>
            <darwin:Longitude>-46.7899</darwin:Longitude>
        </record>
        <record>
            <darwin:InstitutionCode>INS</darwin:InstitutionCode>
            <darwin:CollectionCode>COL</darwin:CollectionCode>
            <darwin:CatalogNumber>27908</darwin:CatalogNumber>
            <darwin:ScientificName>
                Rubus urticaefolius
```

```
            </darwin:ScientificName>
            <darwin:Latitude>-23.412</darwin:Latitude>
            <darwin:Longitude>-46.7899</darwin:Longitude>
        </record>
    </content>
    <diagnostics>
        <diagnostic code="STATUS_INTERVAL" severity="info">
            3600
        </diagnostic>
        <diagnostic code="STATUS_DATA" severity="info">
            1,1,1
        </diagnostic>
        <diagnostic code="MATCH_COUNT" severity="info">
            26
        </diagnostic>
        <diagnostic code="RECORD_COUNT" severity="info">
            3
        </diagnostic>
        <diagnostic code="END_OF_RECORDS" severity="info">
            false
        </diagnostic>
    </diagnostics>
</response>
```

레코드 구조에 NULL 값 또는 맵핑되지 않은 개념이 있을 때 응답에서 해당되는 항목의 속성 “xsi:nil”은 “true” 값이 세팅된다.

DiGIR 필터 인코딩 (filter encoding)

DiGIR 프로토콜은 필터 항목을 통해 정의되는 자체적이고 일반적인 질의 체계를 가지고 있고, 데이터 지역적으로 어떻게 저장되고(관계형 데이터베이스, XML 파일, 객체 데이터베이스 등) 또는 특정한 질의 언어(SQL, XPath 등)에 관해 어떠한 것도 가정하지 않는다. DiGIR 질의와 근접 데이터베이스 질의 언어간의 해석 작업은 완전히 DiGIR 제공자 구현 시스템에 위임된다.

DiGIR 필터는 직접적으로 아래와 같은 하나의 비교 표현문을 포함할 수 있다.

```
<filter>
    <equals>
        <darwin:ScientificName>
            Rubus brasiliensis
```

```
        </darwin:ScientificName>
    </equals>
</filter>
```

비교 연산자는 <equals>, <notEquals>, <lessThan>, <lessThanOrEquals>, <greaterThan>, <greaterThanOrEquals>와 <like>를 포함한다. 이 모든 것은 개념과 하나의 값의 간단한 비교를 표현하기 위해 사용된다.

NULL 값과의 비교는 이미 XML Schema 정의 언어의 일부인 “xsi:nil” 속성을 사용해서 수행될 수 있다:

```
<filter>
    <equals>
        <darwin:Latitude xsi:nil="true"/>
    </equals>
</filter>
```

또 다른 특별 비교 연산자인 <in>은 하나의 개념과 일련의 값의 비교를 표현하기 위해 사용될 수 있다.

```
<filter>
    <in>
        <list>
            <darwin:Genus>Physalis</darwin:Genus>
            <darwin:Genus>Rubus</darwin:Genus>
            <darwin:Genus>Byrsonima</darwin:Genus>
        </list>
    </in>
</filter>
```

비교 표현문은 논리 연산자를 통해 결합될 수 잇다. 이용 가능한 4개의 논리 연산자가 있다: <and>, <andNot>, <or>, <orNot>. 이것 모두는 이진 연산자로 정의되기 때문에, 2개 이상의 비교가 관계될 경우에 논리연산자를 구조에 맞게 끼워 넣는 것이 필요하다.

```
<filter>
    <and>
        <equals>
            <darwin:ScientificName>
                Rubus brasiliensis
            </darwin:ScientificName>
        </equals>
```

```
        <or>
            <greaterThan>
                <darwin:YearIdentified>1980</darwin:YearIdentified>
            </greaterThan>
            <like>
                <darwin:IdentifiedBy>Koch%</darwin:IdentifiedBy>
            </like>
        </or>
    </and>
</filter>
```

다른 개념 스키마의 개념을 같은 필터안에서 결합하여 사용할 수 있는 것은 흥미롭다. 자신의 스키마를 구분하기 위해 개념 항목내에 이름공간(namespace) 접두사가 항상 사용된다.

DiGIR 레코드 구조

DiGIR 검색 요청은 어떠한 내용 항목이 반환되고 어떻게 구조화되어야 하는지를 지정할 수 있다. 이것은 레코드 구조 사양을 통해 이루어진다. DiGIR 레코드 구조는 XML 스키마 정의의 간단한 부분집합을 이용하는데 내용 항목을 XML “ref” 속성으로 참조한다.

```
<structure>
    <xsd:element name="record" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="darwin:InstitutionCode"/>
                <xsd:element ref="darwin:CollectionCode"/>
                <xsd:element ref="darwin:CatalogNumber"/>
                <xsd:element ref="darwin:ScientificName"/>
                <xsd:element ref="darwin:Latitude"/>
                <xsd:element ref="darwin:Longitude"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</structure>
```

레코드 구조의 문법(syntax)은 DiGIR 프로토콜 스키마에 명시되어 있지 않다. 레코드 항목의 정의는 필요하지만 <structure> 항목은 여기에서 실제로 어느 내용이든지 수용할 수 있는 XML 복합타입으로 정의된다.

<record> 항목은 레코드의 반복(looping), 페이징, 제한하기를 할 때의 기본 수단이다. 관계형 데이터베이스에서 보통의 접근방법은 자원 설정동안 “root table”을 가지고, 이것의 레코드를 <record>항목에 연결하는 것이다.

내용항목이 참조되기 때문에 항목의 개수는 개념 스키마의 관계된 항목정의에서 가져와야 한다. 레코드 구조를 사용하는 다른 가능한 방법은 다음 예에서 보여주는 것처럼 특정 항목의 그룹화를 요청하는 것이다.

```
<structure>
    <xsd:element name="record" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="darwin:InstitutionCode"/>
                <xsd:element ref="darwin:CollectionCode"/>
                <xsd:element ref="darwin:CatalogNumber"/>
                <xsd:element ref="darwin:ScientificName"/>
                <xsd:element name="coordinates">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element ref="darwin:Latitude"/>
                            <xsd:element ref="darwin:Longitude"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</structure>
```

마지막으로, 똑 같은 레코드의 구조를 “schemaLocation” 속성을 사용해 원격으로 참조할 수 있다.

```
<structure schemaLocation=
            "http://example.net/searchStructure.xml"/>
```

레코드 구조가 정의되고 사용되는 방법과 관련한 중요한 점 하나는 결과가 항상 맞춤 그룹 항목 또는 개념 항목의 인스탄스여야 하는 것이다. 항목이름을 변경하고 (XML “ref” 속성을 사용하는 항목은 참조되는 항목의 이름을 무시할 수 없다[19]) 값을 XML 속성으로 반환하는 것(속성 선언은 항목을 참고할 수 없고, 이것은 전역

[19] http://www.w3.org/TR/xmlschema-1/#section-Constraints-on-XML-Representations-of-Element-Declarations

속성 선언만 참조할 수 있다[20])이 가능하지 않다. 레코드 구조가 데이터가 어떻게 반환될지 에 대한 유동성을 제공하지만, 프로토콜에 독립적인 내용 항목을 표현하는 외부 XML Schema에 대해 검증할 수 있는 결과를 생산할 만큼 유동적이지는 않다.

단위화된 개념 스키마의 사용 관련하여, 레코드 구조의 항목은 같은 구조내의 다른 스키마의 개념을 완벽하게 참조할 수 있다. 개념 스키마는 항상 "ref" 속성내의 이름공간 접두사와 연관된다.

DiGIR 개념 바인딩(conceptual binding)

이전 예제에서 본 것처럼, 개념 스키마의 내용 항목은 메시지에서 직접 사용(필터 비교 표현문과 목록 요청에서 발생)되거나 참조(레코드 구조 사양에서 발생)된다.

개념 스키마의 각 내용 항목이 DiGIR 프로토콜 스키마에서 정의되는 <searchableData>, <returnableData> 또는 <searchableReturnableData> 등의 3개 추상 항목 중 하나에서 파생되어야 하는 것은 프로토콜 설계 결정이었다. 그러므로, 하나의 개념이 DarwinCore의 "BoundingBox" 항목과 같이 반환가능(returnable)은 아니지만 검색가능(searchable)이 될 수 있다. 또한 검색가능하지 않지만 반환가능 항목이 있을 수 있다(이진 데이터 타입의 경우).

이러한 접근법은 필터 표현문이 접근가능 개념만을 사용할 수 있고, 목록 연산이 반환가능 개념의 값만을 요청할 수 있게 보장한다. 내용 항목이 반환가능 개념만을 가리켜야 한다는 점에서 레코드 구조는 이론적으로 비슷한 제약을 가져야 한다. 그러나 이전에 언급한 것처럼 프로토콜을 이것을 검증하지 않는다.

원래의 접근방법은 또한 데이터 타입에 따라 개념을 분류하기 위해 다른 상속 계층의 추가 가능성을 고려하였지만 XML Schema가 다중 상속을 허용하지 않기 때문에 이것은 부가적인 복잡성을 가져왔을 것이다. 이 경우에 개념이 "alphaSearchableData", 또는 "numericSearchableData" 등등에서 유래되어야 할 것이다. 이것은 스트링(string) 연산이 검색가능 스트링 개념에 대해서 수행될 수 있는 것을 보장할 것이다. 그렇지만 이 접근방법은 채택되지 않았다.

개념이 프로토콜에서 사용되는 방식은 개념 스키마에 2가지 결과를 초래하였다. 첫 번째 것은 모든 개념이 검색가능 그리고/또는 반환가능으로 분류될 필요가 있어 이것들이 명확히 프로토콜에 바인딩되어야 하는 것이었다. 바인딩은 XML Schema "substitutionGroup" 기술로 이루어지고, 따라서 완전히 독립적인 개념 스키마를 사용하는 것은 가능하지 않다.

[20] http://www.w3.org/TR/xmlschema-1/#Attribute Declaration details

다른 결과는 “substitutionGroup” 기술은 “헤드(head)” 항목(이 경우 추상 개념 타입)과 멤버 항목(“실제(real)” 개념 항목)이 스키마의 전역 범위에서 선언되어야만 하는 것이 요구되었다[21].

DiGIR가 결과 메시지에서 맵핑되지 않은 개념과 NULL 값을 다루는 방식 때문에, 개념 스키마의 모든 의무적이지 않은 항목들은 “nillable”로 선언될 필요가 있다.

그래서 DiGIR 호환 개념 스키마는 프로토콜이 정의하는 3개의 추상 데이터 항목중의 하나에서 유래하는 항상 평면적인 리스트의 항목들로 구성된다. 이 접근방법이 약간의 제약성을 초래하지만, 이것은 또한 프로토콜의 개념 사용과 관련하여 부가적인 검증을 가능하게 한다.

DiGIR 연산 호출

불행히도 DiGIR를 사용하여 어떻게 메시지가 교환되어야 하는 것에 대한 공식적인 사양은 없다. 참조 가능한 것으로, 원래의 제공자 구현 시스템은 3~4가지 방식으로 요청을 수용한다.

- DiGIR XML 요청 문서 또는 요청 문서를 가리키는 URL을 포함하는 “request” 또는 “doc” 으로 불리는 하나의 GET 또는 POST 인자
- 요청 문서를 가리키는 URL을 포함한 하나의 GET 또는 POST 기본 인자
- 요청 문서의 일부를 포함하는 일련의 GET 또는 POST 인자: “filter”, “resource”, “startrec”, “maxrecs”, “clientkey”, “recordstruct”, “sortstruct”, 그리고 “countrecs”

5.1.4 현존하는 DiGIR 소프트웨어

현재, DiGIR 프로토콜을 지원하는 3~4개의 소프트웨어가 이미 존재한다. 최초 프로젝트의 부분으로 개발되어 구현 시스템이 가장 널리 쓰이고 있다. 이것들은 모두 sourcefore 사이트의 “digir” 프로젝트에서 무료로 이용 가능하다.

- DiGIR PHP 제공자: DiGIR 네트워크에 근접 데이터 소스를 연결하는 책임이 있는 구성 요소. PHP로 구현되었고 웹 서버와 같이 동작한다. 안정되고 완전한 기능을 한다고 여겨지지만, 동시에 하나 이상의 자원에 전송되는 요청은 지원하지 않는다. 근접 데이터 소스는 관계형 데이터베이스로 제한된다. 대부분의 회사에 대한 드라이버가 존재한다. Sourceforge 사이트에서 이용 가능한 파일 이외에, 각기 다른 플랫폼의 웹 서버와 UDDI 등록 기능을 가진 패키지를 GBIF 사이트[22]에서 구할 수 있다. PHP 제공자는 또한 Specify[23] 소프트웨어의 일부로 배포

[21] http://www.w3.org/TR/xmlschema-1/#Element Equivalence Class
[22] http://www.gbif.org/

되고 있다.

- DiGIR 포탈 엔진: 분산된 데이터 제공자의 DiGIR 네트워크를 추상화하는 책임을 가진 구성요소. 사용자 인터페이스는 검색 에이전트에서 요청을 받아 이것을 관계된 데이터 제공자에게 배분하고 요청자에게 다시 보내기 전에 하나의 문서에 응답을 통합한다. Java로 구현되었고 Tomcat과 동작한다. 제공자 접근점은 수동으로 설정하거나 UDDI 레지스트리에서 얻을 수 있다.
- DiGIR 표현 계층: DiGIR 네트워크를 질의하기 위해 사용자 인터페이스를 제공하는 책임이 있는 구성요소. 이것은 포탈 엔진과 통신한다. Java로 구현되었고 다른 Tomcat 인스탄스와 동작한다.

최초의 제공자 소프트웨어 이외에 2개의 다른 제공자 구현 시스템이 알려져 있다.

- GBIF 데이터저장소 도구(data repository tool)[24]: Python으로 구현한 DiGIR 제공자를 가진 Zope와 미리 설정된 테이블을 가진 MySQL를 포함하는 패키지
- Biota[25] 제공자: Java 구현되고 현재 테스트중인 제공자. 이것은 차기 Biota 배포판의 일부가 될 것이다.

최초의 포탈 엔진이외에 PHP를 사용한, PHP DiGIR 포탈[26]로 알려진 다른 포탈 구현 시스템이 있다. 이것은 사용자 인터페이스를 포함하고 최초 PHP 제공자 구현시스템에서 이용 가능한 소켓 기반 스크립트를 사용하여 몇몇 제공자와 통신한다.

그리고 마지막으로, DiGIR와 관련된 몇몇 추가적인 라이브러리(libraries)가 있다.

- GBIF 포탈 라이브러리[27]: GBIF 데이터 포탈은 제공자와 직접 통신하기 위해 Java 라이브러리를 사용한다.
- Perl 클라이언트 라이브러리: speciesLink 검색 인터페이스[28]는 포탈 엔진과 통신하기 위해 perl 라이브러리를 사용한다. 메타데이터와 검색 메시지만을 다룰 수 있다.
- 글로버스 DiGIR 랩퍼(Globus DiGIR wrapper): SEEK 프로젝트[29]는 또한 글로버스 툴킷[30](Globus toolkit) 위에 포탈 엔진과 통신하기 위한 웹 서비스를 개발하였다.

---

[23] http://www.specifysoftware.org/
[24] http://circa.gbif.net/Public/irc/gbif/ict/library?l=/download gbif tools/gbif repository
[25] http://viceroy.eeb.uconn.edu/biota
[26] http://digirportal.berkeley.edu/
[27] http://sourceforge.net/projects/gbif
[28] http://splink.cria.org.br/simple search
[29] http://seek.ecoinformatics.org
[30] http://www.globus.org/toolkit/

5.2 BioCASe 프로토콜

5.2.1 간략한 역사

BioCASe 프로토콜[31]은 베를린의 식물원 및 식물학 박물관[32](Botanical Garden Botanical Museum)의 생물다양성정보학 부서에서 BioCASE 프로젝트를 위해 DiGIR 프로젝트[33]와 비슷한 목적을 가지고 개발되었다. 그러나 DarwinCore를 기본 개념 스키마로 사용하는 대신에 BioCASE는 생물학적 표본 또는 관찰 데이터 교환을 위해 다소 복잡하고 계층적으로 구조화된 XML 표준인 ABCD[34]를 사용하여 분산되고 이질적인 네트워크 구현을 목표로 하였다. 초기에 이것은 DiGIR 프로토콜을 사용하려고 의도하였으나 XML 대체 그룹(substitution groups)을 사용하는 DiGIR의 개념 바인딩이 아래와 같은 사항을 개념 스키마에 요구하였기 때문에 조만간 DiGIR는 이러한 필요에 적합하지 않다는 것을 발견하였다.

- 개념 스키마에 프로토콜의 특정 부분을 삽입
- 전역 항목 선언의 평면 리스트

가능한 유일한 해결책은 계층적인 항목 리스트를 전역항목으로 포함하는 수정된 ABCD 스키마를 채택하는 것이었을 것이다. 2002년 Indaituba의 TDWG 회의에서 약 2천 개의 전역 항목 리스트가 제시되었고 이것은 수용할 수 없는 것으로 간주되었다. BioCASE 관점에서, 프로토콜은 어떠한 중요한 방법으로든 개념 스키마에 영향을 주지 않아야 한다.

DiGIR 프로토콜을 기초로 하여 개념 스키마의 항목을 참조하기 위하여 간단한 XPaths[35]를 사용하고 서로 다른 개념 바인딩을 가진 새로운 프로토콜이 개발되었다. 이것으로 모든 계층 스키마를 사용하는 것이 가능하였지만 개념 스키마에 대해 강력한 XML 기반 검증을 하지 못하는 단점이 있었다. 추가적으로 가장 두드러지고 새로운 역량(capabilities) 요청과 함께 몇몇 사소한 변화가 프로토콜에 가해졌다. 모든 소프트웨어는 모질라 공개 라이센스(Mozilla public license)를 통해 공개적으로 이용 가능하였다.

[31] http://www.biocase.org/dev/protocol
[32] http://www.bgbm.org/BioDivInf
[33] http://www.biocase.org
[34] http://www.bgbm.org/TDWG/CODATA/Schema/
[35] http://www.w3.org/TR/xpath

5.2.2 설명

DiGIR 프로토콜을 기저로, BioCASe[36] 또한 독립적이고 이질적인 데이터베이스에서 데이터를 가져오기 위한 XML 기반 프로토콜이다. 네트워크 참여자의 동일한 가상 뷰를 이루기 위해, 각 BioCASe 데이터 소스는 하나 또는 여러 개의 개념 스키마(즉 데이터 교환 표준)를 선택하는 것이 필요하고 다음에 근접 데이터베이스의 몇 가지 필드를 이러한 스키마의 (몇 가지)항목에 맵핑해야 한다.

서로 다른 개념 스키마가 각 네트워크에 따라 정의될 수 있고, 따라서 어느 공동체든지 분야에 관계없이 프로토콜을 사할 수 있다. 개념 스키마는 프로토콜과 관계된 추가 항목을 필요로 하지 않고, 재귀 구조를 제외하고 선택(choices)과 반복(repeating) 항목을 포함하는 모든 XML 스키마 구조를 지원한다. 각 XML 항목 또는 속성은 XPath 기반 개념 바인딩으로 참조할 수 있는 개념으로 간주된다. 현재 사용되고 있는 이러한 스키마의 2가지 예는 생물학적 수집물 데이터 접근(Access to Biological Collection Data) 표준 [37] 과 수집물을 서술하는 메타프러파일(metaprofile)[38]이 있다.

BioCASE 네트워크는 중앙 레지스트리, 유닛로더(Unitloader) [39] 라 불리는 메시지 브로커 서비스(message broker service) 그리고 BioCASE 제공자 소프트웨어[40]를 운영하는 제공자로 구성된다.

유닛로더 소프트웨어는 프로토콜 문서를 생성하고 쓰레드(threads)를 통해 요청된 일련의 “데이터소스” 서비스에 이것을 배포하는 Java 클래스이다. 이것은 레지스트리와 직접 연결되어있지 않지만 질의할 원하는 제공자 리스트를 필요로 한다.

제공자는 BioCASE 제공자 소프트웨어를 운영하는 주체(host)로 간주된다. 이것은 근접 데이터소스를 질의, 그래픽적으로 데이터소스[41]를 설정, 데이터소스 서비스로 역할을 하는 많은 랩퍼를 위한 소프트웨어 도구 모음이다. 각각의 데이터소스는 하나의 데이터베이스로 제약되지만 수용되는 어느 개수의 개념 스키마에 맞게 설정될 수 있다. 이러한 방식으로 예를 들어 ABCD와 DarwCore를 같이 사용할 수 있다. DiGIR와는 반대로 모든 데이터소스를 접근을 주는 제공자 서비스는 없지만 대신 각각의 개별 데이터소스는 자신의 접근점 URL을 가진다.

[36] http://www.bgbm.org/biodivinf/Schema/protocol_1_3.xsd
[37] http://www.bgbm.org/TDWG/CODATA/Schema/ABCD-1.20.xsd
[38] http://www.bgbm.org/biodivinf/Schema/BioCASE-MetaProfile-123.xsd
[39] http://www.biocase.org/dev/unitloader
[40] http://www.biocase.org/dev/provider/
[41] http://www.biocase.org/dev/configtool

5.2.3 기술적 세부사항

개괄적인 메시지 형태

BioCASe 메시지는 요청 또는 응답이다 – 두 개 모두 3개의 주요 부분으로 구성된다.

“header” 섹션은 시간기록과 메시지 흐름 체인에 관여한 모든 서비스 리스트와 같은 메시지의 출발지(origin)와 목적지(destination)에 관한 몇몇 정보를 제공한다. 메시지를 생성과 관계된 소프트웨어 버전이 명시된다.

“content” 섹션은 모든 응답 그리고 검색 및 스캔(scan) 요청에만 존재한다. 이것은 개념 스키마에 지정된 형식의 요청된 응답 데이터를 가지거나 데이터소스가 수행할 요청을 지정한다. 이것은 또한 페이징을 위한 상태 정보, 데이터베이스에서 일치된 전체 레코드와 얼마나 많은 레코드가 반환되었고 또는 “버려졌는지(dropped)”와 같은 레코드 숫자 등을 가지는 속성을 포함한다. 이것은 누락 또는 잘못된 필수 데이터 로 인해 개념 스키마에 대해 검증되지는 않았지만 질의와 일치한 레코드의 개수이다.

“diagnostics” 섹션은 디버깅 정보와 경고 또는 에러 메시지를 제공하기 위해 사용된다.

요청 및 응답 타입

DiGIR와 비슷하게 3개의 다른 메시지 타입이 제공되고 있다. 그러나 메타데이터 연산 대신 약간 다른 역량(capabilities) 연산이 있다. DiGIR 목록 타입은 “스캔(scan)”이다(BioCASe 프로토콜이 만들어졌을 당시 초기 제안서에서 불렸던 것처럼).

BioCASe 역량 연산(capabilities operation)

역량 요청은 클라이언트가 어떠한 개념이 제공자 데이터베이스에 맵핑(정의)되었는지에 관한 정보를 제공한다. 이 요청 타입은 모든 맵핑된 개념을 알 수 있는 xpaths 리스트를 반환한다. 역량 요청에는 아무런 인자가 관여하지 않고 연산이 지정되지 않을 경우 이것은 또한 제공자 소프트웨어의 기본 응답이다.

BioCASe 역량 요청 견본:

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://www.biocase.org/schemas/protocol/1.3">
```

```
    <header>
        <version software="unitloader">0.98</version>
        <sendTime>2003-09-25T17:02:45+02:00</sendTime>
        <source>198.14.7.54</source>
        <source>192.168.1.153</source>
        <type>capabilities</type>
    </header>
</request>
```

BioCASe 역량 응답 견본:

```
<?xml version="1.0" encoding="UTF-8"?>
<response xmlns="http://www.biocase.org/schemas/protocol/1.3">
    <header>
        <version software="Python Interpreter">
          2.3 (₩#46, Jul 29 2003, 18:54:32)
          [MSC v.1200 32 bit (Intel)]
        </version>
        <version software="Wrapper">1.4.0 alpha</version>
        <version software="OS">nt</version>
        <sendTime>2003-09-25T17:02:46+02:00</sendTime>
        <source>192.168.1.12</source>
        <destination>192.168.1.153</destination>
        <destination>198.14.7.54</destination>
        <type>capabilities</type>
    </header>
    <content>
        <capabilities>
            <SupportedSchemas request="true"
              namespace="http://www.tdwg.org/schemas/abcd/1.2"
              response="true">
                <Concept>
                  /DataSets/DataSet/DatasetDerivations/
                  DatasetDerivation/DateSupplied
                </Concept>
                <Concept>
                  /DataSets/DataSet/DatasetDerivations/
                  DatasetDerivation/Description
                </Concept>
                <Concept>
                  /DataSets/DataSet/DatasetDerivations/
                  DatasetDerivation/Rights/CopyrighDeclaration
```

</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Rights/IPRDeclaration
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Rights/LegalOwner/URLs/URL
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Rights/RightsURL
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Rights/SpecificRestrictions
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Rights/TermsOfUse
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Statements/Acknowledgement
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Statements/Disclaimer
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Statements/LogoURL
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Statements/StatementURL
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Supplier/Addresses/Address
</Concept>
<Concept>

/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Supplier/Person/PersonName
</Concept>
<Concept>
/DataSets/DataSet/DatasetDerivations/
DatasetDerivation/Supplier/URLs/URL
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceExpiryDate
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceInstitutionCode
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceLastUpdatedDate
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceName
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceNumberOfRecords
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceVersion
</Concept>
<Concept>
/DataSets/DataSet/OriginalSource/
SourceWebAddress
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/
CollectorsFieldNumber
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/
GatheringAgents/GatheringAgent/Person/PersonName

</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/
GatheringDateTime/DateText
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/
GatheringDateTime/ISODateTimeBegin
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Altitude/MeasurementAtomized/MeasurementLowerValue
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Altitude/MeasurementAtomized/MeasurementScale
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Aspect/AspectText
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
BiotopeData/BiotopeText
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Country/CountryName
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Country/ISO2Letter
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
LocalityText
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
NamedAreas/NamedArea/NamedAreaName
</Concept>
<Concept>

/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Slope/MeasurementAtomized/MeasurementLowerValue
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/GatheringSite/
Slope/MeasurementAtomized/MeasurementScale
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Gathering/
Project/ProjectTitle
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
Identification/Identifier/IdentifierPersonName
/PersonName
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
Identification/TaxonIdentified/AuthorString
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
Identification/TaxonIdentified/HigherTaxa/HigherTaxon
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
Identification/TaxonIdentified/NameAuthorYearString
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
Identification/TaxonIdentified/ScientificNameAtomized/
Botanical/FirstEpithet
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
/Identification/TaxonIdentified/ScientificNameAtomized/
Botanical/Genus
</Concept>
<Concept>
/DataSets/DataSet/Units/Unit/Identifications/
Identification/TaxonIdentified/ScientificNameAtomized/
Botanical/Rank</Concept>

```
                <Concept>
                  /DataSets/DataSet/Units/Unit/Identifications/
                  Identification/TaxonIdentified/ScientificNameAtomized/
                  Botanical/SecondEpithet
                </Concept>
                <Concept>
                  /DataSets/DataSet/Units/Unit/UnitID
                </Concept>
                <Concept>
                  /DataSets/DataSet/Units/Unit/UnitStateDomain/
                  SpecimenUnit/UnitPreparation/PreparationType
                </Concept>
            </SupportedSchemas>
            <SupportedSchemas request="true"
                namespace="http://www.namespacetbd.org/darwin2"
                response="true">
                <Concept>/RecordSet/Record/CatalogNumber</Concept>
                <Concept>/RecordSet/Record/CollectionCode</Concept>
                <Concept>/RecordSet/Record/Collector</Concept>
                <Concept>/RecordSet/Record/Country</Concept>
                <Concept>/RecordSet/Record/DateLastModified</Concept>
                <Concept>/RecordSet/Record/Family</Concept>
                <Concept>/RecordSet/Record/FieldNumber</Concept>
                <Concept>/RecordSet/Record/Genus</Concept>
                <Concept>/RecordSet/Record/IdentifiedBy</Concept>
                <Concept>/RecordSet/Record/InstitutionCode</Concept>
                <Concept>/RecordSet/Record/ScientificName</Concept>
                <Concept>/RecordSet/Record/Subspecies</Concept>
            </SupportedSchemas>
        </capabilities>
    </content>
    <diagnostics>
        <diagnostic>OK</diagnostic>
    </diagnostics>
</response>
```

이전 예제는 <source> 항목이 2번 나타나는 것을 보여주고 있는데 첫번째 source 항목은 시작점을 나타낸다. 응답 메시지는 DarwinCore와 ABCD 2개의 설정된 개념 스키마에 대해 맵핑된 개념을 나열한다. 이 스키마는 이름공간으로 식별된다.

BioCASe 스캔 연산(scan operation)

스캔 요청은 각 개념 스키마의 항목 중 xpath로 참조되는 하나의 개념에 집중한다. 이것은 본질적으로 SQL 의 select distinct이고 이 개념에 대해 유일한 값을 반환한다. DiGIR 목록과 대조적으로 스캔에 필터를 지정할 수 없다.

다음은 식물 속(genera)의 유일한 값을 구하는 간단히 요약된 스캔 요청 예제이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://www.biocase.org/schemas/protocol/1.3">
    <header>
        <sendTime>2003-09-25T17:02:45+02:00</sendTime>
        <source>198.14.7.54</source>
        <type>scan</type>
    </header>
    <scan>
        <requestFormat>
          http://www.tdwg.org/schemas/abcd/1.2
        </requestFormat>
        <concept>
          /DataSets/DataSet/Units/Unit/Identifications/
          Identification/TaxonIdentified/ScientificNameAtomized/
          Botanical/Genus
        </concept>
    </scan>
</request>
```

스캔 응답은 각각 정렬된 유일한 값을 <value> 항목내에 전달하고 “recordCount” 내용 속성안에 값의 총계를 제공한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<response
    xmlns=http://www.biocase.org/schemas/protocol/1.3
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.biocase.org/schemas/protocol/1.3
    http://www.bgbm.org/biodivinf/schema/protocol_1_3.xsd">
    <header>
        <version software="Python Interpreter">
          2.3 (₩#46, Jul 29 2003, 18:54:32)
          [MSC v.1200 32 bit (Intel)]
        </version>
        <version software="PyWrapper">0.98a</version>
        <version software="DB module">
          MS SQL Server module v0.91, using mxODBC 2.0.1
```

```
        </version>
        <version software="OS">nt</version>
        <sendTime>2003-09-25T17:02:46+02:00</sendTime>
        <source>192.168.1.12</source>
        <destination>198.14.7.54</destination>
        <type>scan</type>
    </header>
    <content recordDropped="0"
             recordStart="0"
             recordCount="188">
        <scan>
            <value>Acantholimon</value>
            <value>Achillea</value>
            <value>Aethionema</value>
            <value>Ajuga</value>
            <value>Alchemilla</value>
            <value>Alkanna</value>
            <value>Allium</value>
            <value>Alopecurus</value>
            <value>Alyssum</value>
            ...
            <value>Viola</value>
            <value>Xeranthemum</value>
            <value>Ziziphora</value>
        </scan>
    </content>
    <diagnostics>
        <diagnostic>OK</diagnostic>
    </diagnostics>
</response>
```

BioCASe 검색 연산(search operation)

BioCASe 프로토콜은 응답 할 때 (“responseFormat”) 사용될 개념 스키마뿐만 아니라 검색 요청에서 필터가 지정되도록 요구한다. “requestFormat” 이름공간에 의해 상세화되는 질의 필터를 만들기 위해 응답 스키마가 아닌 다른 스키마의 개념을 사용하는 것이 가능하다. 다른 어떤 데이터를 반환하지 않고 단지 일치되는 레코드의 개수를 세기 위해서는 선택 항목인 “count”가 참(true)으로 세팅되어야 한다. 검색은 상태보존(stateful)이다. 페이징은 기본 속성이 0인 “start”와 “limit” 속성을 사용해서 지원된다.

BioCASe 검색 요청 견본:

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://www.biocase.org/schemas/protocol/1.3">
    <header>
        <version software="unitloader">0.98</version>
        <sendTime>2003-09-25T17:02:45+02:00</sendTime>
        <source>198.14.7.54</source>
        <source>192.168.1.153</source>
        <type>search</type>
    </header>
    <search>
        <requestFormat>
          http://www.tdwg.org/schemas/abcd/1.2
        </requestFormat>
        <responseFormat start="0" limit="2">
          http://www.tdwg.org/schemas/abcd/1.2
        </responseFormat>
        <filter>
            <like path="/DataSets/DataSet/Units/Unit/
              /Identifications/Identification/TaxonIdentified/
              NameAuthorYearString">
                Ast*
            </like>
        </filter>
        <count>false</count>
    </search>
</request>
```

ABCD 스키마를 사용하는 응답은 다음과 같다:

```
<?xml version="1.0" encoding="latin1"?>
<response xmlns="http://www.biocase.org/schemas/protocol/1.3">
    <header>
        <version software="Python Interpreter">
          2.3 (₩#46, Jul 29 2003, 18:54:32)
          [MSC v.1200 32 bit (Intel)]
        </version>
        <version software="Wrapper">1.4.0 alpha</version>
        <version software="DB module">
          MS SQL Server module v0.91, using mxODBC 2.0.1
        </version>
        <version software="OS">nt</version>
```

```
        <sendTime>2003-09-25T17:02:47+02:00</sendTime>
        <source>192.168.1.12</source>
        <destination>192.168.1.153</destination>
        <destination>198.14.7.54</destination>
        <type>search</type>
    </header>
    <content recordDropped="0"
             recordCount="2"
             recordStart="0"
             totalSearchHits="122">
        <DataSets xmlns="http://www.tdwg.org/schemas/abcd/1.2">
            <DataSet>
                <OriginalSource>
                    <SourceInstitutionCode>B</SourceInstitutionCode>
                    <SourceName>PonTaurus DB</SourceName>
                    <SourceLastUpdatedDate>
                      2001-03-01
                    </SourceLastUpdatedDate>
                    <SourceNumberOfRecords>3068</SourceNumberOfRecord
s>
                </OriginalSource>
                <DatasetDerivations>
                    <DatasetDerivation>
                        <DateSupplied>2003-08-11</DateSupplied>
                        <Supplier>
                            <Organisation>
                                <OrganisationName>
                                  Botanic Garden and
                                  Botanical Museum Berlin-Dahlem
                                </OrganisationName>
                            </Organisation>
                            <Addresses>
                                <Address>
                                  KÄonigin Luise Str. 6-8, D-14191
                                  Berlin, Germany
                                </Address>
                            </Addresses>
                            <EmailAddresses>
                                <EmailAddress>
                                  m.doering@bgbm.org
                                </EmailAddress>
                            </EmailAddresses>
```

```
                        <URLs>
                            <URL>http://www.bgbm.org</URL>
                        </URLs>
                    </Supplier>
                    <Rights>
                        <LegalOwner>
                            <Person>
                                <PersonName>Markus
DÄoring</PersonName>
                            </Person>
                            <Addresses>
                                <Address>
                                  KÄonigin Luise Str. 6-8, D-14191
                                  Berlin, Germany
                                </Address>
                            </Addresses>
                            <EmailAddresses>
                                <EmailAddress>
                                  m.doering@bgbm.org
                                </EmailAddress>
                            </EmailAddresses>
                        </LegalOwner>
                    </Rights>
                </DatasetDerivation>
            </DatasetDerivations>
            <Units>
                <Unit>
                    <UnitID>1008_1</UnitID>
                    <Identifications>
                        <Identification>
                            <TaxonIdentified>
                                <HigherTaxa>
                                    <HigherTaxon>Fabaceae</HigherTax
on>
                                </HigherTaxa>
                                <NameAuthorYearString>
                                  Astragalus haussknechtii Bunge
                                </NameAuthorYearString>
                                <AuthorString>Bunge</AuthorString>
                                <ScientificNameAtomized>
                                    <Botanical>
                                        <Genus>Astragalus</Genus>
```

<FirstEpithet>
haussknechtii
</FirstEpithet>
</Botanical>
</ScientificNameAtomized>
</TaxonIdentified>
<Identifier>
<IdentifierPersonName>
<PersonName>Markus
DÄoring</PersonName>
</IdentifierPersonName>
</Identifier>
</Identification>
</Identifications>
<UnitStateDomain>
<SpecimenUnit>
<UnitPreparation>
<PreparationType>
dried and pressed
</PreparationType>
</UnitPreparation>
</SpecimenUnit>
</UnitStateDomain>
<Gathering>
<GatheringDateTime>
<DateText>30-07-1999</DateText>
<ISODateTimeBegin>
1999-07-30
</ISODateTimeBegin>
</GatheringDateTime>
<GatheringAgents>
<GatheringAgent>
<Person>
<PersonName>Markus
DÄoring</PersonName>
</Person>
</GatheringAgent>
</GatheringAgents>
<Project>
<ProjectTitle>PonTaurus 1999</ProjectTitle>
</Project>
<GatheringSite>

```
                    <LocalityText>
                      upper Arpalik (Maden) Deresi, slopes along
                      the road from Darbogaz to tarn KaragÄol.
                      N37±27¶47¶¶ O34±36¶82¶¶
                    </LocalityText>
                    <Country>
                        <CountryName>Turkey</CountryName>
                        <ISO2Letter>TR</ISO2Letter>
                    </Country>
                    <NamedAreas>
                        <NamedArea>
                            <NamedAreaName>Nigde</NamedAre
aName>

                        </NamedArea>
                    </NamedAreas>
                    <Altitude>
                        <MeasurementAtomized>
                            <MeasurementScale>Meter</Measure
mentScale>

                            <MeasurementLowerValue>
                              2080
                            </MeasurementLowerValue>
                        </MeasurementAtomized>
                    </Altitude>
                    <BiotopeData>
                    <BiotopeText>
                      Grazed swards and open
                      thorn-cushion communities.
                    </BiotopeText>
                </BiotopeData>
                <Aspect>
                    <AspectText>N</AspectText>
                </Aspect>
                <Slope>
                    <MeasurementAtomized>
                        <MeasurementScale>
                          decimal degree
                        </MeasurementScale>
                        <MeasurementLowerValue>
                          27
                        </MeasurementLowerValue>
                    </MeasurementAtomized>
```

```
                </Slope>
            </GatheringSite>
        </Gathering>
        <CollectorsFieldNumber>
          1008
        </CollectorsFieldNumber>
    </Unit>
    <Unit>
        <UnitID>1008_2</UnitID>
        <Identifications>
            <Identification>
                <TaxonIdentified>
                    <HigherTaxa>
                        <HigherTaxon>Fabaceae</HigherTaxon>
                    </HigherTaxa>
                    <NameAuthorYearString>
                      Astragalus haussknechtii Bunge
                    </NameAuthorYearString>
                    <AuthorString>Bunge</AuthorString>
                    <ScientificNameAtomized>
                        <Botanical>
                            <Genus>Astragalus</Genus>
                            <FirstEpithet>
                              haussknechtii
                            </FirstEpithet>
                        </Botanical>
                    </ScientificNameAtomized>
                </TaxonIdentified>
                <Identifier>
                    <IdentifierPersonName>
                        <PersonName>Parolly</PersonName>
                    </IdentifierPersonName>
                </Identifier>
            </Identification>
        </Identifications>
        <UnitStateDomain>
            <SpecimenUnit>
                <UnitPreparation>
                    <PreparationType>
                      dried and pressed
                    </PreparationType>
                </UnitPreparation>
```

```
                    </SpecimenUnit>
                </UnitStateDomain>
                <Gathering>
                    <GatheringDateTime>
                        <DateText>30-07-1999</DateText>
                        <ISODateTimeBegin>
                          1999-07-30
                        </ISODateTimeBegin>
                    </GatheringDateTime>
                    <GatheringAgents>
                        <GatheringAgent>
                            <Person>
                                <PersonName>Markus
DÄoring</PersonName>
                            </Person>
                        </GatheringAgent>
                    </GatheringAgents>
                    <Project>
                        <ProjectTitle>PonTaurus 1999</ProjectTitle>
                    </Project>
                    <GatheringSite>
                        <LocalityText>
                          upper Arpalik (Maden) Deresi, slopes along
                          the road from Darbogaz to tarn KaragÄol.
                          N37±27¶47¶¶ O34±36¶82¶¶
                        </LocalityText>
                        <Country>
                            <CountryName>Turkey</CountryName>
                            <ISO2Letter>TR</ISO2Letter>
                        </Country>
                        <NamedAreas>
                            <NamedArea>
                                <NamedAreaName>Nigde</NamedAreaNa
me>
                            </NamedArea>
                        </NamedAreas>
                        <Altitude>
                            <MeasurementAtomized>
                                <MeasurementScale>Meter</Measurement
Scale>
                                <MeasurementLowerValue>
                                  2080
```

```
                                    </MeasurementLowerValue>
                                </MeasurementAtomized>
                            </Altitude>
                            <BiotopeData>
                                <BiotopeText>
                                  Grazed swards and open
                                  thorn-cushion communities.
                                </BiotopeText>
                            </BiotopeData>
                            <Aspect>
                                <AspectText>N</AspectText>
                            </Aspect>
                            <Slope>
                                <MeasurementAtomized>
                                    <MeasurementScale>
                                      decimal degree
                                    </MeasurementScale>
                                    <MeasurementLowerValue>
                                      27
                                    </MeasurementLowerValue>
                                </MeasurementAtomized>
                            </Slope>
                        </GatheringSite>
                    </Gathering>
                    <CollectorsFieldNumber>
                      1008
                    </CollectorsFieldNumber>
                </Unit>
            </Units>
        </DataSet>
    </DataSets>
</content>
<diagnostics>
    <diagnostic>OK</diagnostic>
</diagnostics>
</response>
```

BioCASe 필터 인코딩

<filter>는 SQL 문의 where 절을 표현한다. 이것은 XML 태그로 지정된 서로 다른 논리 연산자와 비교연산자의 조합을 사용하는 중복된 구조(nested structure)이다.

다음 연산자들이 지원된다:

- 이진 비교 연산자: equals, notEquals, lessThan, lessThanOrEquals, greaterThan, greaterThanOrEquals, like
- 비교 연산자: isNull, isNotNull
- 다중 인자를 위한 비교 연산자: in
- 일진 논리 연산자: not
- 이진 논리 연산자: and, or

다음은 ABCD에 기초하여 3~4개의 조건문을 설명하는 더욱 복잡한 필터 예제이다.

```
<filter>
    <and>
        <like path="/DataSets/DataSet/Units/Unit/Identifications/
          Identification/TaxonIdentified/NameAuthorYearString">
            Abies*
        </like>
        <or>
            <like path="/DataSets/DataSet/Units/Unit/Identifications/
              Identification/TaxonIdentified/HigherTaxa/HigherTaxon">
                Pinace*
            </like>
            <and>
                <like path="/DataSets/DataSet/Units/Unit/Gathering/
                  GatheringSite/Country/CountryName">
                    *Russia*
                </like>
                <greaterThan path="/DataSets/DataSet/Units/Unit/
                  Gathering/GatheringDateTime/ISODateTimeBegin">
                    2002-04
                </greaterThan>
            </and>
        </or>
    </and>
</filter>
```

2개의 개념을 서로 비교하는 것은 가능하지 않다. IN 연산자는 특별한 형태인데 비교될 개념에 여러 개의 값을 가진다.

```
<filter>
    <in path="/DataSets/DataSet/Units/Unit/
      Identifications/Identification/TaxonIdentified/HigherTaxa/
```

```
        HigherTaxon">
          <value>Pinaceae</value>
          <value>Pinophyta</value>
          <value>Pinophytina</value>
      </in>
</filter>
```

BioCASe 개념 바인딩

이전에 언급한 것 처럼, 개념 바인딩(즉, 개념 스키마의 개념이 어떻게 메시지에서 참조되는가)은 간단한 XPath와 같은 표현을 사용하여 이루어진다. 그래서 프로토콜 메시지는 간단한 XML 파서로 검증될 수 없고 개념 스키마 정의에 대해 메시지를 분석하고 비교하는 별개의 프로그램이 필요하다.

XPaths는 개념에 대한 식별자 역할을 하고 실제 XPath로 해석되지 않는다. 이것은 이름공간으로 구별되는 개념 스키마내의 하나의 개념에 대해 ID 이상의 의미를 갖지 않는다. 그러므로 경로 분리자 '/'를 사용하고 child:axis와 결합한 절대 경로만이 유효하다. 속성은 항목 경로를 가지는 속성 이후에 '[@=name]'를 사용하여 표시한다.

이 기술로 재귀적이지 않은 모든 중복된 스키마와 같이 동작할 수 있다.

BioCASe 연산 호출

메시지 전달을 위해 어떤 CGI 인자가 사용되어야 하는가에 관한 존재하는 사양은 없다. 그렇지만 인자 'query'를 사용하는 것이 권고된다.

5.2.4 현존하는 BioCASe 소프트웨어

BioCASe 프로토콜을 구현하는 소프트웨어는 현재 BioCASE 프로젝트[42] 자체에서 주로 개발되고 있다:

- PyWrapper[43]는 요청 메시지를 파싱하고 응답 메시지를 모으는 제공자 소프트웨어의 핵심이다. 모두 Python으로 작성되었다.
- 근접 질의도구(querytool)를 포함하는 제공자 소프트웨어 패키지[44]. PyWrapper의 수정은 또한 설정도구(configtool)[45]의 수정을 필요로 할 것이다.

[42] http://www.biocase.org/dev
[43] http://www.biocase.org/dev/wrapper/
[44] http://www.biocase.org/dev/provider/
[45] http://www.biocase.org/dev/configtool

- 메시지 브로커 역할을 하는 Java 기반 유닛로더 클래스[46]. 이것은 간단한 API로 프로토콜 메시지를 생성하고, 몇 개의 데이터소스에 병렬로 쓰레드 질의를 보내고, 응답을 한 곳에 모으는 일을 한다. 이것은 또한 많은 시간이 소요되는 완전한 결과를 얻어야 할 때 대안적인 e-mail 서비스를 구현한다.
- BioCASE 수집 메타데이터 네트워크, 특히 Java 기반 메타로더 소프트웨어[47]는 메타프러파일 데이터 제공자 네트워크를 모으는 책임이 있다.
- 오스트리아 GBIF 노드는 현재 또 다른 메시지 브로커와 자신의 국가 네트워크를 위해 색인 시스템을 개발중이다.

[46] http://www.biocase.org/dev/unitloader
[47] http://www.biocase.org/dev/metaloader

6 다른 표준과 기술

통합 과정동안, 다른 표준과 기술이 조사되었다. 특히, 강력하고 일반적인 질의 언어로 두개의 프로토콜을 대체할 수도 있는 XQuery; 시공간과 위치 기반 서비스로 비슷한 문제를 처리하고 있는 OGC 표준; 마지막으로 웹 서비스를 개발하는 수단으로 인기가 증가하는 SOAP.

6.1 XQuery

XQuery는 새로운 세대의 질의 언어 중 일부이고 이것은 현존하는 선택사항(예: SQL)보다 더 일반적인 것을 목표로 한다. 동시에 구조 및 반구조 문서, 관계형 데이터베이스, 객체 저장소, 또는 심지어 웹 서비스[48]를 포함하는 서로 다른 데이터 소스간 동작하도록 명령문(statements)을 표현할 수 있다. XQuery는 검색 및 갱신 연산뿐만 아니라 검색되는 데이터의 변환(transformation) 및 재구성에 사용될 수 있다.

분산되고 이질적인 데이터베이스 네트워크에서, XQuery를 사용하는 가능한 방법은 개념 스키마의 데이터 구조에 기반하여 명령문을 작성하는 것이다. 각 데이터 제공자는 자신의 근접 데이터베이스를 동일한 개념 스키마에 맞추어 맵핑하였을 것이기 때문에 이러한 문장은 근접 데이터 구조에 따라 랩퍼 소프트웨어에 의해 번역될 수 있다. Xqeury의 풍부함과 복잡성 때문에 이것의 파서를 작성하는 것은 너무 많은 경비가 소요될 것이다. 그래서 랩퍼에 이와 같은 파서를 두는 대신에 랩퍼는 번역된 질의를 처리해야 할 XQuery 명령문을 다룰 수 있는 근접 XML 서버 또는 미들웨어(middleware) 소프트웨어에 전달할 수 있다.

고급 사용자가 자신의 XQuery 명령문을 작성할 수 있는 인터페이스가 있을 수 있고, 또한 자동으로 가장 많이 사용되고 요구되는 XQuery 명령문의 타입을 생성할 수 있는 또 다른 간단한 인터페이스가 있을 수 있다.

그러나 XQuery의 유연성의 결과중의 하나는 어떻게 데이터 제공자가 하나의 응답내에 반환될 데이터의 양을 제약할 것인지는 불투명하다는 것이다. 이 쟁점사항은 또한 페이징 기능에도 영향을 줄 것이다. DiGIR 프로토콜에서 두 가지 특징(레코드의 수의 제약 및 페이징)이 존재하고 중요한 것으로 간주되고 있다.

안정되다고 여겨지지만, 현재의 XQuery 사양 버전은 여전히 워킹 드래프트(working draft) 단계이고, 2~3개의 구현 시스템만 존재한다. 현재 대부분의 이용가능한 XML 서버와 XML 미들웨어 도구는 판매 소프트웨어[49]이고 이것 모두가 완전한 XQuery 사양을 지원하는 것은 아니다. 상업 소프트웨어를 사용하는 것은 분

[48] http://www.w3.org/TR/xquery-use-cases/
[49] http://www.rpbourret.com/xml/ProdsMiddleware.htm

명히 수 많은 데이터 제공자로 이루어진 네트워크에는 너무 큰 경제적 부담이다. 그리고 그 경비는 라이센스 뿐 아니라 장비 및 교육도 또한 포함해야 한다.

XQuery 파서는 대부분 순수 XML 데이터베이스에 구현되었고 이것 중의 몇몇은 무료이고 오픈 소스[50]이다. 그러나 이것은 거의 모든 운영 데이터베이스에 대해 빈번한 데이터 추출을 요구하고 보다 큰 데이터 집합을 질의할 때 성능이 또한 문제가 될 수 있다. 더욱이 대부분의 데이터베이스 관라자들은 XML 데이터베이스 제품에 대해 여전히 익숙하지 않다.

반면에 2~3개의 상업 관계형 데이터베이스만이 초기(prototype) XQuery 인터페이스[51]를 출시했고 이것은 현존하는 데이터제공자의 다양한 인프라 구조를 망라하기에는 충분하지 않다.

이전에 제시된 것과 비슷한 시나리오에서 사용될 잠재적인 후보로 하나의 특정한 도구가 식별되었다. Xquark 프로젝트[52]는 분산 환경에서 전체 XQuery 사양을 구현하는 것처럼 보이는 한 쌍의 자유(free) 및 오픈 소스 도구(Xquark Bridge와 Xquark fusion) 작업을 진행하고 있다. 두개의 도구는 플랫폼 독립적이고(Java), 각 근접 데이터 소스는 객체-관계형 맵핑 기술을 통해 자신의 데이터 구조를 개념 스키마에 매핑하는 것이 필요하다. 지원되는 데이터베이스의 수는 아직 제한적이고 웹 서비스 인터페이스는 아직 이용가능하지 않지만, 이러한 도구를 사용할 수 있는 가능성을 평가하기 위해 더욱 자세한 연구를 하는 것은 분명히 흥미로울 것이다. 이러한 연구의 긍정적인 결과로, 프로토콜은 XQuery로 대체될 수 있고 거의 모든 현존하는 소프트웨어는 앞에서 언급한 도구 또는 비슷한 것으로 대체될 수 있다.

6.2 OGC 표준

오픈 지리공간 협회(Open Geospatial Consortium, OGC)는 지리공간과 위치기반 서비스의 표준을 개발하는 국제 조직이다. 적어도 OGC 표준의 2개가 여기서 거론되고 있는 프로토콜의 특별 관심 사항이다. 웹 특징 서비스[53](Web Feature Service, WFS)와 일반 목록 질의 언어[54](Common Catalogue Query Language, CQL).

웹 특징 서비스는 검색, 질의, (삽입, 삭제 그리고 갱신과 같은) 변형 연산을 가능하게 하면서 일련의 (일반 객체로 보일 수 있는) 특징을 노출한다.

각 서비스는 하나 또는 그 이상의 특징 타입(객체의 타입)을 다룰 수 있고 각 특징 타입은 자신의 구조를 정의하는 연관된 XML 스키마를 가진다. 특징 타입과 이것

[50] http://www.rpbourret.com/xml/ProdsNative.htm
[51] http://www.oreillynet.com/lpt/wlg/4991
[52] http://xquark.objectweb.org/index.html
[53] http://www.opengis.org/docs/02-058.pdf
[54] http://www.opengis.org/docs/02-059.pdf

의 구조는 “DescribeFeatureType”을 통해 발견될 수 있다.

연관된 XML 항목 정의는 “substitutionGroup” 메커니즘을 사용해 일반적인 GML[55] 특징항목을 상속 받을 필요가 있지만, 각 특징 타입은 자유롭게 자신의 맞춤 구조를 가진다. 그리고 연관된 XML 타입은 GML “AbstractFeatureType”을 확장할 필요가 있다.

이것의 특성(properties) 가운데 (이것 역시 복잡할 수 있음) 특징(features)은 GML 단순 기하학 타입의 하나에 기초하여 전형적으로 기하학-값의 특성을 갖는다. 간단한 기하학 정보는 2차원의 좌표 또는 선형 보간법에 따른 커브 형태로 표현된다. OGC는 지리적 특징에 사용될 수 있는 많은 공간 연산자를 정의했다: “Equals”, “Disjoint”, “Touches”, “Within”, “Overlaps”, “Crosses”, “Intersects”, “Contains”, “Dwithin”, “Beyond”.

수집과 관찰 행위(events)는 지리적 특징의 범주의 속하기 때문에, 이러한 연산자의 장점을 취할 수 있다. 새로운 프로토콜에 비슷한 공간 연산자를 추가하는 것은 분명히 흥미로울 것이다.

이 특징의 다른 한면은 각각은 데이터베이스 연산을 가능하게 하기 위해 유일한 식별자를 필요로 한다는 것이다. 근접적으로 유일한 식별자는 이 목적에 충분하다. 그러나 서비스 URL로 대표되는 서비스 범위와 결합하여, 근접적으로 유일한 식별자는 쉽게 전역적으로 유일한 식별자가 될 수 있다. WFS 사양은 정확하게 서비스 URL을 포함하는 세계 유일 식별자를 통해서 특징 인스탄스를 참조할 수 있는 유용성에 대해 지적하고 있다. 그래서, 식별자로서의 기능이외에 이것은 또한 주소일 수 있고 특징의 기본적인 표현은 그 주소를 접근함으로써 서비스 될 수 있다. 권고되기는 하지만, 이것의 기능(functionality)은 구현 세부 결정에 맡겨졌고 WFS 사양에서는 다루어지지는 않는다. 우리 경우에, URL 주소를 사용해서 표본 또는 다른 생물다양성 객체를 곧 바로 참조할 수 있는 것은 또한 흥미로울 것이다. 이 경우 유용한 속성은 객체 버전이다. 객체 버전은 선택적인 특징 속성으로 WFS 사양에서 정의되어 있다.

특징(또는 특징의 집합)은 “GetFeature” 연산을 통해 가져올 수 있고 이것은 공간 연산자를 제외한 DiGIR와 BioCASe 필터와 아주 유사한 필터 사양을 수용한다. 비교, 논리 공간 연산자를 포함하는 OGC 필턴 인코딩 정의는 CQL 사양에서 다루어지고, 이것은 자신의 식별자를 통해 특징을 참조하는 가능성을 포함하고 있다.

필터 인코딩 사양에 정의된 비교 연산자는 다음과 같다: "PropertyIsEqualTo", "PropertyIsNotEqualTo", "PropertyIsLessThan", "PropertyIsGreaterThan", "PropertyIsLessThanOrEqualTo", "PropertyIsGreaterThanOrEqualTo", "PropertyIsLike", "PropertyIsNull", "PropertyIsBetween".

[55] http://www.opengis.org/docs/02-023r4.pdf

논리연산자는 다음과 같다: "and", "or", "not".

기본적 특징 표현은 자신의 완전한 구조이지만, “GetFeature” 요청은 특정한 특성을 요청할 수 있다. 특징의 성질(properties)은 간단한 XPath 표현문(부분적인 특징 구조에 대해 두개의 필터와 “GetFeature” 요청안에서)으로 참조된다. 서비스는 또한 요청에서 요구되는 것과 관계없이 항상 포함되어야 할 필수적인 성질에 대한 결정할 수 있다.

WFS 사양은 현재 제안되는 새로운 프로토콜에 핵심적이지 않지만 특징 잠금(“GetFeatureWithLock”, “LockFeature”)과 트랜잭션(삽입, 갱신, 삭제 연산을 가진 “Transaction”) 등을 포함하는 추가적인 방법을 다루고 있다. 트랜잭션은 동시에 몇 가지의 연산을 포함할 수 있다.

WFS 사양에 따르면, 그러나 서비스는 모든 메소드(method)를 구현하지 않을 수 있다. 서비스의 완전한 설명은 “GetCapabilities” 요청을 사용하여 가져올 수 있다. 이와 같은 요청에 대한 응답은 서비스, 제공되는 요청 타입, 이용 가능한 특징 타입의 리스트 (그리고 각각에 대한 가능한 연산), 필터 인코딩 역량에 관한 섹션 (이용 가능한 함수와 연산자를 나열)에 관한 일반적인 정보를 포함한다.

ABCD와 DarwinCore와 같은 생물다양성정보 분야에서 정의되고 있는 표준은 어쩌면 전역 특징 타입 정의로 볼 수 있다. 그렇지만 ABCD의 경우, 특징의 가장 논리적인 후보는 “unit” 항목일 것이다. 이것은 위에서 정의된 모든 메타데이터 항목을 배제할 것이다. DarwinCore의 경우, 이것은 항목은 일반적인 “생물학 레코드” 항목의 특징 성질이 될 수 있다. 이러한 변화에 관계없이, 특징 타입 정의는 다른 GML 항목에서 유도된 항목이어야 하고 이것의 타입은 GML에 정의된 추상 타입을 확장(extend)하여야 하기 때문에 두 경우의 스키마는 직접 사용될 수 없다. DiGIR/DarwinCore에서와 같이, WFS는 내용 정의를 프로토콜에 바인딩한다.

WFS를 이 보고서에서 고려하고 있는 프로토콜과 비교할 때, WFS 사양에서 다루고 있지 않지만 DiGIR와 BioCASe에서 정의되는 한가지 요청 타입(“inventory”와 “scan”)이 있다. 이 요청 타입은 맵핑된 “개념”(즉 WFS 용어에서 특징 성질)의 독특한 값을 가져올 수 있는 가능성을 제공한다.

지금 DiGIR의 검색 요청에서 레코드 구조 사양을 통해 할 수 있는 것처럼, 특징 성질을 재구조화하여, 특징의 맞춤 뷰를 얻는 것이 또한 가능해 보이지 않는다.

확장성을 고려할 때, 특징 성질은 특징 타입 정의를 변화(새로운 항목을 포함 또는 XML 타입을 확장)시키어 확장될 수 있다. 특징 성질의 이름은 이름공간으로 한정(qualify)지을 수 있다.

특정한 구조적 서술 데이터[56](Structured Descriptive Data, SDD) 필요성과 관련하여, WFS에서 데이터를 가져올 때 용어 또는 사전 항목을 참조하는 "정규화된(normalized)" 특징을 출력하는 것이 가능하지 않은 것 같다. 그리고 다른 특징 타입의 인스탄스를 반환하는 역량이 현재의 SDD 항목간의 관계(relationships)를 표현하기 위해 충분한지는 분명하지 않다. 이런 역량은 보통 서로 다른 특징 타입과 관련된 다중질의를 포함하는 요청과 연관된다.

WFS를 포함한 모든 OGC 표준에 관한 중요 정보 하나는 많은 수로 증가하는 기존 SOAP 도구와 랩퍼의 이익을 얻기 위해 완전히 SOAP과 WSDL에 기반으로 사양이 변화하고 있다는 것이다[57]. 그러므로 WFS 사양으로의 이전을 심각히 고려하기 전에 다음 버전을 기다리는 것이 현명할 것이다.

6.3 SOAP

SOAP은 분산환경에서 정보교환을 위한 일반화된 XML기반 프로토콜이다. 이것은 원격 메소드를 호출할 수 있는 기능 등의 확장 가능한 메시징 틀(framework)을 정의한다. 또한 이것은 인자와 반환 값을 인코딩하기 위한 간단한 방법을 제공한다. 메시지는 실제로 여러 개의 기반 프로토콜(HTTP, TCP, SMTP) 위에서 교환될 수 있 모든 이러한 설계는 플랫폼 또는 프로그래밍 언어에 세부적인 것이 될 수 있는 모든 사항을 추상화하려고 시도한다.

SOAP은 기존의 프로토콜이 이것을 기반으로 할 수 있는 추가적인 프로토콜 계층임이 분명하다. 단지 SOAP을 사용하는 것은 DiGIR와 BioCASe간의 차이를 해소될 수는 없을 것이다. 그러나, 통합 프로토콜은 SOAP에 기반할 수 있고 또는 이것과 독립적일 수 있다.

현재, 상당한 수의 웹 서비스가 SOAP을 사용하여 구현되었고 대부분의 프로그래밍 언어에 대해 이용 가능한 SOAP 랩퍼가 있다. 이러한 상황은 웹 서비스를 구현할 때 SOAP을 당연하게 선택하도록 한다.

실제로, SOAP과 웹 서비스는 동의어가 아니다. W3C의 정의에 따르면, "웹 서비스는 URI로 식별되는 소프트웨어 시스템이고, 이것의 공개 인터페이스와 바인딩은 XML를 사용하여 서술된다. 다른 소프트웨어 시스템으로 이것의 정의를 발견할 수 있다. 그 이후 이러한 시스템은 인터넷 프로토콜로 전송된 XML 기반 메시지를 사용하여 정의에서 서술된 방식으로 웹 서비스와 상호 작용할 수 있다[58]."

분명히, SOAP이 완벽하게 웹 서비스 아이디어에 적합하기는 하지만, SOAP은 이러한 정의를 만족시키는 시스템을 개발하기 위한 요구사항은 아니다. 아직 형식

[56] http://160.45.63.11/Projects/TDWG-SDD/
[57] http://www.opengis.org/press/?page=pressrelease&view=20040318OWS2PR
[58] http://www.w3.org/TR/2002/WD-ws-gloss-2002111

(formal) 인터페이스 서술이 부족하다는 사실을 제외하고 DiGIR와 BioCASe 현재 구현 시스템은 이미 웹 서비스로 볼 수 있다. 이러한 맥락에서, 인터페이스는 보통 현재 SOAP 메시지, HTTP GET과 POST 동사, MIME 형태에 대한 바인딩을 포함하는 WSDL[59]를 통해 서술된다. SOAP 사용하지 않고 XML 메시지를 교환하는 서비스에 대한 특정한 바인딩은 MIME 바인딩 사양의 MimeXml 항목[60]을 통해서 WSDL로 서술할 수 있다.

그럼에도 불구하고, 여기에서 토론되고 있는 프로토콜의 몇몇 측면과 SOAP 기반 프로토콜간에는 많은 유사한 점들이 있다. 두 가지 모두 헤더와 내용 (또는 몸체) 부분으로 이루어진 유사한 메시지 구조를 가지고 있다. 불가능하지는 않지만, XML 스키마를 통하여 몸체 부분내의 내용을 검증하는 것이 어렵게 만들기 때문에, 이러한 일반적인 설계는 통상적으로 SOAP을 사용하는 단점으로 간주된다. 비슷한 접근 방식 때문에, DiGIR와 BioCASe는 적어도 검색 응답 메시지에서 같은 제약을 가지고 있지만 다른 타입의 메시지는 XML 스키를 통해서 완벽하게 검증될 수 있다. SOAP을 사용하면, 모든 메시지 타입의 내용은 이 제약을 가지게 될 것이다.

프로토콜 사양의 관점에서, SOAP으로 이동하는 것은 상대적으로 쉬울 것이다. 헤더 항목은 SOAP 헤더로 이동할 수 있고, 내용 항목은 SOAP 몸체 부분으로 이동할 수 있고, SOAP의 표준 결함 전략(standard fault strategy)은 진단항목 대신에 사용될 수 있을 것이다.

구현 관점에서, 개발자들은 수 많은 기존 SOAP 랩퍼의 이익을 취할 수 있을 것이다. 사용되고 있는 SOAP 바인딩 양식에 따라, 랩퍼가 완전히 XML 파싱, 데이터 직렬(serialization), 비직렬(deserialization)을 담당할 수 있어 개발자가 프로그래밍 할 때 사용하는 것처럼 객체와 변수를 직접 사용할 수 있게 할 것이다. 비록 랩퍼가 올바르게 필터와 레코드 구조 인자와 같은 더욱 복잡한 데이터 구조를 다룰 수 있는가는 분명하지 않지만 일반적으로, 이것은 서버와 클라이언트 개발을 쉽게 할 것이다. 그러나 이러한 설비 사항들은 대부분 하나의 특정 SOAP 바인딩 스타일인 RPC/encoded와 관계되어 있다. SOAP에는 4가지 가능한 바인딩 스타일[61]이 있다. SOAP 인코딩 사용은 실제로 서로 다른 랩퍼간 상호운용성 문제 때문에 권고되지 않고 있다. 그리고 RPC 바인딩 스타일 또한 확정성 문제[62] 때문에 권고되지 않고 있다. 그래서 SOAP을 사용할 경우, 확장성과 상호운용성 쟁점사항을 피하기 위해, document/literal로 불리는 바인딩을 사용하는 것이 권고되고 이것은 개발자 측면에서 메시지를 파싱하고 생산하는데 현재까지 진행된 만큼의 같은 양의 일이 필요함을 의미한다.

현존하는 아키텍쳐에서 SOAP 랩퍼를 도입함으로써 성능이 어떻게 영향을 받을지는 불명확하다. 왜냐하면 메시지 파싱 작업의 부분이 현재 소프트웨어에서 SOAP

[59] http://www.w3.org/TR/wsdl.html
[60] http://www.w3.org/TR/wsdl# mime:mimeXml
[61] http://www-106.ibm.com/developerworks/webservices/library/ws-whichwsdl/
[62] http://www-106.ibm.com/developerworks/webservices/library/ws-soapenc/

랩퍼로 옮겨갈 것이 때문이다. 비록 PHP의 경우 최신 버전(5.0.0)은 이미 내장 SOAP 모듈을 포함하지만, 특히, 해석 언어(interpreted language)를 위한 소프트웨어는 아마도 성능이 느려질 수 있다. 이것은 더욱 상세한 사례 연구를 통해 효과적으로 평가될 수 있을 것이다.

이론적으로 랩퍼의 사용 없이 SOAP 기능을 기존 소프트웨어에 끼워넣는 것이 가능하지만 새로운 사양을 고민하고 최신의 것으로 유지기 위하여 추가적인 프로토콜 단계(level)가 필요할 것이다. 이러한 노력은 오랜 시간동안 매우 클 수 있다. 그래서, SOAP을 사용하는 경우, SOAP 랩퍼를 사용하는 것이 권고된다. 그리고 이것은 아마도 코드에 상당한 영향을 야기할 것이다.

SOAP 랩퍼가 사용되는 방식에 따라 클라이언트 또는 서버 코드가 그 특정 랩퍼 구현과 웹 서비스 SOAP 바인딩 자체에 묶이는(bound) 경향이 있다는 것을 언급하는 것은 흥미롭다. 그러므로, 서로 다른 SOAP 랩퍼간, 또는 같은 랩퍼의 서로 다른 버전, 또는 서로 다른 웹 서비스 바인딩간의 이동은 쉽지 않고 시간이 오래 소요되는 작업일 수 있다. 이미 관련된 WSDL 파일의 서비스의 논리적인 서술을 직접적으로 사용하여 웹 서비스 접근에 독립적으로 바인딩을 제공하는 더욱 일반적인 랩퍼[63]가 있다. SOAP을 사용하는 경우, 개발자를 위한 권고는 이러한 일반적인 랩퍼를 사용하는 것, 또는 자신의 코드와 SOAP 랩퍼사이에 적어도 하나의 계층을 사용하는 것이다.

SOAP을 사용하는 것과 특정 XML 프로토콜에 관련된 어플리케이션(application)을 유지하는 것 두개 중에 어느 것을 결정하는 것은 이 경우 쉬운 작업이 아니다. 두가지 접근방법은 장점과 단점을 가진다. SOAP 기반 프로토콜은 SOAP 공동체에서 개발 되고 있는 모든 도구와 확장성에서 이익을 얻을 수 있다. 예를 들어 보안 전략[64]과 메시지 중계 도구(message brokering tools)이다. 개발자들은 이용 가능한 많은 랩핑 도구를 가질 수 있을 것이다. 반면에 서로 다른 SOAP 랩퍼[65]간에 여전히 3~4가지의 상호운용성 문제가 있고, 앞으로 SOAP과 관련된 쟁점사항이 대부분의 개발 업무 시간을 또한 차지할 가능성이 있다[66]. 코드 및 성능에 미칠 상당한 효과와 관련된 위험성을 주로 고려할 때, 권고안은 프로토콜을 SOAP과 관계없이 유지하는 것이다. 미래에 SOAP을 사용하는 장점이 더욱 명확해진다면, 프로토콜의 새로운 버전은 이것에 바탕을 둘 수 있다.

그러나, 프로토콜을 SOAP에 기반하지 않는다고 해서, 메시지 브로커 위에 SOAP 서비스를 가지는 못하는 것은 아니다. 메시지 브로커는 확실히 생물다양성 공동체에서 가장 많이 사용되는 서비스의 일종이기 때문에 이것은 흥미로운 사례연구가 될 것이다. 이것 위의 SOAP 서비스는 아마 사용자 인터페이스의 개발과 또한 서로 다른 타입의 분산 웹 서비스를 연결하려고 하는 네트워크를 가진 메시지 브로커의

[63] http://www-106.ibm.com/developerworks/webservices/library/ws-wsif.html
[64] http://www.nwfusion.com/news/tech/2002/1216techupdate.html
[65] http://www.xmethods.com/soapbuilders/interop.html
[66] http://www.artima.com/webservices/articles/whysoapP.html

통합을 쉽게 할 수 있을 것이다.

7 통합 제안

7.1 종합적 전략

제안된 프로토콜은 DiGIR와 BioCASe의 현재의 차이를 제거하는 것을 목적으로 하지만 새로운 특징을 포함하고 다른 표준의 새로운 아이디어를 결합한다. 새로운 프로토콜이 상대적으로 단기간에 구현될 수 있도록 기존의 소프트웨어와 네트워크에 대한 영향을 또한 고려하였다.

다른 기술 및 표준을 검토할 때, XQuery는 주목할 만한 유연성과 능력 때문에 미래에 채택될 잠재적인 질의 언어로 고려되었다. 아직 워킹 드래프트 수준이지만, 곧 표준이 될 것으로 기대된다. 잘 알려지고 세계적으로 수용되는 질의 언어를 사용함으로써 모두는 다른 공동체가 생산한 도구, 문서 및 지식의 이익을 얻을 수 있다. 그러나 확실히 XQuery 파서(이것은 개수와 스펙과의 일치면에서 아직 제한적이다)에 대한 의존이 있을 것이다. 분산 이질적인 시나리오에서 XQuery 사용은 근접 데이터 모델을 추상화하는 방법을 찾기 위한 추가 연구를 또한 필요로 할 것이다. XQuery를 사용하는 일관적인 제안은 더욱 깊은 연구와 제안되기 전에 프로토타입에 대한 작업을 할 가치가 있을 것이다.

OGC 사양, 즉 WFS와 CQL,은 DiGIR와 BioCASe와 비교할 때 흥미로운 유사점들을 가지고 있다. CQL 필터링 사양은 실제로 DiGIR와 BioCASe에서 사용되고 있는 필터보다 더욱 강력하고, 그러므로 제안하는 프로토콜에 새로운 아이디어를 제공하였다. 그리고 질의의 객체로 “특징”을 정의할 때 WFS 사양은 다른 패러다임(paradigm)을 사용하는데 이것은 어떻든 DiGIR 레코드를 가져오는 것과 비슷하지만 BioCASe 문서를 가져오는 것과는 다르다. OGC 사양이 새로운 아이디어를 위해 소스였지만, 이것은 통합하려고 하는 프로토콜의 기존 기능중의 몇몇 부분(목록 여산)이 또한 부족하였다. 이러한 누락된 기능에 관계없이, 단지 하나의 OGC 사양 채택을 제안하는 것은 통일성에 맞지 않을 것이다. 왜냐하면 대부분의 사양이 조금씩 연결 되기 때문이다. – 확장된 GML 사양을 포함하여 모든 것을 채택하는 것이 아마 필요할 것이다. OGC 사양이 SOAP 프로토콜에 맞게 변화하고 있다는 것을 고려하면, 진정으로 이것을 채택하려는 제한은 우선 새로운 버전을 기다려야 할 것이다.

일반적인 프로토콜 계층으로 SOAP을 사용하는 것을 또한 고려하였다. 이것은 현재의 프로토콜에 어떤 기능도 추가하지 않을 것이지만 구현 시스템은 기존의 SOAP 랩퍼, 코드 생성자(code generators) 및 더욱 쉬운 인터페이스 정의(WSDL은 SOAP 바인딩을 포함한다)로부터 이익을 얻을 수 있다. SOAP 메시지 구조와 DiGIR와 BioCASe에서 사용하는 일반적인 메시지 형태(제안되는 프로토콜에서 유지)간의 유사성은 SOAP으로 이전을 분명히 쉽게 할 것이다. 그러나 상위 단계 프로토콜 계층으로서 SOAP을 채택하는 일관된 제안은 SOAP 구현 시스템간 존재하

는 많은 수의 성능 및 상호운용성 문제 때문에 프로토타입 작업과 철저한 테스트를 할 필요가 있을 것이다.

제안되는 프로토콜은 주요 메시지 구조를 유지하였고 이미 DiGIR와 BioCASe에 있는 연산을 통합하였다.

7.2 서비스 타입

분산 데이터베이스에서 검색 및 가져옴 연산을 수행하는 네트워크의 전형적인 아키텍쳐는 적어도 3개의 서로 다른 서비스가 관여한다.

- 메시지 브로커 서비스: 다른 서비스에 메시지를 배포고 응답을 한곳에 모으는 책임이 있다.
- 제공자 서비스: 메시지 브로커의 특화된 버전, 메시지는 근접 자원에만 배포되기 때문임. 이것은 해당 데이터 소스에 많은 설정을 할 수 있는 소프트웨어 설치에 해당한다.
- 데이터소스 서비스: 요청 과정의 맨 끝에 위치하는 것으로, 이것은 요청을 근접 질의 언어로 번역하고 특정한 데이터 저장소에서 데이터를 가져오는 책임이 있는 서비스이다. 이것은 보통 하나의 데이터베이스 또는 이것의 부분집합과 연관된다.

모든 서비스는 분명히 공통의 프로토콜을 제시하는 많은 유사성을 가진다. 그렇다 하더라도, 서비스는 분명히 동일하지 않다. 그리고 모든 서비스의 메시지를 검증하기 위해 같은 프로토콜 스키마를 사용하는 것은 이 스키마를 더욱 복잡하고 (비현실적인 항목의 조합을 가능하게 하면서) 작은 제약사항을 두게 할 것이다. 모든 서비스를 목표로 하는 하나의 사양은 바람직할만한 명료함이 부족할 것이다.

만약 모든 네트워크를 계단 방식으로 추상화하면서 우리가 다른 메시지 브로커를 접근하는 메시지 브로커 서비스가 설정될 수 있게 생각한다면 차이는 더욱 분명해진다. 이 시나리오에서 데이터소스에 도착하는 많은 방법이 있을 수 있다. 무한 루프를 피하도록 주의가 요구되고, 질의를 최적화하기 위해 네트워크의 나무 형태를 가져오는 특별 추가 요청이 바람직할 것이다.

여기에 있는 통합 제안은 데이터소스와 직접 통신하기 위해 사용되는 프로토콜에만 중점을 둔다. 이것은 통합 과정에서 가장 중요한 부분으로 인식되었다.

7.3 접근점

DiGIR와 BioCASe 네트워크간 주요 차이점 중의 하나는 데이터소스 서비스가 DiGIR 네트워크에는 존재하지 않는다는 것이다. DiGIR 자원은 헤더내의 특정 속성

을 통하여 프로토콜에 의해서만 가리켜진다. 반면에 BioCASe는 제공자 서비스를 가지지 않는다. – 이것은 데이터소소를 이것의 접근점을 통해서 직접 처리한다.

접근점은 서비스와 통신하기 위해 사용하는 URL이다. UDDI와 같은 저장소를 통한 서비스 발견을 최적화 하기 위해 특히, 제공자가 서로 다른 주제(themes)와 관련된 3~4가지의 데이터소스를 가질 때 각 데이터소스가 자신소유의 접근점을 가지는 것이 바람직하다. 발견 메커니즘, 메시지 브로커 설정, 그리고 클라이언트 소프트웨어는 특정한 데이터소스에 직접 통신하고 참조할 수 있는 것으로 이익을 얻는다. 제안된 프로토콜에서 각 데이터소스는 자신의 접근점을 가져야만 한다.

7.4 개념 바인딩

DiGIR와 BioCASe간의 중요 차이점 중의 하나는 어떻게 개념이 참조되는가와 관련이 있다. DiGIR는 프로토콜에서 정의한 추상 항목의 확장으로서 개념을 정의하는 것을 선정하였고 이것은 대체 그룹(substitution group)을 통해 이루어진다. 이 접근 방법은, 개념이 어떻게 사용되는지를 조정, 더욱 많은 검증 가능성을 제공하였지만, 이것은 또한 개념 스키마를 프로토콜 정의에 묶이는 전역 항목의 평면 리스트로 제한하였다.

이것은 BioCASE가 외부적이고 계층적인 스키마의 어떠한 항목이든지 참조할 수 있도록 자신의 프로토콜을 개발하게 된 주된 요인이었다. BioCASe는 자신의 간략형태(/)안에서 자식 축을 사용하고 간단한 절대 XPath 표현문을 통해 개념을 참조한다.

개념 바인딩과 관련한 새로운 프로토콜의 주요 요구사항은 다음과 같았다.

- 프로토콜로부터 완전히 외부적이고 독립적인 스키마에서 개념을 참조할 수 있는 것
- 개념을 참조할 때 개념 스키마를 지정하는 것. 그리하여 데이터소스가 다중 개념 스키마를 사용할 수 있도록 가능하게 하는 것.
- 개념 스키마가 서로 다른 형태로 정의될 수 있도록 가능하게 하는 것

제시된 프로토콜은 외부 개념 정의에 대한 모든 종류의 참조를 가능하게 하고 서로 다른 스키마의 개념들을 분명히 구분할 수 있게 하면서, BioCASe의 개념 바인딩을 확장한다. 제안된 개념 바인딩은 다음과 같은 성질을 가진다:

- 개념은 XML 속성인 “path”를 통해 참조되고 프로토콜에 완전히 독립적이다.
- 경로는 개념에 대한 식별자로 간주될 수 있다.
- 개념 경로는 각각의 개념 스키마 이름공간 접두사가 와야 한다(이것이 역량 응답안에서 나열될 때를 제외하고).

개념 스키마 접두사 이후에, 개념 식별자는 거의 모든 스트링을 사용할 수 있다. 개념 스키마가 순수한 XML 스키마로 정의될 때, 개념을 식별하기 위한 제시된 형태는 인스탄스 문서에서 해당되는 항목을 가리키는 간단한 XPath 표현문을 사용하는 것이다.

다음 XML Schema는 간단한 개념 스키마로 볼 수 있다:

```
<schema xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="dataset">
        <complexType>
            <sequence>
                <element name="record"
                        minOccurs="0"
                        maxOccurs="unbounded">
                    <complexType>
                        <sequence>
                            <element name="scientificName" type="string"/>
                            <element name="basisOfRecord" type="string"/>
                        </sequence>
                        <attribute name="catalogNumber"
                                type="string"
                                use="required"/>
                    </complexType>
                </element>
            </sequence>
        </complexType>
    </element>
</schema>
```

이 경우에, 개념들은 프로토콜내에서 다음과 같이 참조될 수 있다:

```
<concept path="cs:dataset/record/scientificName"/>
```

또는

```
<concept path="cs:dataset/record/@catalogNumber"/>
```

여기에서 “cs” 는 개념 스키마와 연관된 이름공간 접두사이고 문서의 앞에 다음과 같이 선언된다:

```
xmlns:cs="http://example.net/schemas/cs/1.0"
```

7.5 필터 인코딩

필터 인코딩과 관련하여 DiGIR와 BioCASe의 차이점은 다음과 같다:

- DiGIR는 부정(negation)을 표현하기 위해 결합된 논리 연산자를 사용(<antNot>, <orNot>) 하지만 BioCASe는 단항 <not> 연산자를 사용한다.
- DiGIR는 NULL 값을 표현하기 위해 "xsi:nil" 속성을 사용하지만, BioCASe는 특정한 비교 연산자를 가진다 (<isNull>, <isNotNull>).
- DiGIR는 개념 항목의 내용으로 값을 정의하지만, BioCASe는 비교 연산자 항목의 내용으로 값을 정의한다.

이러한 차이를 분석하면서, 제시된 프로토콜을 부정을 표현하기 위한 단항 논리 연산자를 가지는 더 낳은 방법을 고려하였다. 실용적인 용어로는, 필터가 앞서는 이항 논리 연산자 없이 부정조건으로 시작할 수 있도록 한다(이것은 DiGIR에서 가능하지 않다). 결과적으로 모든 결합 연산자는 제외되었다. <isNull> 연산자는 "xsi:nil"보다 더욱 명시적인 것으로 생각되었고 선정된 개념 바인딩 기술에 적합하였다. 끝으로, 연산자 항목 또는 개념 항목의 내용으로 값을 정의하는 대신에 하나의 속성으로 새로운 <literal>을 가지는 것이 더 낳은 해법으로 고려되었다.

제시된 프로토콜은 많은 변화가 있었고 아래와 같은 추가적인 기능을 제공하기 위하여 새로운 특징들을 융합하였다:

- 이항 논리 항목내에서 두 가지 이상의 조건을 사용하는 가능성 (이 경우, 이항 논리 항목의 위치는 각각의 논리 연산자를 통해 연결된 조건들의 순서에 대한 경계만을 나타낸다).
- 같은 조건내에서 두개의 개념을 비교하는 가능성 (하나의 예는 최소 고도가 최대 고도보다 높은 레코드에 대한 전형적인 데이터 정제 요청 검색이 될 수 있다).
- 인자를 사용하는 가능성 (서비스가 환경변수의 값을 찾는 것을 지시)
- 수치 연산자를 사용하는 가능성

공간 연산자와 함수와 같은 다른 중요한 개선은 또 다른 연구를 필요로 하기 때문에 향후 버전을 위해 남겨 놓았다.

새로운 필터 인코딩에 관한 요약된 예제는 다음과 같다:

7.5.1 표현문

DiGIR와 BioCASe 프로토콜에 새로운, 표현문은 필터의 기본 단위이다. 이것들은 비교 명령문을 만들기 위해서만 사용되고 유효한 표현문은 리터럴(literal), 개념, 인자 또는 수치 연산으로 구성될 수 있다.

리터럴은 고정된 값을 나타내고 다음과 같은 형태를 가진다:

<literal value="42"/>

개념은 개념 스키마에 정의된 개념을 참조하기 위해 제안된 개념 바인딩을 사용한다 (이것은 차례로 근접 데이터 저장소에 맵핑된다.):

<concept path="cs:dataset/record/scientificName"/>

인자는 리터럴과 비슷하지만 고정된 값 대신에 HTTP POST와 GET 환경변수와 같은 외부 소스에서 값을 가져온다. POST/GET 변수내의 인자의 이름은 인자 정의에서 이루어진다.

<parameter name="sname"/>

앞의 인자 표현문은 다음 예제 URL에 대해 "Ficus"의 값을 가진다.

http://example.net/a.cgi?sname=Ficus

4개의 기본 수치 연산 <add>, <sub>, <mul>, 은 모두 이항이고 인자로 2개의 표현문을 가진다. 첫번째 인자는 최좌측의 것으로 다루어진다. 수치 표현문 13+7은 다음과 같은 표기된다:

```
<add>
  <literal value="13"/>
  <literal value="7"/>
</add>
```

7.5.2 비교 연산자

제안된 연산자는 이것들이 취하는 인자(argument)의 수에 따라 분류될 수 있다:

- 단항 연산자: <isNull>
- 이항 연산자: <equals>, <greaterThan>, <lessThan>, <greaterThanOrEquals>, <lessThanOrEquals>, <like>
- 범위가 없는 연산자: <in>

모든 연산자는 첫번째 인자로 개념을 취한다. 단항 연산자는 어떤 부가적인 인자를 취하지 않는다. 이항 연산자는 둘째 인자로 표현문을 취할 수 있다. 범위가 없는 연산자는 부가적인 인자로서 하나 또는 그 이상의 리터럴(literal) 표현문을 취한다.

간단한 비교는 다음과 같다:

```
<equals>
    <concept path="cs:dataset/record/scientificName"/>
    <literal value="Stipa"/>
</equals>
```

7.5.3 논리 연산자

통합 제안은 논리 연산자를 기본적인 3개로 줄였다: <and>, <or>, <not>. 단항 <unary> 연산자는 비교 또는 논리 연산자인 하나의 인자를 가진다. 다른 2개의 연산자 <and>와 <or>는 이항 형태에서 제한 없는 것으로 바뀌어서 어떤 수의 인자를 가질 수 있지만 적어도 2개는 가져야 한다. 더욱 자세한 표현문의 아래와 같다:

```
<and>
    <like>
        <concept path="cs:dataset/record/scientificName"/>
        <literal value="Stipa%"/>
    </like>
    <equals>
        <concept path="cs:dataset/record/basisOfRecord"/>
        <literal value="observation"/>
    </equals>
    <not>
        <in>
            <concept path="cs:dataset/record/@catalogNumber"/>
            <values>
                <literal value="101"/>
                <literal value="102"/>
            </values>
        </in>
    </not>
</and>
```

7.6 응답 구조와 뷰

검색 연산에서 구조화된 응답을 반환할 수 있음에도 불구하고, DiGIR와 BioCASe는 어떻게 결과가 지정되고 생성되는 점에서 상당한 차이가 있다.

DiGIR는 검색 응답을 생성하기 위해 “record structure” 구조에서 의존한다. DiGIR 레코드 구조는 응답에서 어떤 개념이 반환되고 어떻게 구조화되어야 하는가를 나타내기 위해서 스키마(XML Schema 언어의 부분집합 사용)를 사용한다. 동시에 각 DiGIR 자원은 설정의 과정에 응답 메시지에서 “record” 항목의 순환 및 생성의 기초 역할을 하는 ‘루트 테이블(root table)”을 정의할 필요가 있다. 이러한 이유로, 레코드 구조는 개념 또는 복합 타입을 참조하는 일련의 항목을 포함하는 복합 타입으로 구성된 <record> 항목의 정의를 포함해야 한다. 개념은 XML Schema “ref” 속성으로 참조되기 때문에, 값은 해당되는 개념의 같은 이름을 가진 XML 항목의 내용으로서만 반환될 수 있다.

반면에 BioCASe는 데이터소스가 사용하는 개념 스키마의 구조에 기초하여 응답을 생성한다. 각 개념 스키마는 응답을 맵핑하고 생성하기 위한 기반으로 사용되기 위해서 또 다른 XML 파일로 변환된다. 결과적으로 응답은 항상 개념 스키마의 가튼 구조를 따르지만 이것은 하나의 “루트 테이블”의 레코드에 묶이지 않는다.

제안된 프로토콜은 더욱 일반적인 대안으로 “response structures”를 사용하는 “record structure”의 아이디어를 확장함으로써 두 가지 접근방법의 해결점을 찾으려고 하였다.

응답 구조는 또는 XML Schema 언어의 스키마로 개념이 어떻게 반환되어야 하는가를 지정하지만 고정된 레코드 바인딩을 가지지 않는다.

응답 구조는 실제로 계산과 페이징을 위한 참조로 사용되는 맵핑 섹션과 색인 항목 정의를 가진 뷰 정의의 일부이다.

뷰는 다음과 같은 형태를 가진다:

```
<view>
    <structure>
        <schema
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://example.net/schemas/sp1">
            <xs:element name="dataset">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="specimen"
                                    minOccurs="0"
                                    maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element                    name="acnum"
type="xs:string"/>
```

```
                                        <xs:element                    name="sname"
type="xs:string"/>
                                    </xs:sequence>
                                    <xs:attribute name="lastupdate"
                                                  type="xs:dateTime"
                                                  use="optional"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </schema>
        </structure>
        <indexingElement path="/dataset/specimen"/>
        <mapping xmlns:s="http://example.net/cs/1.0">
            <nodes>
                <node path="/dataset/specimen/acnum"/>
                <concept path="s:CatalogNumber"/>
            </nodes>
            <nodes>
                <node path="/dataset/specimen/sname"/>
                <concept path="s:ScientificName"/>
            </nodes>
            <nodes>
                <node path="/dataset/specimen/@lastupdate"/>
                <concept path="s:DateLastUpdated"/>
            </nodes>
        </mapping>
</view>
```

<schema>내의 첫번째 <element>정의는 응답을 만들 때 루트 항목으로 항상 고려되어야 한다. 그래서 이 구조에 따르는 (외부 프로토콜 항목을 배제한) 가능한 결과는 다음과 같을 수 있다.

```
<dataset xmlns="http://example.net/schemas/sp1">
    <specimen lastupdate="2002-05-15T13:20:00Z">
        <acnum>423</acnum>
        <sname>Thylacinus cynocephalus</sname>
    </specimen>
    <specimen lastupdate="2001-02-19T16:04:01Z">
        <acnum>567</acnum>
        <sname>Sarcophilus Harrisii</sname>
```

    </specimen>
</dataset>

맵핑 섹션은 향후 프로토콜의 버전에서 확장될 수 있는 간단한 1:1 맵핑을 사용한다. 이것은 응답 구조에서 데이터 소스에서 사용되는 개념 스키마의 개념으로 노드(내용 항목 또는 속성)를 맵핑한다.

XML Schema 언어는 광범위하고 복잡하기 때문에, 랩퍼가 모든 사양을 지원할 것으로 기대되지 않는다. 이 언어의 축소된 부분집합만이, 즉 기본적으로 목표 이름공간(target namespace), (기수를 갖는) 항목, 속성, 순서(sequences), 그리고 (단순 및 복합 내용을 갖는) 복합 및 단순 타입의 근접 선언 등이 요구된다. XML Schema 언어의 추가적인 면은 랩퍼가 지원할 수 있고 역량 응답에서 광고 될 수 있다.

복합 항목과 이것의 부분 항목 또는 속성사이의 포함 관계는 응답을 생성하기위해 데이터소스 백엔드 특정 질의 언어(datasource backend specific query language)에 번역되어야 한다. 관계형 데이터베이스의 경우, 테이블(tables), 조인(joins), 그리고 개념 스키마와의 매핑 등과 관련된 데이터소스의 설정 세팅을 고려하여 하나 또는 다수의 SQL 명령문이 사용될 수 있을 것이다.

결과 구조는 항목의 내용 또는 속성의 값, 수용되는 모든 맞춤 이름 등과 같이 다방면으로 개념을 배열할 수 있고 결과는 외부 XML Schema에 대해 검증될 수 있다.

개념 스키마에서 응답 구조의 개념(notion)을 분리함으로써, 다중 개념 스키마에서 서로 다른 뷰를 취할 수 있을 뿐만 아니라 개념 스키마가 (아마도 XML Schema가 아닌) 다른 방법으로 정의될 수 있는 여지가 있다.

7.7 개괄적인 메시지 형태

개괄적인 메시지 형태는 유지되었다. 루트 항목, <request> 또는 <response>는 <header> 세션과 연산(operation) 이름을 가진 뒤따라오는 다른 섹션을 포함해야 한다. 단지 하나의 차이점은 이전의 두개의 프로토콜은 헤더내에 <type>을 사용하여 연산을 정의하고, 그 다음으로 헤더에 뒤이어 일반적인 <content> 항목을 사용했다 그러므로, 지정된 타입과 실제 내용 항목간에 불일치가 발생할 수 있었다. 일반적인 <content> 항목을 연산 이름으로 대체함으로써 검증은 더욱 엄격해졌고 스키마는 더욱 가독성이 있게 되었다.

요청에서, 연산 항목은 건네 받을 가능한 인자를 포함한다.

<?xml version="1.0" encoding="utf-8"?>

```
<request xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <search>
        <!-- search operation specific parameters -->
    </search>
</request>
```

응답에서, 연산 항목은 가능한 결과를 포함하고 뒤를 이어 경고 또는 에러뿐만 아니라 임의의 적절한 추가 정보를 포함하는 <diagnostics> 섹션을 포함한다:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <search>
        <!-- search operation result -->
    </search>
    <diagnostics>
        <!-- diagnostics information -->
    </diagnostics>
</response>
```

7.7.1 헤더

헤더 또한 본질적으로 동일하게 유지되었지만 몇몇의 추가적인 변화가 제시되었다. <type> 항목을 제거하는 것 이외에 제안된 주요한 변화는 다음과 같다:

- 데이터소스는 이제 자신의 접근점으로 참조되기 때문에, DiGIR 네트워크에서 사용되던 “resource” 속성은 목적지 항목에서 제거되었다
- <source> 항목은 통신 과정에 관여한 모든 서비스(stages)가 사용하는 주소를 추적(track)하기 위해 여러 번 발생할 수 있게 하였다. 중간의 서비스는 리스트의 끝에 자신의 주소를 반드시 포함해야 한다. 추적 과정을 보관함으로써, 계단 방식의 메시지 브로커의 구현 시스템은 무한 루프를 피할 수 있을 것이다.
- 접근점은 내용 대신에 이제 <source>의 속성이고 ”sendtime”의 시간기록은 속성으로 통합되었다.
- <source> 내부에 메시지를 처리하는 데 사용되는 소프트웨어의 이름과 버전을 가리키기 위해 선택적인 <software> 항목이 있다.
- 목적지 항목은 더 이상 필요하지 않다. 왜냐하면 프로토콜이 접근점으로 도달할

수 있는 데이터소스 서비스만을 다루기 때문이다.

메시지 브로커가 발송한 요청의 헤더는 다음의 예제와 같다:

```
<header>
    <source accesspoint="11.12.13.14"
            sendtime="2001-12-17T09:30:47-05:00">
        <software name="generic client software"/>
    </source>
    <source accesspoint="15.16.17.18"
            sendtime="2001-12-17T09:30:49-05:00">
        <software name="generic portal" version="0.95"/>
    </source>
</header>
```

이전 요청의 응답 헤더는 다음과 같을 수 있다:

```
<header>
    <source accesspoint="http://example.net/datasource/a.cgi"
            sendtime="2001-12-17T09:30:50-05:00">
        <software name="generic wrapper" version="1.1.4"/>
    </source>
    <destination accesspoint="11.12.13.14"/>
</header>
```

7.7.2 진단

응답에 나타나는 <diagnostic> 섹션은 이전의 것과 거의 같고 다음과 같이 하나 또는 그 이상의 <diagnostic> 항목을 포함한다.

```
<diagnostics>
    <diagnostic code="PRG_MISSING_LIBRARY" type="error">
        Could not find module MBSTRING
    </diagnostic>
</diagnostics>
```

이전의 “severity”은 “type”으로 이름이 변경되었고 이전과 같이 다음의 항목을 포함한다: debug, info, warn, error, fatal

그러나, 클라이언트 소프트웨어 관점에서, 진단 코드의 통합은 중요하게 생각되었다. 완전한 코드 리스트는 프로토콜의 마지막 사양에서 나타나야 한다.

7.8 요청 연산과 응답 타입

프로토콜을 위한 제안된 연산은 다음을 포함한다:

- Metadata: 서비스를 서술하는 기본 정보를 포함한다.
- Inventory: 개념의 리스트에서 구별되는 값을 가져올 때 사용된다.
- Search: 맞춤 응답 구조로 근접 데이터 소스에서 데이터를 가져올 때 사용되는 주요 연산
- View: 근접 데이터에서 미리 정의된 뷰를 얻을 때 사용된다.
- Capabilities: 서비스 기능에 관한 핵심 사항과 기술 정보를 포함한다.
- Ping: 서비스가 이용 가능한지 체크하기 위해 모니터링 목적으로 사용된다.

7.8.1 메타데이터 연산

메타데이터의 주요 아이디어는 레이블, 초록, 키워드, 관련 객체 및 연락 정보 등을 포함하는 서비스에 대한 몇몇 기본 서술을 가져오는 것이다.

기존의 DiGIR 메타데이터 연산은 이 제안의 기반으로서 역할을 했지만 알려진 많은 쟁점사항을 처리하기 위해 몇 가지 면에서 변화가 있었다. 주요 변화는 다음과 같다:

- 데이터 접근점의 포함 (데이터소스와 직접 통신을 한다는 점만을 고려하면 중복된 것으로 볼 수 있지만 실제로 응답이 서비스의 주소를 알지 못하는 클라이언트 소프트웨어로 갈 수 있다는 것을 고려한 핵심적인 것이다).
- 많은 언어로 서비스 될 내용 항목에 “lang” 속성의 포함과 “unbounded”로 바뀐 기수(cardinality)
- 중복되었던 <implementation> 항목의 제거 (헤더 항목은 <software>항목에 이 정보를 이미 포함하고 있다).
- <minQueryTermLength>를 역량 응답으로 이동
- 역량 응답에서 <maxElementLevels>과 <maxElementRepetitions>으로 대체된 <maxSearchResponseRecords>와 <maxInventoryResponseRecords>의 제거. <maxElementLevels>은 응답 구조에서 XML 깊이 차원의 역할을 하는 최대 중복(nested) 항목을 제한하는데 사용될 수 있다. <maxElementRepetitions>은 검색 응답에서 XML 너비 차원으로 작용을 하는 각 항목의 최대 반복 생산 횟수를 제한하는데 사용될 수 있다.
- 특정한 데이터소스에만 의미가 있는 <recordBasis>와 <recordIdentifier> 항목의 제거
- <useRestrictions>을 일반적인 <rights> 항목으로 변경
- 이전의 자원 <name>을 <label>로 이름 바꿈

- 리스트에서 이용 가능한 근접 뷰의 포함
- <dateLastUpdated>와 <numberOfRecords> 항목을 뷰의 선택 속성으로 변경 (이제 데이터소스는 서로 다른 것들의 표현 역할을 하기 때문에)

언급할 만한 다른 사항은, 데이터소스가 제공자 서비스를 통해서 도달할 수 있더라도, 메타데이터 응답에서 보여지는 분명한 관계설정(explicit relationship)이 없다는 것이다. 데이터소스는 완전히 독립적인 서비스로 보여진다.

"제공자(provider)" 용어는 과거에 모든 가능한 의미로 많은 혼란을 초래하였다: 제공자 서비스(provider service), 제공자 소프트웨어(provider software), 원본 데이터 제공자(original data provider), 그리고 제공자 기관(provider institution). 제안된 프로토콜에서 데이터소스 메타데이터 응답에서 제공자 서비스와 분명한 관계설정이 없지만 제공자 기관에 명확한 감사표시(credits)를 제공할 수는 있다. 실제로 이제 데이터소스 서비스는 원한다면 많은 객체(조직, 기관, 회사 등)에 감사표시를 제공할 수 있다.

객체는 또한 동시에 하나 이상의 역할(role)을 할 수 있고, 역할은 네트워크에 의해 정의된다. 개체와 연관된 연락정보 리스는 DiGIR에서 사용되던 것과 같다. 다른 흥미로운 점은 같은 객체가 하나 이상의 서비스로 참조될 수 있는 것이다. 이러한 경우를 위해, 이제 많은 서비스에 걸쳐 반복 없이 객체의 리스트를 생산하는 데 사용될 수 있는 이제 세계 유일 식별자로 <identifier> 항목이 있다.

메타데이터 요청은 어떤 인자도 필요하지 않다:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <metadata/>
</request>
```

메타데이터 응답은 아래와 같다:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <metadata>
        <label lang="en">Plant specimen database</label>
        <label lang="pt_BR">
```

```
    Banco de dados de Esp¶ecimes de Plantas
</label>
<accesspoint>
    http://example.net/datasource/a.cgi
</accesspoint>
<abstract lang="en">
    This resource contains specimen records
    of plants from all over the world
</abstract>
<keywords lang="en">plant, specimen</keywords>
<citation lang="en">
    Plant specimen database. Retrieved on (date
    accessed). http://example.net/datasource/a.cgi
</citation>
<rights lang="en">Users may not distribute, modify,
    transmit, reuse, repost, transfer, or use any
    content or information from this service for
    commercial purposes without prior written permission.
</rights>
<conceptualSchemas>
    <conceptualSchema
        namespace="http://example.net/schemas/specimen/1.0"/>
</conceptualSchemas>
<views>
    <view name="specimen"
            dateLastUpdated="2004-08-01T20:00:00-03"
            numberOfRecords="100"/>
</views>
<relatedEntities>
    <entity lang="en">
        <identifier>
            http://example.net/organisation/mydata.xml
        </identifier>
        <name>Biodiversity Informatics Institute</name>
        <name lang="pt_BR">
            Instituto de Inform¶atica para Biodiversidade
        </name>
        <acronym>BII</acronym>
        <logoURL>http://example.net/mylogo.gif</logoURL>
        <role>provider</role>
        <role>host</role>
        <description>
```

```
                Organisation created for this example
            </description>
            <relatedInformation>
                http://example.net/organisation/
            </relatedInformation>
            <contact type="administrative">
                <name>Person A</name>
                <title>Curator</title>
                <email>person_a@example.net</email>
                <phone>+ 111 11 111111</phone>
            </contact>
            <contact type="technical">
                <name>Person B</name>
                <title>Sysadmin</title>
                <email>person_b@example.net</email>
                <phone>+ 111 11 222222</phone>
            </contact>
        </entity>
    </relatedEntities>
</metadata>
</response>
```

마지막 예제에서와 같이, “lang” 속성은 또한 <metadata>, <entity> 또는 <contacts> 항목에서 기본 세팅으로 선언될 수 있다.

7.8.2 목록 연산

개념에서 독특한 값을 가져오기 위해 사용되던 목록 연산은 몇몇 사소한 차이점이 있지만 DiGIR와 BioCASe 모두에 있다. 서로 다른 이름(DiGIR에서 “inventory”, BioCASe에서 “scan”)을 제외하고, DiGIR는 요청에서 필터를 수용하고 서로 다른 2개의 계산 과정을 가진다: 독특한 값의 총 개수와 각 독특한 값의 발생 횟수. 이 모든 특징은 새로운 프로토콜에 있다. 그러나 DiGIR와 BioCASe는 개념을 조합하였을 때는 독특한 값을 요청할 수 없지만, 이제는 이것이 가능하다.

제안된 프로토콜에서 목록 연산의 주요 특징은 다음과 같다:

- 요청에서 선택적인 필터가 사용될 수 있다.
- 하나 또는 그 이상의 개념이 인자로 사용될 수 있다 (하나 이상의 개념이 지정될 때, 이러한 값의 독특한 조합이 반환된다).
- 개념은 서로 다른 개념 스키마의 부분일 수 있다.
- 선택적인 “count” 인자는 근접 데이터소스 저장소의 독특한 레코드의 총 개수

와 각 레코드의 발생 횟수를 요청하는데 사용될 수 있다.

- 페이징은 “start”와 “limit” 인자를 사용하여 수행될 수 있다. (“start”는 반환 될 레코드의 첫번째 색인을 나타내고 “0”부터 시작한다. 반면 “limit”는 반환될 레코드의 개수를 나타낸다)
- 응답은 <summary> 항목을 포함하고 여기에서 “start”는 반환될 첫번째 레코드의 색인, “next”는 다음 번 요청으로 가져올 수 있는 다음 레코드의 색인, “totalReturned”는 반환된 레코드의 개수, “totalMatched”는 질의와 일치한 레코드의 총 개수(반환될 필요는 없음)를 가리킨다.

목록 요청은 다음과 같다:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <inventory count="true"
               start="0"
               limit="2"
               xmlns:ns1=http://example.net/ns1
               xmlns:ns2="http://example.net/ns2">
        <concepts>
            <concept path="ns1:record/taxon/fullname"/>
            <concept path="ns2:gazeteer/location/isocountry"/>
        </concepts>
        <filter>
            <!-- filter specific elements -->
        </filter>
    </inventory>
</request>
```

이 요청에 대한 목록 응답은 다음과 같을 수 있다:

```
<?xml version="1.0" encoding="utf-8"?>
<response>
    <header>
        <!-- header specific elements -->
    </header>
    <inventory>
        <record count="13">
            <value>Abies alba Mill.</value>
            <value>DE</value>
```

```
        </record>
        <record count="45">
            <value>Quercus robur L.</value>
            <value>DE</value>
        </record>
        <summary start="0"
                 next="2"
                 totalReturned="2"
                 totalMatched="117"/>
    </inventory>
    <diagnostics>
        <!-- diagnostics information -->
    </diagnostics>
</response>
```

<record>내의 값의 순서는 요청에서 지정한 개념의 순서와 일치하여야 한다.

응답에서 “totalMatched”와 “count” 속성은 요청이 셈(counting)을 요구할 때만 나타난다.

요청이 셈을 요청하지만 “limit”을 0으로 지정한 경우, 어떤 레코드도 반환되지 않지만, “totalMatched”는 응답에 나타난다.

7.8.3 검색 연산

검색 연산은 이미 토론된 주요 아이어의 2가지를 단순히 결합한다: 뷰와 필터. 추가 인자가 부분적인 뷰를 요청하고, 페이징과 셈을 수행하기 위해 사용될 수 있다.

검색 연산의 주요 특징은 다음과 같다:

- 검색 요청은 응답이 뷰 정의를 이용하여 응답이 어떻게 결과를 구조화해야 하는가를 지정할 수 있다.
- 뷰 정의는 요청에서 완전히 선언될 수 있다. 즉, 이것은 원격에 위치하고 참조될 수 있다 또는 아무것도 지정되지 않으면 기본적인 근접 뷰가 사용될 수 있다.
- 부분 뷰는 뷰에서 노드를 지정함으로써 요청될 수 있다.
- 필터는 이제 선택적이다.
- 페이징과 셈은 뷰 섹션의 <indexingElement>에 기반한고 있다.

“응답구조와 뷰” 섹션의 예제로 사용된 뷰가 “http://exampl.net/viewlib/specimen.vwd” 에서 이용 가능하다고 가정하면, “/dataset/specimen/sname” 항목만을 요청하는 검색요청은 다음과 같을 수 있다:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <search count="true" start="0" limit="2">
        <view location="http://example.net/viewlib/specimen.vwd"/>
        <partial>
            <node path="/dataset/specimen/sname"/>
        </partial>
        <filter xmlns:s="http://example.net/cs/1.0">
            <equals>
                <concept path="s:CatalogNumber"/>
                <literal value="423"/>
            </equals>
        </filter>
    </search>
    <diagnostics>
        <!-- diagnostics information -->
    </diagnostics>
</request>
```

요청과 관련하여:

- 어떠한 <view> 섹션도 지정되지 않으면, 기본적인 근접 뷰가 사용되어야 한다 (그리고 근접 뷰가 없다면, 에러가 발생해야 한다).
- 특정한 노드를 가진 <partial> 섹션이 지정되지 않으면, 모든 뷰 구조가 반환되어야 한다.
- 어떠한 <filter>도 존재하지 않으면, 어떠한 필터링도 사용되지 않는다.

이 요청에 대한 검색 응답은 다음과 같을 수 있다.

```
<?xml version="1.0" encoding="utf-8"?>
<response>
    <header>
        <!-- header specific elements -->
    </header>
    <search>
        <dataset xmlns="http://example.net/schemas/sp1">
            <specimen>
                <acnum>423</acnum>
```

```
                <sname>Thylacinus cynocephalus</sname>
            </specimen>
        </dataset>
        <summary start="0" totalReturned="1" totalMatched="1"/>
    </search>
    <diagnostics>
        <!-- diagnostics information -->
    </diagnostics>
</response>
```

<acnum> 항목은 요청되지 않았지만 뷰 정의에 따른 의무항목이기 때문에 이것은 항상 반환되어야 한다는 것에 주목하시오. 그러나 선택 속성 “lastupdate”는 반환되지 않았다.

검색 요청에서 얼마나 많은 항목이 반환되어야 하는 가를 지정(<indexElement> 태그를 사용)할 수 있을지라도, 응답은 “maxElementRepetitions”으로 불리는 서비스 세팅에 따라 제한될 수 있다. 이 세팅은 XML 결과에 너비 차원으로 작용을 한다(<search> 항목내의 XML 내용만을 고려하면서). 이것은 동일한 “xsd:definition” 정의에서 발생한 어떠한 항목도 허락된 최대 반복 횟수보다 더 많이 반복될 수 없다는 것을 의미한다. 이 세팅은 데이터 제공자에게 서버의 과부를 피하고 완전한 데이터 덤핑 요청을 피할 수 있는 조금의 제어를 제공하고, 동시에 이것은 페이징 제한으로 볼 수 있다.

7.8.4 뷰 연산

이 연산(DiGIR와 BioCASe에 나타나지 않은)은 제안된 프로토콜의 모든 특징을 정의한 후에 자연스러운 결과로 발생했다. 실제로 이것은 정규 검색 연산을 통해 이미 성취한 기능 이상을 제공하지 않는다. 그러나 이것은, 원천 데이터 제공자에 직접 접근을 이용하고 새로운 가능성을 열어 놓으면서, 검색을 수행 할 때 더 간단한 방법을 제공한다.

검색 응답은 본질적으로 뷰 정의와 필터 조합의 결과이므로, 근접 데이터에 직접 접근은 인자화된 뷰의 근접 정의를 통하여 상당히 쉽게 할 수 있다. 이 아이디어는 데이터소스는 자신의 근접 데이터에서 하나 또는 그 이상의 XML 뷰를 정의 할 수도 있다는 것이다.

이러한 근접 뷰는 개념 스키마가 사용되는 것을 고려하여 데이터 제공자의 뜻에 따라 설계될 수 있다. 즉, 그들은 원격에 미리 정의된 뷰를 간단히 참조할 수 있다. 각 근접 뷰는 검색과 뷰 연산이 참조할 수 있는 근접 유일 이름(local unique name)을 가진다. 근접 뷰의 하나는 기본적인 것으로 정해져야 한다.

“필터 인코딩” 섹션에서 이미 본 것처럼, 필터내의 비교 표현문은 리터털 값 또는 환경 변수 (HTTP GET 또는 POST 인자)에서 얻는 인자화된 값을 사용할 수 있다. 이 전체의 인프라구조는 근접 데이터에 간단한 직접 접근을 제공할 만큼 충분하다.

근접 뷰는 다음과 같이 역량 응답에서 정의되고 나열될 수 있다.

```
<views default="specimen">
    <view name="specimen" xmlns:s="http://example.net/cs/1.0">
        <structure>
            <schema
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://example.net/schemas/sp1">
                <xs:element name="dataset">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="specimen"
                                        minOccurs="0"
                                        maxOccurs="unbounded">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="acnum"
                                                    type="xs:string"/>
                                        <xs:element name="sname"
                                                    type="xs:string"/>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </schema>
        </structure>
        <indexingElement path="/dataset/specimen"/>
        <mapping>
            <nodes>
                <node path="/dataset/specimen/acnum"/>
                <concept path="s:CatalogNumber"/>
            </nodes>
            <nodes>
                <node path="/dataset/specimen/sname"/>
                <concept path="s:ScientificName"/>
            </nodes>
```

```
        </mapping>
        <filter>
            <equals>
                <concept path="s:CatalogNumber"/>
                <parameter name="id"/>
            </equals>
        </filter>
    </view>
</views>
```

뷰 연산은 입력으로 XML 메시지를 사용하지 않고 HTTP GET 또는 HTTP POST 호출을 통해서만이 이용 가능한 유일한 것이다. 앞에서 정의한 뷰의 간단한 호출은 다음과 같을 것이다.

http://example.net/a.cgi?operation=view&name=specimen&id=123

이 응답은 XML 표현으로 근접 데이터의 부분일 것이다. 주어진 예제에서, 인자화된 필터는 “id” 인자의 값을 가지고 근접 맵핑된 “CatalogNumber” 개념과 비교하는데 사용할 것이다. “CatalogNumber”가 유일한 값을 가진 필드와 맵핑하면 응답은 표본 레코드의 XML 표현일 것이다. 이것은 심지어 책갈피(bookmarked)로 또는 참조될 수 있을 것이다:

```
<?xml version="1.0" encoding="utf-8"?>
<dataset>
    <specimen>
        <acnum>123</acnum>
        <sname>Rubus rosaefolius</sname>
    </specimen>
</dataset>
```

뷰 연산의 응답은 기본적으로 특정한 프로토콜 항목을 포함하지 않는 유일한 것이다(헤더와 진단처럼). 이것은 뷰 구조 정의에서 주어진 XML 표현만을 포함한다. 그러나, “verbose” 인자를 사용하여 프로토콜 항목의 출현을 요청하는 것은 가능하다(이 경우 응답은 검색 응답의 동일한 구조를 따른다).

뷰는 또한 관계된 다중의 “객체”(색인 항목)”가 있을 때 페이징을 수행할 수 있다. 페이징은 “start”와 “limit”인자를 사용하여 할 수 있다.

뷰는 개념 스키마의 개념에 대해서만 정의되기 때문에, 동일한 개념 스키마를 사용하고 필요한 개념을 맵핑한 임의의 데이터소스는 뷰를 사용할 수 있다. 뷰의 라이브러리 또한 가능하고 서비스 설정동안 사용될 수 있다.

예제로 주어진 뷰의 흥미로운 점 하나는 각각의 호출 문자열은 해당되는 (underlying) 객체에 대해 세계 유일 식별자 역할을 할 수 있다는 것이다. 그러므로 이것은 주소일 수 있는 특별한 종류의 식별자이고 이것으로부터 항상 객체의 최신 버전을 직접 가져올 수 있다.

7.8.5 역량 연산

이 연산은 최초 각 스키마로부터 모든 맵핑된 개념뿐만 아니라 데이터소스에서 사용되는 개념 스키마를 가리키기 위해 BioCASe에서 구상되었다. 이것은 서비스에 대한 메타데이터로 볼 수 있지만 서비스에 대한 기본적인 서술(메타데이터)과 또 다른 기술적 정보(역량) 등 두개 연산이 더욱 편리한 것으로 결정되었다. DiGIR 메타데이터 항목의 몇몇은 여기에 서술된 역량 연산으로 옮겨졌다. 역량 연산의 아이디어는 서비스에 대한 설정, 기능, 그리고 기술 정보를 가져오는 것이다.

역량 요청은 인자를 필요로 하지 않는다:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <capabilities/>
</request>
```

역량 응답은 3~4가지 서로 다른 섹션을 포함한다:

```
<?xml version="1.0" encoding="utf-8"?>
<response>
    <header>
        <!-- header specific elements -->
    </header>
    <capabilities>
        <schemas>
            <!-- conceptual schemas being used -->
        </schemas>
        <views>
            <!-- possible XML views from local data -->
        </views>
        <settings>
            <!-- general service settings -->
        </settings>
```

```
        <operators>
            <!-- supported operators in filters -->
        </operators>
        <structure>
            <!-- supported set of response structure language -->
        </structure>
    </capabilities>
</response>
```

개념 스키마와 맵핑된 개념

이 섹션은 서비스가 사용하고 2개의 의무 속성(이름공간과 위치)을 가진 개념 스키마 리스트를 포함해야 한다. 각 개념 스키마는 맵핑된 개념의 리스트를 갖고 각각은 2개의 선택적인 속성을 있다.

- Searchable: 개념이 필터 표현문에서 사용될 수 있는지를 가리킴 (기본으로 참)이다).
- Mandatory: 개념이 모든 검색 응답에서 나타나야 하는지를 가리킴 (기본으로 거짓이다).

이 섹션은 다음과 같을 수 있다:

```
<schemas>
    <conceptualSchema
        namespace="http://example.net/schemas/specimen/1.0"
        location="http://example.net/schemas/specimen/1.0/sp.xsd">
        <concept path="scientificName"/>
        <concept path="catalogNumber"/>
        <concept path="basis" searchable="false"/>
        <concept path="ipr" mandatory="true"/>
    </conceptualSchema>
</schemas>
```

뷰(Views)

이 섹션은 검색과 뷰 요청에 의해 사용 가능한 모든 근접 뷰를 간단히 나열한다 각 뷰는 유일한 이름을 가지고 그것 중의 하나는 기본 값으로 선택되어야 한다. 뷰는 데이터 제공자를 위한 선택적인 특징이다.

```
<views default="specimen">
    <view name="specimen">
```

```
        <!-- view definition -->
    </view>
    <view name="collector">
        <!-- view definition -->
    </view>
</views>
```

개괄적인 세팅

클라이언트에 흥미로울 적어도 5가지 가능한 세팅이 있다.이것 모두로 서비스 제공자는 과대한 양의 데이터 요청으로 야기되는 서버 부하를 피할 수 있도록 제어를 할 수 있다:

- minQueryTermLenght: “like” 표현문에서 사용되는 와일드카드 문자열의 최소 길이
- maxElementRepetitios: 응답에서 임의의 반복할 수 있은 항목에 대해 허락된 최대 반복 횟수
- maxElementLevels: 응답에서 허락된 단계(level)의 최대 개수 (내용 섹션에만 관계됨)
- maxResponseTags: 응답에서 반환될 수 있는 태그의 최대 개수 (내용 섹션에만 관계됨)
- maxResponseSize: 응답에서 반환이 허락된 킬로 바이트(kilobytes) 단위의 최대 크기

이 섹션은 다음과 같을 수 있다.

```
<settings>
    <minQueryTermLength>3</minQueryTermLength>
    <maxElementRepetitions>200</maxElementRepetitions>
    <maxElementLevels>30</maxElementLevels>
</settings>
```

지원되는 연산자

이 섹션은 필터 인코딩이 어떤 연산자를 사용할 수 있는지를 가리키기 위해 사용된다. 연산자는 3개의 범주로 나뉜다.

- 논리: “and”, “or”, 그리고 “not” 연산자를 다룰 수 있는 역량을 가리키는 데 사용된다.
- 비교: “in”, “isNull”, “like”, 그리고 “basicComparativeOperators” (=,<,<=,>,>=)의 지원을 가리키기 위해 사용된다.

- 기본 수치 연산자: 더하기, 빼기, 곱하기 그리고 나누기의 지원을 가리키기 위해 사용된다.

이 섹션은 다음과 같을 수 있다:

```
<operators>
    <logical/>
    <comparative>
        <basicComparativeOperators/>
        <in/>
        <isNull/>
        <like/>
    </comparative>
    <basicArithmeticOperators/>
</operators>
```

지원되는 응답 구조

이 섹션은 응답 구조를 해석하기 위하여 서비스에 의해 XML Schema 언어의 어느 부분집합이 지원되는지를 가리키기 위해 사용된다. 모든 랩퍼 구현 시스템에 대해 최소 부분집합이 요구되고, 이것은 <basicSchemaLanguage> 태그로 표현된다. 추 XML Schema 언어의 추가적인 특징은 선택적으로 랩퍼에 의해 지원될 수 있다.

이 섹션은 다음과 같을 수 있다:

```
<structure>
    <basicSchemaLanguage/>
    <choice/>
</structure>
```

<group>, <import>, <references>, <extension>, <restriction>, 그리고 <substitutiongroup> 등의 다른 특징을 포함한다.

7.8.6 핑(Ping) 연산

이 새로운 연산은 메타데이터 요청(이것은 보통 몇몇의 근접 데이터베이스의 연결을 필요로 한다)을 사용할 필요 없이 서비스의 더 낮은 모니터링을 위해 소개되었다 요청과 응답은 매우 간단하고 도착가능, 응답 시간 그리고 랩퍼 소프트웨어가 올바르게 설치되었는지를 체크하기 위해 사용될 수 있다.

핑 요청 견본:

```
<?xml version="1.0" encoding="utf-8"?>
<request xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <ping/>
</request>
```

핑 응답 견본:

```
<?xml version="1.0" encoding="utf-8"?>
<response xmlns="http://example.net/protocol/1.0">
    <header>
        <!-- header specific elements -->
    </header>
    <pong/>
    <diagnostics>
        <!-- diagnostics information -->
    </diagnostics>
</response>
```

7.9 연산 호출

입력으로 XML 문서를 사용할 수 있는 연산에 대해, 서비스와의 통신은 하나의 HTTP POST 또는 HTTP GET 인자를 통해서 이루어지는 것으로 의견 일치를 보았다. “request”로 불리는 이 인자는 지정된 프로토콜에 맞는 XML 메시지 또는 XML 메시지를 가리키는 URL을 포함해야 한다. 이 연산은 다음을 포함한다: “metadata”, “capabilities”, “inventory”, “search” 그리고 “ping”.

“metadata”, “capabilities”, 그리고 “ping”과 같은 복잡한 인자를 요구하지 않는 연산은 또한 HTTP POST 또는 HTTP GET에 값이 연산의 이름을 가지는 “operation” 인자를 건넴으로써 호출될 수 있다.

뷰 연산은 요청에서 XML 메시지를 사용하지 않는다, 그래서 이것은 다음과 같은 HTTP POST 또는 HTTP GET 인자로만 호출된다: “operation” (항상 “view”과 같다), “name” (뷰의 이름), “verbose” (선택적으로 추가의 프로토콜 항목을 요청한다.), “start” (가져올 첫번째 객체의 색인), 그리고 “limit” (가져올 객체의 총계).

서비스가 인자가 없이 호출될 때, 기본적인 연산은 “메타데이터” 연산이다.

7.10 알려진 그리고 가능한 제약 사항들

모든 노력이 통합 과정에 집중되었기 때문에 생물다양성 공동체에 표준으로 제안된 다른 데이터 교환 표준 스키마에 의한 이 프로토콜의 사용은 알맞게 연구되지 않았다. SDD와 분류학적 개념 전송 스키마(Taxonomic Concept Transfer Schema, TCS)와 같은 스키마는 검색 응답을 생산할 때 다음과 같이 이 프로토콜에 의해 해결 또는 해결될 수 없는 특징을 요구한다.

- 재귀 항목을 반환
- 상호 관계된 항목을 반환 (구조의 다른 부분에서 항목이 다른 것을 참조할 때)

제시된 프로토콜이 유연한 응답 구조를 수용하기 때문에, 어느 정도의 맵핑 기술과 앞서 말한 스키마에 원하는 결과를 생산할 수 있는 응답 구조 정의의 조합을 사용하는 것이 가능할 것이다. 그러나 이러한 쟁점사항을 잘 조사하기 위해서는 추가 연구가 요구된다.

생물다양성데이터의 많은 부분이 어느 정도 지리학적 성질을 가지기 때문에, OGC CQL 사양에서 제공하는 것처럼, 프로토콜에서 공간 연산자를 지원하는 것이 매우 바람직할 것이다.

제안된 프로토콜에서 필터내의 맞춤 함수의 정의와 사용이 또한 가능하지 않다. 이것은 필터링 역량을 향상시킬 수 있는 흥미로운 특징이 될 것이다.

제안된 프로토콜을 따르는 소프트웨어를 개발할 때, 가장 큰 도전은 아마도 동적으로 자신의 색인 항목 정의를 가진 요청된 맞춤 구조에 응답을 만드는 것일 것이다. 동적 응답 구조를 가진 BioCASe 경험을 비추어 볼 때 이것은 가능한 것처럼 보인다. BioCASe 프로토콜은 맞춤 응답 구조를 가능하게 하지 않았지만, 이것을 저장하는 내부적인 방법은 매우 유사하고 작동이 성공적으로 증명되었다.

비슷한 구현 접근방법을 사용하여, SQL의 동적 생성은 (테이블은 정점(vertices)으로, 외부/1차 키 관계설정은 모서리(edge)로 하는) 데이터베이스 구조의 acyclic 연결 그래프 표현에 기반을 둘 수 있다. 주어진 반응 구조에 대해서, 동적 SQL 조인(joins)을 생성하는 사용될 수 있는 요청된 테이블의 경로를 유일하게 식별이 가능하도록 그래프의 루트 나무 표현이 추출될 필요가 있을 것이다.

반복되는 항목의 식별 문제를 제외하고 계층적이고 검증된 XML은 생성될 수 있다. 관계형 데이터베이스 구조가 “relational” XML 구조와 서로 다르다면, 몇몇 경우에 XML 생성에 사용되는 알고리즘이 “많은(many)” 면의 관계설정을 표현하는 정확한 XML 노드를 식별하는 문제를 가질 수 있다. 임의의 반복될 수 있는 노드는 앞선 XML 노드와 관련된 부모 레코드에 조인(join)된 각각의 테이블 레코드에 대해

XML 결과에서 생성될 필요가 있다. 연결상태(chain)의 최상위에 어떤 속성 또는 다른 어떤 맵핑된 부분(sub) 항목 없이 계단적인 반복 항목이 있을 때 문제가 발생할 수 있다. 이 경우에 알고리즘은 어느 노드에서 반복이 발생해야 하는지를 모를 수 있다. BioCASe 시스템이 응답을 위해 구조를 고정하고 근접적으로 구조를 설정한 것처럼, 이것을 해결하기 위해 근접 제공자 설정을 사용하는 것이 가능했다. 그러나 맞춤 구조 시나리오에서 이 문제를 해결하기 위한 몇몇 아이디어(직접 반복할 수 있는 항목을 맵핑하는 가능성을 포함하고 알고리즘에 추가적인 참조를 제공)가 제안되었다. 이러한 아이디어는 아마 대부분 문제를 제거할 수 있을 것이지만 모든 상황은 아니다. 그래서 사용되는 알고리즘과 제공되는 맵핑 여하에 따라 데이터소스 서비스가 지원할 수 없는 몇몇 특정 타입의 구조가 있을 수 있다.

비정규화된 데이터베이스의 경우 생성되는 XML의 자동적인 정규화는 바람직하지 않으므로 응답 XML에서 중복 데이터는 피할 수 없을 것이다. 그러나 비정규화가 전체 데이터소스에 대한 개념((수집물 이름, 기관 이름과 같은 전역 메타데이터 개념)의 동일 값을 사용한 결과라면, 개념 스키마를 맵핑할 때 고정 값을 가능하게 하여 이 문제를 극복할 수 있다. 이 경우에 랩퍼는 쉽게 반환되는 중복 정보를 피할 수 있을 것이다.

7.11 소프트웨어 영향

모든 변화와 제안된 추가사항은 확실히 DiGIR와 BioCASe 프로토콜을 다루는 것이 필요한 모든 소프트웨어에 영향을 줄 것이다. 모든 타입의 서비스는 영향을 받을 것이다.

가장 영향이 적은 소프트웨어는 메시지 브로커일 것이고 이것은 자신의 IP 주소를 헤더내에 기록하는 것이 필요할 것이고 아마 다른 에러코드를 다룰 것이다. 특히, DiGIR 메시지 브로커는 접근점에 의한 데이터소스를 처리해야 할 것이다(설정과 메시지 라우팅에 영향을 주는).

양쪽의 클라이언트 소프트웨어는 필터와 응답 구조 사양에의 변화된 부분에 맞게 있을 자신을 변화시킬 필요가 있을 것이다. 그러나, 응답 파서는 다중 개념을 수용하는 목록 또는 새로운 뷰 연산과 같은 새로운 기능을 이용하기를 원하는 것을 제외하면 많은 변화가 없을 것이다.그러나 항목 이름에서 몇몇 변화를 고려할 필요가 있을 것이다.

당분간, DiGIR 제공자 서비스는 공식적으로 프로토콜에 의해 다루어지지 않을 것이다. 이것의 설정과 설치는 일련의 데이터소스 서비스를 서비스하기 위해 변화될 필요가 있을 것이다.

DiGIR 제공자 (또한 데이터소스 서비스로 보여질 때)와 BioCASe 랩퍼는 가장 많은 영향을 받는 소프트웨어일 것이다. 새로운 필터와 응답구조 사양을 수용하고 파

싱하는 것, 임의의 검증된 구조에 맞게 결과를 생산하는 것, 다른 많은 변화를 고려하면 것은 짧은 기간 안에 가능할 지라도 상당한 작업을 요구할 것이다. 그러나 뷰 연산과 같은 새로운 특징들 그리고 다중 개념을 가진 목록 연산은 많은 영향을 초래하지 않을 것이다.

통상 랩퍼와 통합되는 설정자는 자신을 제안된 서비스 메타데이터와 프로토콜에서 이용 가능한 추가적인 설정 인자에 맞게 수정할 필요가 있을 것이다.

이러한 모든 결과를 고려할 때, 네트워크는 모든 자신의 소프트웨어 구성요소를 갱신하는 전략을 주의깊게 결정해야만 한다.

7.12 이전 전략(Migration strategy)

변경될 소프트웨어를 선정할 때, 새로운 프로토콜에 상응하여 데이터소스가 되어야 하는 DiGIR 제공자와 BioCASe 랩퍼에 우선순위가 명확하게 주어져야 한다. 이러한 구성요소에 변경을 할 때 추천방법은 중요한 서비스 다운 시간 없이 점차 대체될 수 있도록 이전의 프로토콜과 가능하면 호환성을 유지하는 것이다.

이전 것과의 호환은 요청에 의해 사용되는 프로토콜을 발견하기 위해서 이러한 구성요소 위에 추가적인 검증을 포함하고, 그 다음 서로 다른 라이브러리 또는 프러시져(procedure)로 서로 다른 프로토콜의 각각의 메시지를 처리함으로써 성취될 수 있다. 이것은 이상적으로, 설치된 BioCASe 랩퍼의 경우, 동일한 접근점을 유지하면서 행해져야 한다. 이것은 같은 서비스를 접근하는 다른 네트워크의 다른 클라이언트 소프트웨어 또는 메시지 브로커가 서로 다른 시간에 갱신될 수 있도록 할 수 있을 것이다. BioCASe 랩퍼의 또 다른 가능성은 분리되고 병렬적인 서비스로서 새로운 데이터소스를 설치 및 설정하는 것이다. 그러나 이것은 어느 때에 자신을 사용하는 네트워크의 UDDI 레지스트리 갱신이 요구될 것이다.

3~4개의 데이터소스를 대신해 작동하는 DiGIR 제공자 각각의 현재 자원에 대해 반드시 새로운 접근점을 가져야 할 것이다. 주제별 DiGIR 네트워크는 점진적으로 분리된 메시지 브로커와 클라이언트 소프트웨어를 설치 및 설정할 수 있을 것이다. 그 다음 모든 데이터소스가 갱신되었을 때 자신의 공식적인 주소를 변화시켜야 할 것이다.

새로운 버전의 데이터소스를 설치할 때 설정파일을 자동으로 갱신하는 도구가 분명히 필요할 것이다. 가능하다면, 가장 좋은 해결은 설정 파싱 소프트웨어(설정자 또는 랩퍼)가 이전 및 신규 버전의 설정파일을 읽고 필요할 때 새로운 버전을 작성할 수 있도록 하는 것이다. 이것은 필요한 독립적인 소프트웨어 구성요소의 수를 감소시키다.

또 다른 대안은 서로 다른 프로토콜간에 메시지 번역자 서비스를 할 수 있는 프락

시 서비스를 개발하는 것이다. 그러나 제안된 프로토콜의 새로운 특징은 기존의 것으로 번역될 수 없기 때문에 이것은 쉬운 일이 아닐 것이다. 비록 여기에서 그 경우가 아니어야 할지라도, 새로운 프로토콜에 나타나지 않은 기존 프로토콜의 임의의 특징은 반대방향으로의 번역 수행을 어렵게 할 수 있을 것이다. (단지 2~3가지의 더 이상 쓰이지 않는 메타데이터 항목이 제거되었다.)

7.13 개념 스키마에 대한 고찰

제안된 프로토콜은 개념 스키마와 맵핑된 데이터의 가능한 뷰를 명확히 분리한다. 이 사실은 3~4가지 방법으로 정의될 수 있는 개념 스키마를 구상할 때 커다란 유연성을 제공한다. 개념 스키마는 사용되고 있는 동일한 파싱 기술에서 이익을 얻기 위해 XML Schema 언어를 필수적으로 꼭 사용하기 보다는, 적어도 XML을 사용할 수 있다. 데이터소스가 프로토콜에 묶이지 않는 다중 개념 스키마를 수용하기 때문에, 개념 스키마의 단위화 및 확장은 다른 수단으로 성취될 수 있다.

개념 스키마를 단위화하는 것의 장점은 이것이 자신의 개념 도메인간 부분적인 공통부분만을 가진 다른 네트워크에 의해 재사용될 수 있다는 것이다. 단위화 될 때, 하나의 개념 스키마의 변화는 개념 도메인의 다른 부분에 영향을 미치지 않을 것이다. 데이터 교환의 관점에서, 개념이 동일한 스키마 또는 그렇지 않은가는 중요하지 않다. 왜냐하면 이것은 서로 다른 뷰에 의해 자유롭게 결합되고 모아질 수 있기 때문이다.

그러나 개념 스키마를 어떻게 가장 잘 표현하고 단위화 하는가는 아직 오픈 쟁점사항이다. – 이것은 이 연구의 일부가 아니었다.

프로토콜이 제공하는 가장 큰 유연성은 DarwinCore와 같은 스키마가 자신의 현재 접근 방법을, 개념정의의 간단한 리스트, 유지 또는 그렇지 않음 간에 선택하는 것을 가능하게 한 것이다. 데이터를 교환할 때 올바른 개념 집결(grouping)과 기수는 XML 뷰로 성취할 수 있다. 그리고 ABCD와 같은 XML 스키마가 개념 스키마로 간주될 때 단위화 되는 것이 또한 가능하다

서로 다른 스키마가 다른 형태로 동일한 개념을 정의하면 같은 프로토콜을 사용하더라도 데이터 제공자간에 완전한 상호운용성을 이루는 것은 어려울 수 있다. 고전적인 예제는 날짜(dates)가 어떻게 표현될 수 있는가와 관계이다. DarwinCore와 같이 날짜를 3가지 개념(일(day), 월(month), 년(year))으로 분리하는 개념 스키마는 다음과 같은 조건을 나타내기 위해 더 많은 프로토콜 특징을 필요로 한다: 날짜 >= '2002-05-19'. 이 조건을 나타내는 연관된 필터 인코딩은 다음과 같을 것이다:

```
<or>
    <greaterThan>
        <concept path="c:year"/>
```

```
        <literal value="2002"/>
    </greaterThan>
    <and>
        <equals>
            <concept path="c:year"/>
            <literal value="2002"/>
        </equals>
        <or>
            <greaterThan>
                <concept path="c:month"/>
                <literal value="05"/>
            </greaterThan>
            <and>
                <equals>
                    <concept path="c:month"/>
                    <literal value="05"/>
                </equals>
                <greaterThanOrEquals>
                    <concept path="c:day"/>
                    <literal value="19"/>
                </greaterThanOrEquals>
            </and>
        </or>
    </and>
</or>
```

ABCD와 같이, 하나의 개념을 사용하여 날짜를 정의하는 개념 스키마는 같은 조건을 다음과 같이 나타낼 것이다:

```
<greaterThanOrEquals>
    <concept path="c:ISODateTime"/>
    <literal value="2002-05-19"/>
</greaterThanOrEquals>
```

이 예제는 개념 스키마 정의간의 차이점이 같은 프로토콜을 사용하더라도 상호운용성 쟁점사항에 어떻게 도달하는지를 잘 보여준다.

개념의 의미는 서로 다른 개념 스키마를 다룰 때 더욱 많은 문제를 피하기 위해 가능하면 명확해야 한다. 다른 예제는 표본에 대해 최신 식별로 수용되는 것으로 “ScientificName”으로 불리는 개념을 정의하는 개념 스키마일 것이다. 다른 개념 스키마는 최신의 것 뿐만 아니라 표본의 임의 식별로서, 다른 의미를 가진 한 개념에 대해 같은 이름을 사용할 수 있을 것이다. 표본과 식별간 1:1 관계를 가정하는

응답 구조는 후자의 개념 정의에서 확실히 잘못된 결과를 얻을 것이다.

때때로 기수는 개념 정의에서 끼워 넣어질 수 있다. “collector” 개념은 하나의 문자열에 이어진 3~4가지의 수집자 이름을 포함할 수 있다. 반면에 다른 개념 스키마는 단일 수집자 개념이 여러 번 발생할 수 있다. 응답 구조는 더 많은 문제를 피할 수 있도록 이러한 차이점을 알아야만 한다.

그래서 DiGIR와 BioCASe 네트워크간의 완전한 통합과 상호운용성을 이루기 위한 다음 단계는 적어도 완전히 호환되는 개념의 집합을 사용해서, 개념 스키마 수준에서 협력하는 것이다; 또는 더 좋은 것은, 개념 라이브러리 또는 생물다양성데이터를 위한 온톨로지에서 유래될 수 있는 공통 집합의 개념을 채택하는 것이다.

8 추가적인 권고안

전체 통합 과정동안, 제안되고 있는 프로토콜과 상호 관련한 3~4가지의 추가전인 토픽이 또한 토론되었다. 이것은 권고된 사양의 리스트 및 바람직한 도구에서부터 조사하기에 흥미로울 수 있는 특정 연구 영역까지 다양했다. 이러한 토론의 요약을 다음 절에 나타내었다.

8.1 사양

공통의 프로토콜에 의견 일치에 도달하자마자, 우선 필요한 것 중의 하나는 새로운 프로토콜의 모든 측면을 서술하고 필요한 구현 세부사항을 포함하는 공식적인 사양을 산출하는 것이다. 웹 서비스 기술에서 완전한 이익을 얻고 현존하는 네트워크의 일부분인 구성 요소간의 상호연산을 다루기 위해 추가적인 사양을 산출하는 것은 또한 매우 바람직할 것이다.

8.1.1 WSDL 서비스 사양

웹 서비스 정의에 완전히 상응하기 위해서는 서비스 인터페이스와 바인딩의 XML 서술을 가지는 것이 필요하다. 이것은 통상 형식에 맞게 서비스와의 모든 가능한 상호작용을 지정하는 WSDL 문서로 이루어진다. 이론적으로 그와 같은 서술은 서로 다른 서비스간에 상호작용을 자동화하고 더욱 복잡한 특정의 워크플로우(workflow)를 도울 수 있을 것이다. 서비스와 통신하기 위해 WSDL 문서를 취해서 자동으로 특정한 프로그래밍 언어의 코드를 생성하는 많은 수의 도구가 이미 존재한다. 제안된 프로토콜을 서술하는 WSDL 문서는 그러므로 다른 도구뿐만 아니라 네트워크 구성 요소의 개발을 촉진하는 추가적인 수단을 제공할 것이다.

8.1.2 제공자와 메시지 브로커 서비스

분산 데이터를 질의하는 전형적인 네트워크의 일부인 다른 종류의 서비스, 즉, 제공자와 메시지 브로커에 의해 제안되는 프로토콜은 거의 완전한 형태로 사용될 수 있을 것이다. 이러한 서비스는 공식적으로 이 연구에서 다루어지는 않고 이것은 적어도 추가 사양을 필요로 할 것 같다.

분산 질의 아키텍쳐의 전형적인 구성 요소에 의해 사용되는 모든 서비스를 망라하는 일련의 사양을 가지는 것은 분명히 추가적인 도구의 개발을 더욱 쉽게 할 것이다. 더욱이 동일한 프로토콜을 사용함으로써 이러한 도구는 잠재적으로 모든 네트워크에 공유될 수 있을 것이다.

8.2 다른 제안된 도구

새로운 프로토콜에 적응시키기 위해 현재의 코드 기반을 갱신할 때, 또한 기존의 네트워크를 새로운 소프트웨어 배포판을 사용하도록 이전할 때, 몇몇의 추가적인 도구는 매우 큰 가치가 있을 것이다.

8.2.1 서비스 검증자

기존 데이터소스 소프트웨어를 갱신하고 새로운 프로토콜을 따르는 새 구현 시스템을 생산하는 과정동안, 구현시스템이 프로토콜에 일치하는지를 자동으로 체크하고 검증 할 수 있는 외부도구를 가지는 것은 매우 바람직할 것이다. XML 검증은 데이터소스 서비스의 정확한 기능을 확인하기 위한 단지 첫 걸음이다. 유효한 XML 문서내부에서 조차도 필터와 응답 구조에 대해 허위 응답을 생산할 수 있는 많은 가능한 구조물이 있다. 이러한 상황에서 조심스러운 주의력과 더욱 상세한 테스트가 요구될 것이다.

8.2.2 자동 갱신

데이터 제공자 소프트웨어를 자동으로 갱신할 수 있도록 사용될 수 있는 임의의 어떤 도구가 있는지 검사를 하는 것은 또한 가치가 있을 것이다. 또는 심지어 이와 같은 도구를 개발하는 것은 가능할 것이다. 이러한 경우에, 네트워크는 단기간에 모든 데이터 제공자의 소프트웨어를 갱신하는 일정을 잠정적으로 정할 수 있을 것이다. 이 동일한 도구는 과감한 네트워크 이전 기간동안 뿐만 아니라, 종종 발생하는 정기적인 소프트웨어 갱신 기간동안에도 사용될 수 있을 것이다.

8.2.3 프락시 서비스

제안된 프로토콜의 메시지와 기존의 메시지간의 번역이 완전히 가능하지는 않을지라도, 프락시 서비스는 점진적인 네트워크의 이전을 도울 수 있을 것이다. 어느 정도 그리고 어느 경우에 번역 서비스가 서로 다른 프로토콜간에 성공적으로 통신을 중재할 수 있는지를 평가하는 초기 연구를 수행하는 것이 필요할 것이다. 결정에 따라, 프락시 서비스의 개발과 사용은 네트워크를 갱신할 때 중요한 역할을 할 것이다.

8.3 추가적인 연구

새로운 아이디어의 소스일 뿐만 아니라, 통합 과정도안 고려된 몇몇 기술은 더욱 상세한 조사를 할 가치가 있다. 이와 같은 연구의 결론은 프로토콜의 향후 버전을

다른 방향으로 이끌 수 있을 것이다.

8.3.1 XQuark 프로토타입

XQuery 섹션에서 이미 언급한 것처럼, 특정한 도구가 프로토콜을 위한 잠재적인 대안으로서 식별되었다. 생물다양성 네트워크의 필요를 고려하고 여기에서 제안된 것과 비교하여 가능한 장점과 제한점을 더 자세히 평가하기 위해, 프로토타입을 개발하는 것이 필요할 것이다. 다음 항목들에 대해 특별한 주의가 요구된다:

- Xqeury 명령문을 생성할 수 있는 인터페이스를 가지는 가능성을 검토한다.
- 사용되고 있는 맵핑 언어의 제한점과 역량을 확인한다.
- 어떻게 개념 스키마가 정의되는지 확인한다.
- 맵핑 인터페이스를 검토한다.
- 지원되는 데이터베이스 드라이브를 확장하기 위한 작업의 양을 검토한다.
- 대용량 데이터베이스와 성능시험을 수행한다.
- 데이터 제공자가 요청마다 반환되는 데이터의 양을 제한할 수 있는 방법이 있는지 검토한다.

8.3.2 SOAP 프로토타입

전체 네트워크를 SOAP 기술에 기반하는 장점과 제한점에 대한 더 낳은 검토를 위해, 프로토타입 서비스를 개발하는 것이 필요할 것이다. 이미 언급한 것처럼, 이 서비스는 메시지 브로커의 상위 또는 포괄적인 데이터 제공자의 상위의 추가 계층이 될 수 있을 것이다. 다음 항목들에 대해 특별한 주의가 요구된다:

- 특히 필터와 응답 구조와 같은 복합 데이터 타입과 관련한 상호운용성 쟁점사항을 확인하다.
- 기존의 서비스와 비교하여 성능 테스트를 수행한다.

결과에 관계없이 이러한 도구와 기술에 대한 빈번한 연구는 확실히 새로운 생각을 가져올 것이고 현존하는 네트워크의 향상에 기여할 것이다.

8.3.3 온톨로지(Ontologies)

또 다른 광범위한 연구 영역은 개념 스키마를 어떻게 더 잘 표현하는가와 관련 있다. 현재 사용되고 있는 개념 스키마의 몇몇은 자신의 데이터 타입의 성질 정의(property definition)에 제한적이다. 클래스와 관계설정을 포함하는 풍부한 표현은 특히 종 상호작용과 같은 더욱 복잡한 상황을 표현할 필요가 있는 네트워크에, 확실히 더 많은 가능성을 제공할 것이다.

많은 수의 온톨로지 언어가 견고성(robustness)과 표현도(expressivity)의 수준에 따라 이용 가능하다. Semantic Web 아이디어를 가능하도록 하는 많은 것들이 만들어졌고 XML에 기반하고 있다. 더욱 풍부한 맵핑을 가진 더욱 풍부한 개념 표현은 조만간 생물다양성 네트워크에 필요할 것이다.

9 마지막 의견

몇 가지 이유로, DiGIR와 BioCASe 네트워크의 공통 프로토콜의 채택은 우선 사항으로 간주될 수 있다.

- DiGIR와 BioCASe 프로토콜에 의존하는 서로 다른 구현 시스템의 수는, 이미 기존의 코드 베이스의 갱신을 어렵게 만들면서 최근에 상당히 증가하였다. 두개의 상응하는 프로토콜을 생산하는 노력은 보통 가능하지 않기 때문에, 구현 시스템은 현존하는 프로토콜의 하나에 묶이는 경향이 있다. 그러나 모든 소프트웨어 구현 시스템이 모두 상호운용 가능하도록 하는 것은 매우 바람직하다. 상호운용 서버 소프트웨어를 사용하여, 데이터 제공자는 더 많은 가시성(visibility)을 얻을 것이다. 반면에, 완전히 상호운용 가능한 클라이언트 소프트웨어는 더 많은 수의 데이터 제공자를 접근할 수 있을 것이다. 임의의 소프트웨어 구현 시스템은 공통 프로토콜 사용을 선호할 것으로 기대되고, 그러므로 공통 프로토콜이 공동체에 의해 채택되면 계획중인 새로운 구현 시스템은 상당한 개발 시간을 절약할 수 있을 것이다.
- 새로운 주제별 네트워크가 또한 계획되고 구현되고 있다. 공통 프로토콜에 상응하는 이용 가능한 도구가 있으면, 설치된 소프트웨어에 대해 차후 갱신을 피하면서, 이것은 이러한 네트워크에서 즉시 사용될 수 있을 것이다.
- 현재 이미 상당히 많은 데이터 제공자의 수는 정기적으로 계속 증가하고 있다. 도구가 공동 프로토콜을 나중에 지원하고 네트워크가 공동 프로토콜을 나중에 채택하면 할수록, 미래에 이것을 이전하는 것은 더욱 어려울 것이다.

네트워크가 이전 과정을 시작할 수 있도록 제안된 프로토콜은 상대적으로 짧은 기간동안 기존의 소프트웨어에 의해 지원될 수 있을 것이다. 많은 특징이 포함되었을지라도, 이것 중의 대부분은 처음부터 구현할 필요가 없다. 프로토콜과의 일치성을 해치지 않으면서 많은 수준의 기능은 점진적으로 성취될 수 있다.

그러나, 프로토콜이 채택되도록 공통의 의견일치에 도달하기 위해, 공동체와 더 많은 토론이 여전히 필요하다 현재의 형태로 수용 또는 그렇지 않든 간에, 이 연구는 생물다양성 데이터 서비스의 완전한 통합을 위한 또 다른 과정이 되기를 바라는 희망으로 수행되었다.

10 참고문헌

ABCD: Access to Biological Collection Data. Task Group web site:
http://bgbm3.bgbm.fu-berlin.de/TDWG/CODATA/default.htm

BioCASE: Biological Collection Access Service for Europe. Web site:
http://www.biocase.org/
Protocol XML Schema:
http://www.bgbm.org/biodivinf/Schema/protocol 1 3.xsd

CQL: Open Geospatial Consortium, Common Catalogue Query Language, "Filter Encoding Implementation Speci¯cation", Panagiotis A. Vretanos, version 1.0.0, Adopted Speci¯cation, 19 September 2001.
http://www.opengis.org/docs/02-059.pdf

DarwinCore: A set of data element de¯nitions designed to search primary biodiversity data. Task Group web site:
http://darwincore.calacademy.org

DiGIR: Distributed Generic Information Retrieval. Web site:
http://digir.net
Protocol XML Schema:
http://digir.net/schema/protocol/2003/1.0/digir.xsd

GML: Open Geospatial Consortium, "Geography Markup Language Implementation Speci¯cation", Simon Cox, Paul Daisey, Ron Lake, Clemens Portele, Arliss Whiteside, version 3.00, 29 January 2003.
http://www.opengis.org/docs/02-023r4.pdf

SDD: Structure of Descriptive Data. Task Group web site:
http://160.45.63.11/Projects/TDWG-SDD/

Semantic Web: World Wide Web Consortium, "Semantic Web Activity".
http://www.w3.org/2001/sw/

SOAP: World Wide Web Consortium, "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, 24 June 2003.
http://www.w3.org/TR/soap12-part1/

TCS: Taxonomic Concept Transfer Schema. Task Group web site:
http://www.soc.napier.ac.uk/tdwg/index.php

UDDI: OASIS Standards Consortium, "Universal Description, Discovery, and Integration". Web site:
http://www.uddi.org/

WFS: Open Geospatial Consortium, "Web Feature Service Implementation Speci¯cation", Panagiotis A. Vretanos, version 1.0.0, Adopted Speci¯cation, 96
19 September 2002.
http://www.opengis.org/docs/02-058.pdf

WSDL: World Wide Web Consortium, "Web Services Description Language (WSDL) 1.1", W3C Note 15 March 2001.
http://www.w3.org/TR/wsdl.html

XML: World WideWeb Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", W3C Recommendation, 04 February 2004.
http://www.w3c.org/TR/REC-xml

XML Schema: World Wide Web Consortium, "XML Schema speci¯cation (Part0: Primer, Part1: Structures, and Part2: Datatypes)", W3C Recommendation, May 2001.
http://www.w3.org/XML/Activity.html#schema-wg

XPath: World Wide Web Consortium, "XML Path Language (XPath) Version 1.0", W3C Recommendation, 16 November 1999.
http://www.w3.org/TR/xpath

XQuery: World Wide Web Consortium, "XQuery: A Query Language for XML", W3C Working Draft, 23 July 2004.
http://www.w3c.org/TR/xquery/