ISBN 978-89-6211-485-0 93500

# 생명정보 분석 모델링을 위한 워크플로우 기반 서비스 프레임워크

한 영 만

hans@kisti.re.kr

한국과학기술정보연구원

Korea Institute of Science and Technology Information

# 목 차

1. 개요 ·····1

2. 워크플로우 기반의 생명정보 분석과정 모델링 ·····5

2.1. 생명정보 분석과정의 특징과 워크플로우 ·····5

2.2. 국내외 사례 분석 ·····6

3. 워크플로우 기반 생명정보 서비스 프레임워크 구현방법 ·····12

3.1. 웹서비시스 기반 클라이언트-서버 아키텍처 ·····12

3.2. 온톨로지 기반 이질적 생명정보 데이터 통합 ·····13

3.3. XML 기반 생명정보 분석 도구 통합 ·····14

3.4. 워크플로우 실행 엔진 모델 ·····17

3.5. 슈퍼컴퓨팅 연계를 통한 대규모 분석처리 지원 ·····19

3.6. 클라이언트-서버 간 데이터 동기화 최적화 ·····23

3.7. 협력연구를 위한 워크플로우 공유 환경 구현 ·····25

4. 결론 ·····26

# 표 차례

표 1. 비즈니스 워크플로우와 생명정보 워크플로우와의 차이점 ·····6

표 2. 생명과학 분야에서의 XML 정의 스키마 ·····16

표 3. 생명정보 분석 워크플로우의 세부 클래스 모델 세부사항 ·····18

표 4 Continuation 기반 Comet 기술 적용시 서버 자원 소요량 ·····24

# 그림 차례

그림 1. NCBI GenBank 데이터 증가추세 ·····1
그림 2. Francis Collins가 제시한 향후 Genomics의 발전 방향 ·····2
그림 3. 2005년 까지 전 세계 생명공학 시장 규모 ·····2
그림 4. 2010년까지의 전 세계 생명정보학 시장규모 ·····3
그림 5. Biological Workflow for Promoter Identification ·····4
그림 6. 생명정보 분석 과정에서의 협력 연구 행위 흐름도 ·····5
그림 7. Taverna 구성도 ·····7
그림 8. myExperiment.org 포털 메인 화면 ·····7
그림 9 BioWBI-WEE 시스템 아키텍처 ·····8
그림 10. DAS를 이용한 분산 주석 개념도 ·····8
그림 11. GEL을 통한 Wildfire 서버 처리 작업 개념도 ·····9
그림 12. Pegasys 클라이언트/서버 아키텍처 ·····9
그림 13. BioWMS 전체 실행 개념도 ·····11
그림 14. Biowep 시스템 아키텍처 ·····11
그림 15. 생명정보 서비스 프레임워크 구현을 위한 웹서비시스 기반 클라이언트-서버 아키텍처 ·····12
그림 16. 생명정보 분석 도구 입출력 양식을 위한 OWL 파일 ·····14
그림 17 Jena API를 통한 생명정보 데이터 OWL 파싱 및 트리뷰 ·····14
그림 18. 생명정보 분석도구 클래스 다이어그램 ·····15
그림 19. 생명정보 분석도구에 대한 XML 정의 예 ·····17
그림 20. 생명정보 분석 워크플로우 실행개념도 ·····18
그림 21. 생명정보 분석 워크플로우 클래스 계층 모델 ·····18

그림 22. 워크플로우 클래스 모델에 대한 Hibernate OR 맵핑 예 ……20
그림 23. 세포 사이즈 증가에 따른 컴퓨팅 자원 요구량 ……20
그림 24. 슈퍼컴퓨팅 환경에서의 워크플로우 실행 개념도 ……21
그림 25. Jetty Comet 기반 생명정보 워크플로우 데이터 동기화 실행모델 25

## 1. 개요

'인간 게놈 프로젝트'가 성공적으로 진행된 이래로, 최근의 자동 DNA 시퀀서, DNA 마이크로어레이, 이미지 분석기 등과 같은 고속 분석 기기들의 발달은 대량으로 생물학적 데이터를 기하급수적인 증가를 가져왔다. <그림 1>은 미국 NCBI의 GenBank 데이터 증가 추세에 대한 통계 자료로 인간 게놈 프로젝트 이후로 등록되는 유전자 데이터의 양이 기하급수적으로 증가하고 있는 것을 볼 수 있다. 이러한 폭발적인 데이터의 생산은 생명과학 연구의 패러다임 전환을 가져왔다. 즉, 매일 발견되는 대용량의 생물학적 데이터를 분석하여 의미 있는 생물학적 정보를 찾아내는 것이 중요한 경쟁력이 된 것이다. IT 기술의 발달과 더불어 전통적으로 실험실에서 행해지던 많은 일들이 지식 기반으로 컴퓨터상에서 이루어 질 수 있게 되었다. 예를 들면, DNA의 어떤 부분이 생명의 다양한 화학 작용을 제어하는가? 새로운 단백질의 기능은 무엇인가? 새로운 단백질의 구조를 예측 할 수 있는가? 등의 질문에 대해서 생명정보학은 잠재적으로 의미 있는 답을 줄 수 있다. 생명정보학이 현재 응용되는 생물학 분야는 유전체학, 전사체학, 단백질체학, 대사체학, 약리유전체학 등 분자생물학의 모든 분야를 포함하고 있다.

미국의 국립인간게놈연구소(NHGRI)의 소장이었던 Francis Collins가 제시한 향후 유전체학의 발전방향에 대한 삽화(<그림 2>)에서 보는 바와 같이 생명정보학 분야

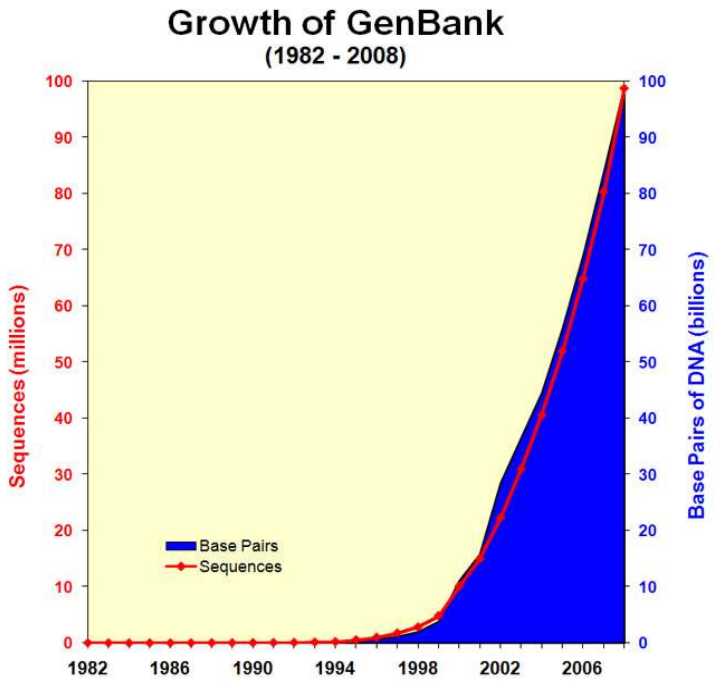


그림 1. NCBI GenBank 데이터 증가추세,
http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

는 게놈 유전체 학에서의 하나의 큰 기둥으로서의 역할을 담당하며 다른 연구 분야와의 유기적 협력관계를 유지할 것으로 예상된다. 또한 <그림 3>에서 예시한 바와 같이 2005년 전 세계 생명공학 시장의 총매출은 1,263억 달러로 2001년에서 2005년까지 5년 동안 연평균 성장률이 12.8%로 지속적인 성장세를 보이고 있으며 2010년까지 생명 정보 연구 분야의 폭발적인 시장 수요 증가가 현실화 되고 있는 시점에 있다(<그림 4>). 또한, IT 기술의 발전과 생명정보 데이터 증가 추세와 발맞추어 생명정보 분석과 관련한 다양한 프로그램들이 기하급수적으로 증가하는 추세에 있다.

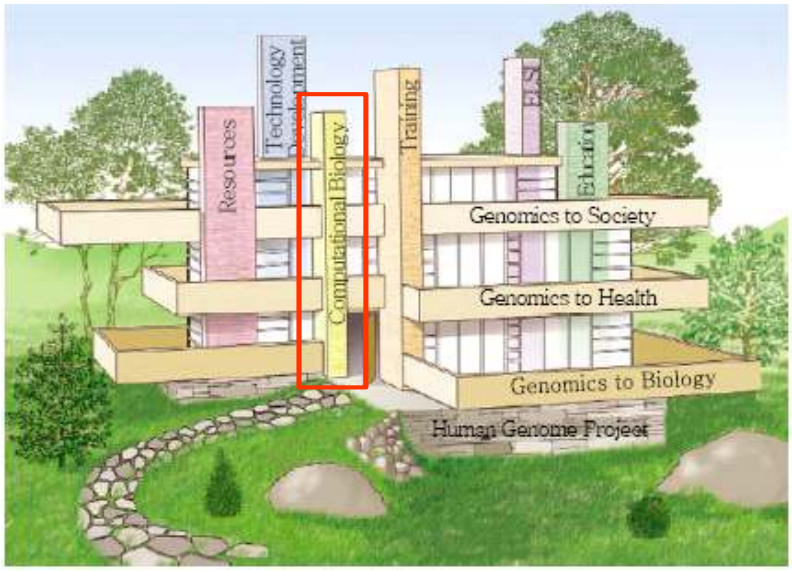


그림 2. Francis Collins가 제시한 향후 Genomics의 발전 방향

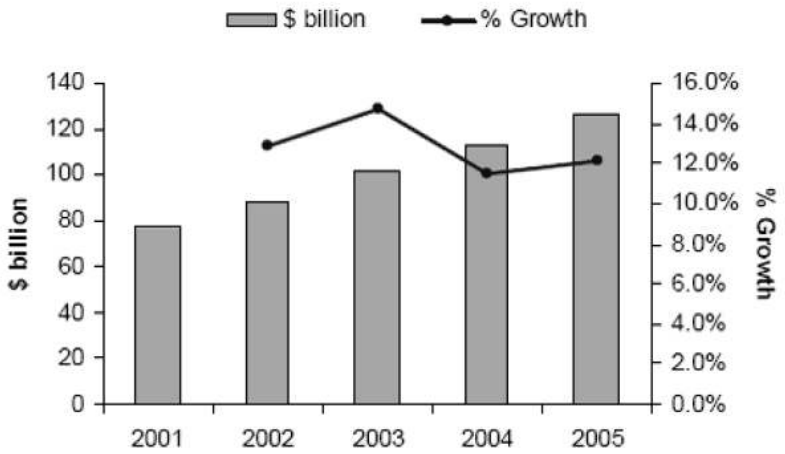


그림 3. 2005년 까지 전 세계 생명공학 시장 규모,Global Biotechnology(2006. 8), Datamonitor

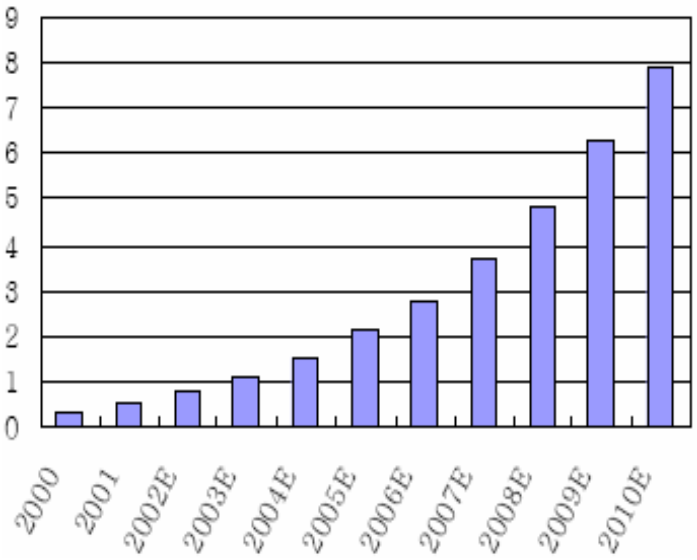


그림 4. 2010년까지의 전 세계 생명정보학 시장규모($US Billon), "Bioinformatics Opportunities," Digital Vector Market Report, 2004.

이러한 생명정보 분석 기술의 발전에도 불구하고, 생명과학 연구자들이 자신의 연구에 적합한 기술을 활용하기에는 몇 가지 문제점이 있다.

첫째, 생명정보 데이터 및 도구의 이질성이다. 현재까지 국내는 물론 전 세계 생명 과학 분야 기업이나 연구 기관들은 연구 결과로부터 얻어진 생물 정보 데이터를 각기 다른 독자적인 포맷으로 저장되고 배포되어 왔으며 생명 과학 연구에 필요한 분석 도구들도 역시 각자의 프로그래밍 언어와 개발 환경을 기반으로 개발된 것이 현실이다. 이러한 까닭에 실로 다양하고 이질적인 생명정보 분석도구들이 각기 다른 입출력 포맷과 사용자 인터페이스를 갖고 분산되어 있어, 연구자들은 자신의 연구 환경에 맞는 분석도구를 선택하고 그것을 활용하는 데 있어 많은 어려움을 겪게 된다. 특히 IT 기술이 부족한 생명과학 연구자들이 각기 다른 생명정보 분석도구의 입출력 방식과 사용자 인터페이스를 익히는 것은 매우 어려운 일이다. 이것이 생명과학 연구자들이 생명정보 분석기술의 유용함에도 불구하고 그것의 사용을 꺼리는 가장 큰 요인 중의 하나이다. 따라서 이질적이고 분산된 생명정보 분석도구들에 대한 직관적이고 일관성 있는 통합 사용자 인터페이스가 절실히 요구된다.

둘째 여러 분석 도구 간의 입출력 연계가 어렵다는 것이다. <그림 5>에서 보이는 바대로 일반적인 생명정보 분석 과정은 여러 생물 정보 데이터베이스를 검색하여 다양한 정보를 축출하고 이에 대한 다양한 분석도구의 적용 및 결과물 분석 등의 여러 단계의 단위 분석 도구의 입출력 연계로 이루어진다. 즉, 레고 블록을 조립하

는 것과 같이 각각의 단위 분석 도구들을 끼워 맞추어 자신이 원하는 기능을 발휘하는 완성품을 만들어 내는 과정인 것이다. 앞서 제기한 바대로 대부분의 분석도구들이 가가기 다른 입출력 포맷을 갖고 있어 분석 도구 간의 입출력 연계는 매우 어려운 일이다. 입출력 연계를 위한 데이터 변환 프로그램을 별도로 작성하면 되겠지만, 이것은 전체 분석과정의 결과물의 생물학적 분석이 중요한 생명과학 연구자에게는 별도의 부담스러운 일이다.

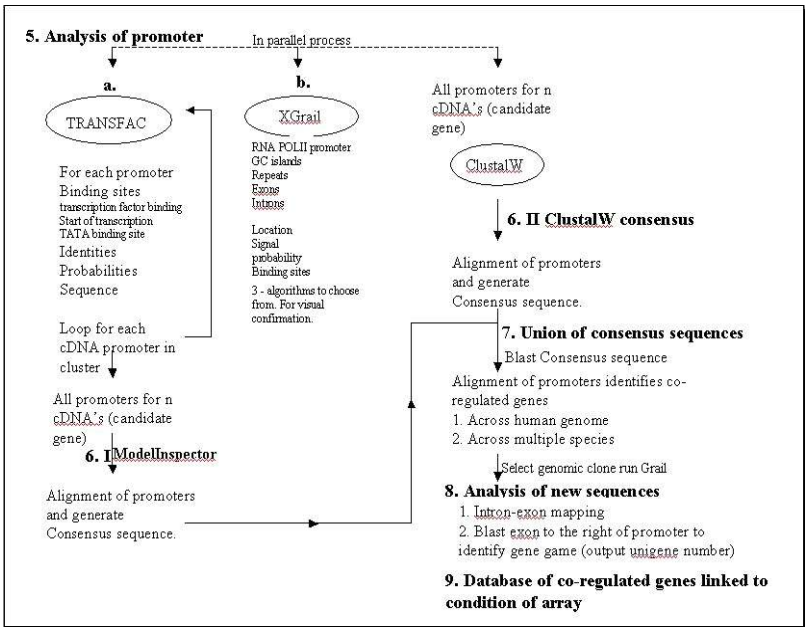


그림 5. Biological Workflow for Promoter Identification

셋째, 생명과학 연구자, IT 개발자, 생명정보 연구자 들 간의 협력 연구 환경이 미흡하다는 것이다. 매일 발견되는 폭발적인 생명정보 데이터로부터 생물학적으로 의미있는 정보를 분석하기 위해서는 생명과학 연구자, IT 개발자, 생명정보 연구자 들 간의 협력 연구가 매우 중요하다. <그림 6>은 생물학적 질문에 대한 일반적인 생명정보 분석 과정에서의 각 분야 연구자들의 역할 또는 행위들에 대한 흐름도이다. 이렇듯 생물학적으로 중요한 질문에 대한 잠재적인 답을 생명정보 분석 과정을 통해 효과적으로 구해내기 위해서는 각 분야 연구자 간의 유기적 협력이 매우 중요하며, 그것을 위한 일관성 있고 통합된 협업 연구 환경이 절실히 요구된다.

넷째, 대용량•대규모 분석 환경이 미흡하다는 것이다. 앞서 예시한 바와 같이 '인간 게놈 프로젝트'가 성공적으로 진행된 이래로 생명정보 데이터는 폭발적으로 증가해 왔으며 그러한 대용량 데이터를 누가 더 빠르고 정확하게 분석하여 생물학적으로 의미 있는 정보를 유추해내는 것이 가장 중요한 일이다. 게다가 최근 선진국들을 중심으로 빠르게 발달하고 있는 첨단 생명과학 연구는 기존의 소규모 연구방

식에서 벗어나서 점차적으로 대용량의 생물학 정보들을 대상으로 복잡한 계산과정을 요구하기 때문에 고도의 컴퓨팅 인프라를 필요로 하게 되었다. 특히, 암과 같은

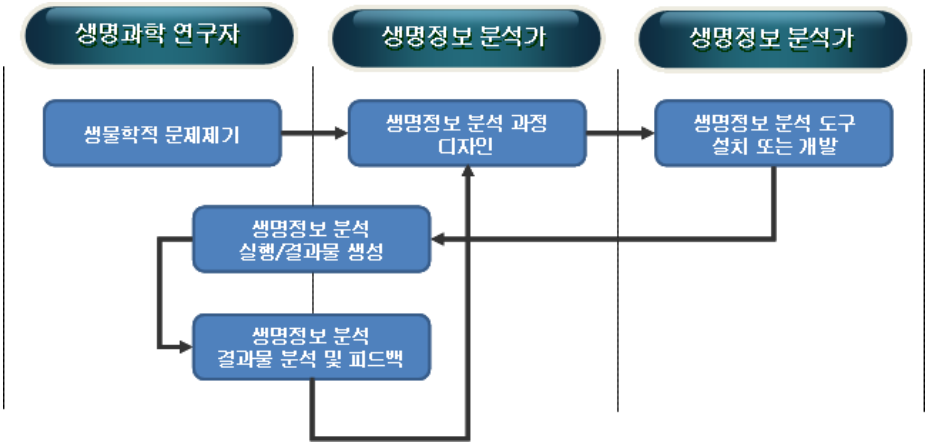


그림 6. 생명정보 분석 과정에서의 협력 연구 행위 흐름도

다인자 유발 질병의 경우, 기존의 단편적인 생명과학 연구방법으로는 한계가 있으므로 세포내의 전체적인 요인들에 대한 통합적인 분석법이 요구되며, 이를 위해 현재까지 밝혀진 방대한 데이터를 통합하고 이를 대규모 컴퓨팅자원을 활용하여 분석할 수 있는 슈퍼컴퓨팅 인프라 환경이 필요하다. 그러나 이와 같은 조건에 부합하는 전산자원들은 하드웨어 자체의 단가가 매우 높을 뿐 아니라, 이를 효과적으로 관리하고 활용할 수 있는 전문 인력을 요구하기 때문에 개별적인 연구실 단위로는 제대로 된 연구기반을 갖추는 것이 현실적으로 불가능한 실정이다.

본 연구 보고서에서는 생명정보 분석 기술의 활용에 있어서의 이러한 문제점들을 해결하기 위하여 이질적인 생명정보 데이터와 도구를 확장적으로 통합하고 생명정보 분석과정을 효과적으로 모델링하고 실행할 수 있는 워크플로우 기반의 서비스 프레임워크의 구현 방법 및 세부 기술 요소에 대해 제시하고자 한다.

# 1. 워크플로우 기반의 생명정보 분석과정 모델링

## 2.1. 생명정보 분석과정의 특징과 워크플로우

최근 선진국 들을 중심으로 워크플로우를 기반으로 한 생물정보 서비스 통합이 활발히 이루어지고 있다. 왜 워크플로우 기반 통합인가? 이에 대한 대답은 의외로 간단하다. 생물정보 분석 과정이 하나의 워크플로우이기 때문이다. 앞서 Promoter 식별을 위한 <그림 5>에서 보인 바와 같이 일반적인 생물정보 분석 과정은 여러 생물 정보 데이터베이스를 검색하여 다양한 정보를 축출하고 이에 대한 다양한 분석도구의 적용 및 결과물 분석 등의 여러 단계의 단위 분석 업무의 워크플로우 형태로 수행된다. 즉, 레고 블록을 조립하는 것과 같이 각각의 단위 분석 도구(조립부

품)들을 끼워 맞추어 - 만약 입출력 연계가 맞지 않는 다면 두 도구를 연결해주는 새로운 이음쇠를 직접 만들어 계속해서 연결하면서 - 자신이 원하는 기능을 발휘하는 완성품(워크플로우)을 만들어 내는 과정인 것이다.

Workflow라는 단어가 'Work + Flow'로 구성되어 있는 것처럼 워크플로우 모델링 구현에 있어 중심 관점은 대상 도메인에 따라 달라지며 특히 생명공학 분야에서의 워크플로우는 가설과 검증, 그리고 중간 단계에서의 결과물의 분석 등이 중요하기 때문에 일반적인 비즈니스 워크플로우와는 다른 특징들을 갖고 있다. 이를 <표 1>에 정리하였다. 4장에서는 <표 1>에서 제시한 생명정보 분석 워크플로우의 특징을 고려하여 워크플로우 기반 생명정보 분석 서비스 프레임워크의 실제적인 구현방법에 대해 살펴보도록 한다.

| 구 분 | 비즈니스 워크플로우 | 생명정보 워크플로우 |
|---|---|---|
| 중심대상 | • Transaction<br>• Process & state | • Problem solving<br>• Knowledge |
| 개방성 | • 파트너 간 폐쇄적 | • 개방적 커뮤니티를 통한 과학적 데이터 교류 |
| 검증과 오류처리 | • 상호 간의 협의된 정책에 의해 일관된 방식 | • Learning from mistakes<br>• 중간 결과물에 대한 검증이 중요 |
| 재사용 | • 중요하지 않음 | • 재사용이 중요 |
| 유연성 | • Workflow는 좀처럼 변경되지 않음 | • Based on experiments<br>• Rapid & flexible |
| 데이터 유형 | • Small amounts of data<br>• 정형화된 타입 | • Large amounts of data<br>• 다양하고 비정형화된 데이터 타입과 포맷 |
| 흐름유형 | • Data와 Control 흐름이 분리되어 진행<br>• Process 흐름에서의 객체 상태가 중요 | • Data와 Control 흐름이 통합 되어<br>• 진행<br>• Process 흐름에서의 데이터 값이 중요 |

표 1. 비즈니스 워크플로우와 생명정보 워크플로우와의 차이점

## 2.2. 국내외 사례 분석

### 2.2.1. Taverna

Taverna[1]는 EPSRC(Engineering and Physical Sciences Research Council)에서 추진하는 myGrid 프로젝트의 한 부분으로서 다양한 생명정보 분석 서비스를 워크플로우기반으로 통합하여 실행 및 관리를 할 수 있도록 하는 도구로서 <그림 7>에서 보는 바와 같이, 다양한 생명정보 서비스를 구성하기 위한 SCUFL(Simple

Conceptual Unified Flow Language) 워크플로우 정의와 특정 SCUFL 워크플로우를 생성하고 편집하기 위한 Workbench, 그리고 생성 된 워크플로우를 실행하고 단계별 결과를 반환하는 워크플로우 실행 엔진으로 구성된다. Taverna 워크벤치는 2009년 11월 18일 현재 2.1 베타 2 버전까지 릴리즈 된 상태에 있으며, 다양한 형태의 웹서비시스, Java 동적 스크립트, R 스크립트 등을 실행할 수 있는 유용한 기능들이 추가 개발되고 있는 상태에 있다. 특히, 비교적 최근 오픈하여 서비스 중에 있는 myExperiment(http://www.myexperiment.org) 포털(<그림 8>)은 Taverna에서 작성된 생명정보 워크플로우를 사용자들 간에 공유하고 활용할 수 있도록 하는 유용한 기능들을 제공하고 있다.

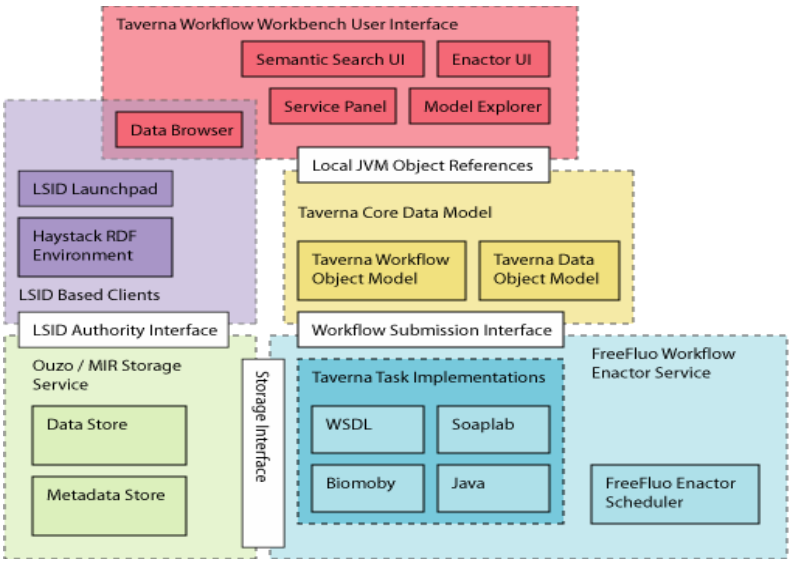


그림 7. Taverna 구성도

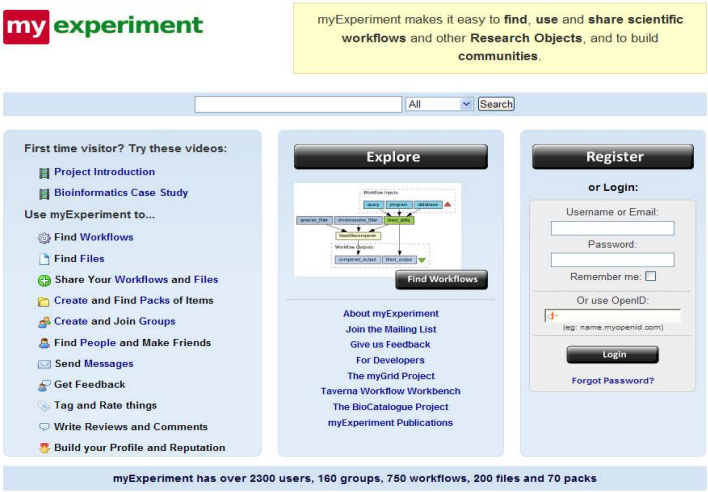


그림 8. myExperiment.org 포털 메인 화면

### 2.2.2. BioWBI-WEE

BioWBI-WEE[2]은 미국의 대규모 IT 솔루션 업체인 IBM에서 개발한 Web Services 기반의 생명정보 워크플로우 분석 도구로서, <그림 9>에서 예시한 바와 같이 워크플로우를 모델링하기 위한 웹 기반의 BioWBI(Bioinformatics Workflow Builder Interface)와 생성된 워크플로우를 실행하고 그 결과물을 반환하고 관리하는 WEE(Workflow Execution Engine)으로 구성된다.

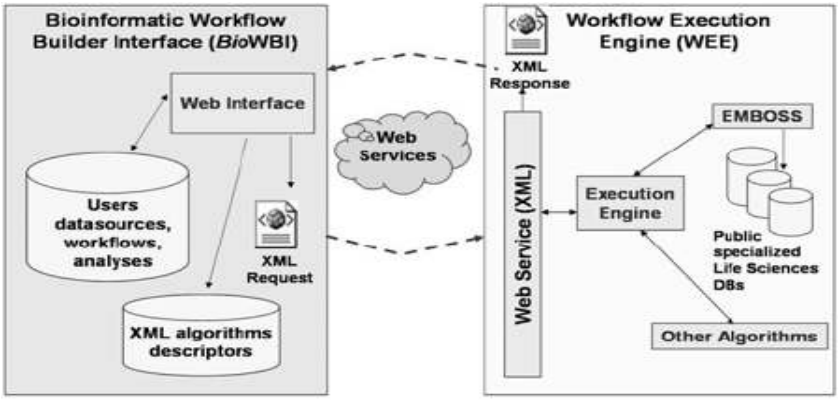


그림 9 BioWBI-WEE 시스템 아키텍처

### 2.2.3. DAS

DAS(Distributed Annotation System)[3]는 Cold Spring Harbor 연구소의 Lincoln D. Stein 등이 제안한 분산 형 유전자 주석 모델로서 다양한 서비스를 제공하지는 않지만 유전자에 대한 주석 정보를 웹서비스 기능을 이용하여 통합 제공하고 있으며 <그림 10>에서 예시한 바와 같이 지역적으로 분산된 여러 그룹의 연구자들이 하나의 유전자 서열을 보면서 이에 대한 주석을 나름대로 입력할 수 있도록 하는

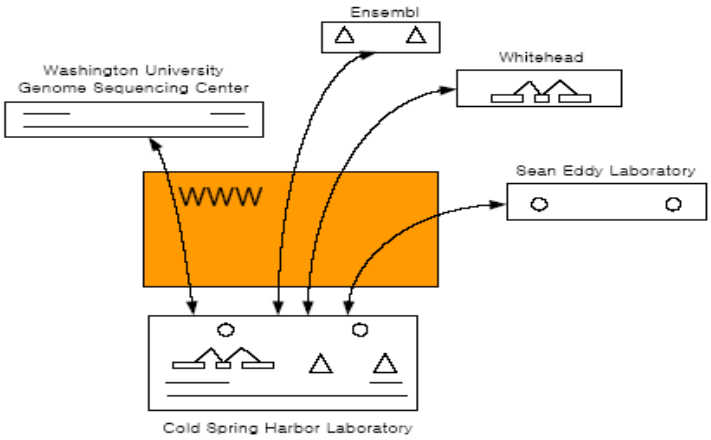


그림 10. DAS를 이용한 분산 주석 개념도

분산 표준 환경을 제공한다.

### 2.2.4. Wildfire

Wildfire[4]는 싱가폴의 A*STAR 산하 바이오인포매틱스 연구기관에서 2005년도에 발표한 생명정보 워크플로우 관리 시스템으로서 클라이언트/서버 아키텍처로 구성되어 있다. 클라이언트 프로그램은 주로 EMBOSS[5] 프로그램을 구성요소로 하여 생명정보 워크플로우를 시각적으로 모델링 할 수 있도록 한다. 사용자가 클라이언트 프로그램을 통하여 워크플로우를 구성하여 실행하면 서버에서는 해당 워크플로우에 대한 GEL(Grid Execution Language)[6]언어로 작성된 스크립트를 생성하고 이것을 스케즐러를 통해 수행하게 된다. 작업 스케즐러는 Condor Grids, SGE, PBS, 그리고 LSF와 같은 배치큐 시스템과 연계하여 각각의 워크플로우 구성 컴포넌트에 해당하는 작업을 클러스터 환경에서 분산 병렬 처리 할 수 있도록 한다. Wildfire는 워크플로우의 실행을 분산 컴퓨팅 환경에서 처리하기 때문에 대용량 분석 작업을 수행 할 수 있다는 장점이 있다. <그림 11>은 GEL을 통한 Wildfire 서버 처리 작업에 대한 개념도이다.

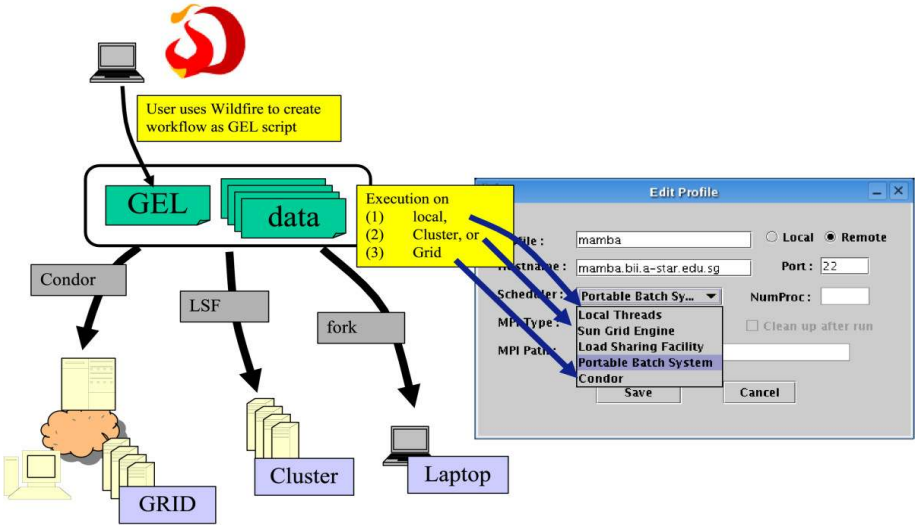


그림 11. GEL을 통한 Wildfire 서버 처리 작업 개념도

### 2.2.5. Pegasys

Pegasys[7]는 2004년도에 캐나다 UBC 센터에서 발표한 생명정보 워크플로우 시스템으로서 <그림 12>에서 보는 바와 같이 클라이언트/서버 아키텍처로 구성되어 있다. 클라이언트 프로그램은 주어진 서열의 생명정보 분석 과정을 워크플로우 형태로 모델링하기 위한 사용자 인터페이스를 제공한다. 생성된 워크플로우는 서버로 보내져 스케즐러를 통해 클러스터 컴퓨팅 노드 위에서 실행되고 결과물이 수집된다. 수집된 결과물은 입력 서열과 연관되어 데이터베이스에 저장된다. 저장된 데이터는 GFF, GAME XML 등의 형식으로 다운로드가 가능하다.

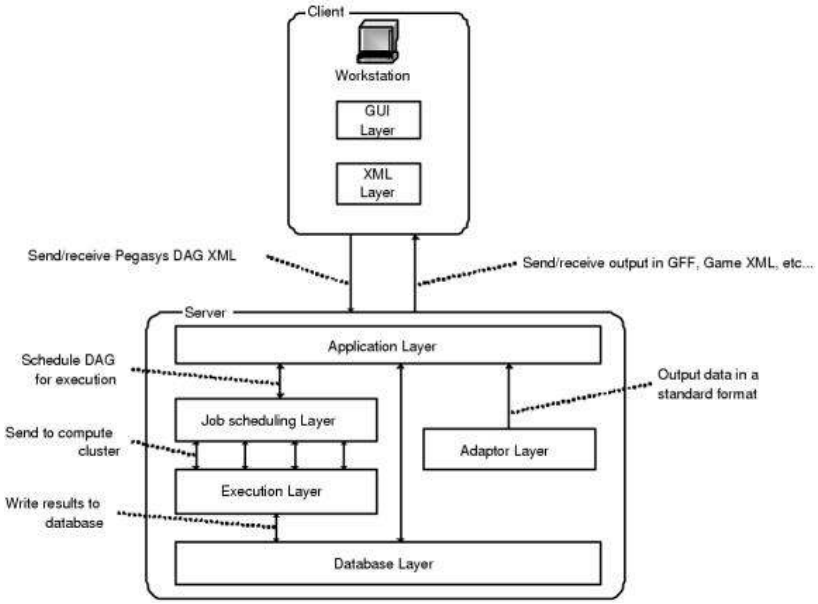


그림 12. Pegasys 클라이언트/서버 아키텍처

### 2.2.6. BioWMS

BioWMS[8]는 2007년도 이탈리아 Camerino 대학에서 발표한 웹 기반의 생명정보 워크플로우 시스템으로서 대행자 기반 미들웨어로 구현되어 졌다. 사용자가 웹 기반의 WebWFlow 에디터를 통하여 워크플로우 모델을 작성하면 XpdlCompiler를 통해 WfMC의 표준 중 하나인 XPDL 형태로 변환되게 된다. 생성된 XPDL은 서버의 Hermes 워크플로우 엔진을 통해 정의된 프로세스를 수행하게 된다. 관리자는 Hermes GUI를 통해 워크플로우 내 프로세스의 실행현황을 모니터링 할 수 있도록 구성되어 있다. <그림 13>은 BioWMS의 전체 실행 개념도이다.

### 2.2.7. Biowep

Biowep[9]는 2006년 이탈리아 Genoa의 국립암연구센터에서 발표한 것으로 다양한 생명정보 분석 도구로 구성된 워크플로우를 웹 상에서 선택•실행하여 결과물을 얻을 수 있도록 하는 시스템이다. <그림 14>에서 보는 바와 같이 Biowep 시스템 아키텍처는 Taverna 워크벤치 또는 BioWMS를 사용하여 하나의 워크플로우 모델을 생성하기 위한 Workflow Manager, 사용자 계정을 관리하고 사용자 별 워크플로우와 실행 결과물을 검색하고 선택할 수 있도록 하는 User Interface, 그리고 선택된 워크플로우를 Taverna FreeFuo 또는 BioWMS의 Hermes 상에서 실행하고 제어할 수 있도록 하는 Workflow Executor로 구성된다.

### 2.2.8. 국내 현황 및 시사점

앞서 제시한 바와 같이 국외의 경우 워크플로우 기반의 생물정보 서비스 통합에 관련된 연

구 및 서비스가 매우 활발히 이루어지고 있다. 특히 Taverna의 경우 유럽의 BT 분야 e-Science

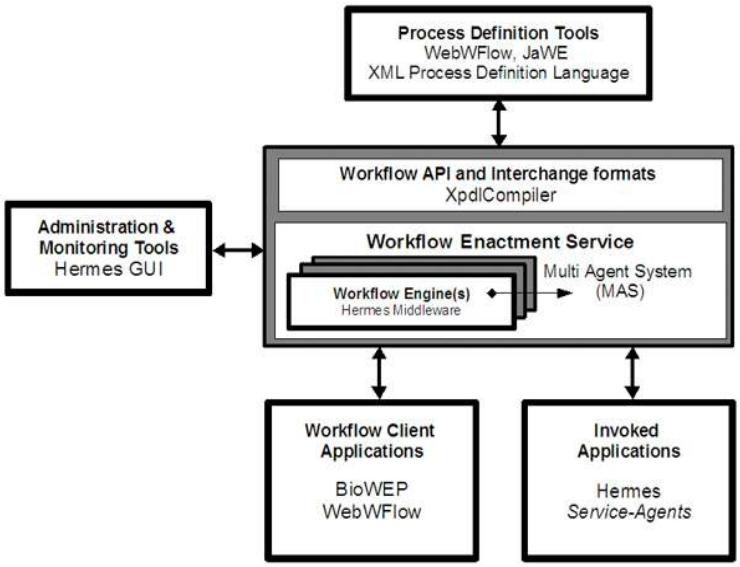


그림 13. BioWMS 전체 실행 개념도

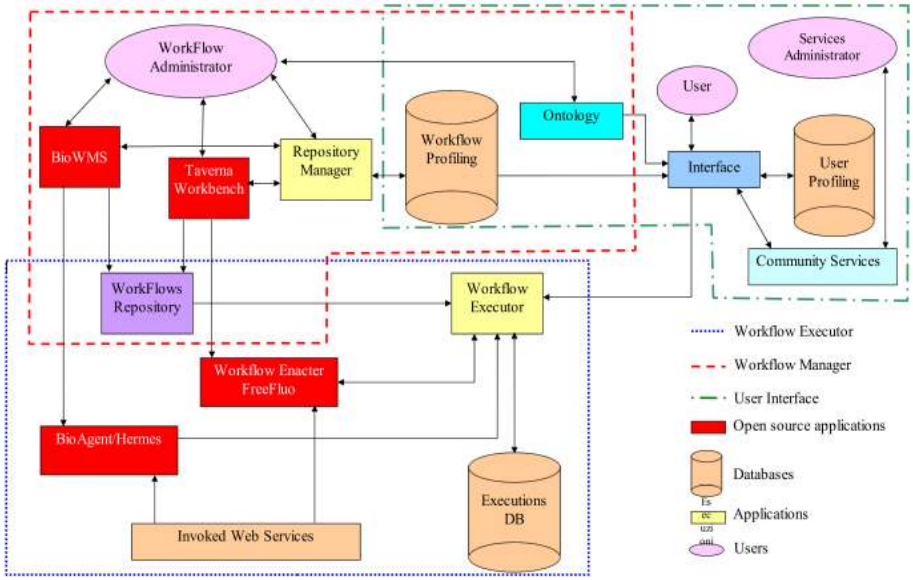


그림 14. Biowep 시스템 아키텍처

환경 구축에 있어 핵심적인 역할을 담당해내고 있다. 국내의 경우, 한국생명공학연구원 국가생물자원정보관리센터는 2007년 9월 생물학 연구자들이 가장 많이 사용하는 대표적인 생물정보학 분석도구를 통합하여 웹에서 자동적으로 분석 가능한 Biopipe[10] 베타 버전을 공개하였다. BioPipe 베타버전은 이용자가 필요한 분석도구를 검색하고, 마우스로 분석 파이프라인을 연속적으로 설계할 수 있도록 하여, 실행할 경우 설계된 절차에 따라 자동으로 분석이 가능하도록 하여 보다 손쉽게 연구자들이 원하는 데이터를 분석할 수 있도록 개발되었다. 웹 기반의 소프

트웨어로 개발된 BioPipe는 베타 서비스 단계로 웹 인터페이스를 통한 워크플로우 편집은 아직까지는 불편한 상태에 있고, 지속적인 사용자 인터페이스 개선 작업이 진행 중에 있다. 그럼에도 불구하고, 국내 대부분의 생물정보 프로그램들은 각각의 연구 목적에 따라 중복 개발되어 사용되고 있으며 통합적인 워크플로우 플랫폼의 개발은 타 선진국에 비해 거의 전무한 실정이다. 타 산업의 경우 해당 업무 효율성 극대화를 위해 다양한 워크플로우 시스템들이 개발되고 적용되는 것에 반하여 생물정보 연구 분야에 대한 국내 IT 업체들의 인식이 낮기 때문에, BT에 대한 높은 관심에도 불구하고 IT 인프라와의 연계가 제대로 진행되지 않는 형편에 있다.

# 2. 워크플로우 기반 생명정보 서비스 프레임워크 구현방법

## 2.1. 웹서비시스 기반 클라이언트-서버 아키텍처

대용량의 생명정보 데이터를 처리하고 시•공간의 제약 없이 워크플로우의 실행 현황을 모니터링 하기 위해서는 클라이언트-서버 아키텍처가 적합하다. 앞서 제시한 사례에서 보듯이 Taverna를 제외한 대분의 생물정보 워크플로우 시스템 서비스는 대용량 데이터 처리 및 효율적인 자료 관리를 위해 클라이언트-서버 아키텍처를 구현하고 있다. 다양한 사용자 클라이언트 환경을 지원하기 위해 클라이언트-서버 간의 데이터 객체 송수신은 웹서비시스[11]를 통해 이루어지도록 설계한다. 웹서비시스는 네트워크상의 애플리케이션 서비스들을 원활하게 연결하기 위한 오픈 아키텍처 프로토콜 표준으로서 이미 많은 생물정보 연구와 관련한 서비스 기관(또는 벤더)에서 웹서비시스를 통한 생물정보 관련 데이터베이스 및 서비스를 제공하고 있다. <그림 15>는 웹서비시스를 기반으로 한 생명정보 서비스 프레임워크의 클라이언트-서버 아키텍처이다.

## 2.2. 온톨로지 기반 이질적 생명정보 데이터 통합

온톨로지(Ontology)는 전산학 혹은 인공지능 분야에서 1970년대 후반 내지 1980년대 초반에서 사용죄기 시작하였고 그 이후 많은 개념상의 혼란이 있었다. 이에 1993년 Gruber가 “온톨로지는 어떤 관심분야에서의 개념의 정형화된 명세화이다(An ontology is a an explicit and formal specification of a conceptualization of a domain of interest)”[12]라는 말로 정의하였고, 이후 많은 학문 분야에서 받아들여져 응용되고 있다. 온톨로지에서의 지식은 클래스(class), 관계(relation), 함수(function), 공리(axion), 인스턴스(instance)의 다섯 가지 요소를 이용하여 형식화된다. 클래스는 일반적으로 개념어에 해당되며, 관계는 개념들을 규정하는 속성의 유형을 의미한다. 함수는 관계가 특정 값을 가질 때 성립되는 것이며, 공리는 논리의 전개나 추론의 근거가 되는 것으로 참으로 인정되는 문장을 의미한다. 그리고 인스턴스는 이와 같은 요소들이 결합되어진 실제의 값을 의미한다[13]. 온톨로지를 표현하기 위한 사용되는 표준 규약으로는 RDF, OWL, SWRL 등이 있다. RDF[14]는

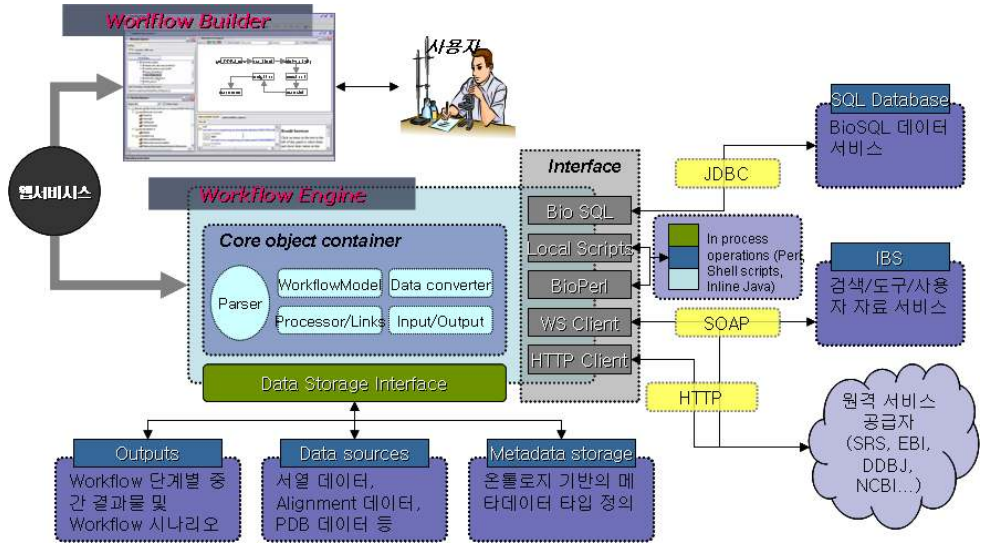


그림 15. 생명정보 서비스 프레임워크 구현을 위한 웹서비시스 기반 클라이언트-서버 아키텍처

XML에서 발전한 형태이며, subject, object, predicate으로 이루어지며, 단순하게 개념 혹은 인스턴스 사이의 관계를 나타낸다. 일반적으로 복잡한 제약조건이 필요 없는 일반 응용 분야에서 RDF를 많이 사용한다. OWL[15]은 관계들 간의 계층적 구조, 관계 인스턴스 내에서의 논리적 제약조건 등을 포함한 언어이다. 정밀하고 논리적인 추론을 필요로 하는 경우에 사용한다. SWRL[16]은 추론을 위한 규칙을 정의하기 위하여 사용한다.

생명정보학 분야에서의 온톨로지는 다양한 생물학적 의미들을 표현하는 구조로 되어 있으며, 생물학 데이터의 의미를 효과적으로 해석할 수 있는 매우 중요한 기술로 인식되어 많은 세부 도메인에서 개발되고 사용되고 있다. 가장 대표적인 예로 유전자 온톨로지(Gene Ontology)[17]가 있다. 유전자 온톨로지는 유전자의 위치, 유전자와 단백질에서 수반된 생물학적 기능이나 프로세스 등에 대한 온톨로지 표현 규약이다. EcoCyc 온톨로지[18]는 분자생물학과 생명정보 분야에서 이질적 정보 통합을 위해 정의되었는데 TAMBIS[19] 프로젝트에서 사용되어 지고 있다. 그 외에도 MGED(Microarray Gene Expression Data)[20] 온톨로지는 마이크로어레이 유전자 발현 실험 및 데이터와 관련된 명세를 규약하고, BioPAX[21]는 신호전달 경로에 대한 온톨로지 명세를 정의하고 있다.

생명정보 분석 과정의 워크플로우 모델링을 위한 생명정보 데이터 양식에 대한 OWL 기반 온톨로지 표현을 <그림 16>과 같이 기술할 수 있다. 각각의 생명정보 분석도구에 대한 입출력 형식은 서열데이터, 서열정렬데이터, 단백질구조, 단백질간 상호작용 등과 같은 생명정보 데이터 일 수 도 있지만 문자열이나 숫자와 같은 Literal 형식일 수도 있다. 따라서 생명정보 분석 과정에 대한 워크플로우 모델링을 위한 OWL 표현에는 생명정보 데이터 양식과 Literal 형식을 모두 포함한다.

이렇게 정의된 OWL 표현 파일을 프로그램 코드에서 핸들링하기 위해서는 Jena와 같은 온톨로지 처리 API가 필요하다.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.bioworks.org/bioentry#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.bioworks.org/bioentry">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="EMBLFormatReport">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >EMBL format report</rdfs:label>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >EMBL format report</rdfs:comment>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="EMBLFormat"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="ENZYMEEntry">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ENZYME Entry</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="SwissProtFormatReport"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ENZYME Entry</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="BtwistedReport">
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >btwisted report</rdfs:comment>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >btwisted report</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="GeneralReport"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="PFAMFormat">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >PFAM format</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="DatabaseEntry"/>
    </rdfs:subClassOf>
    <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >PFAM format</rdfs:comment>
  </owl:Class>
  ...
</red:RDF>
```

그림 16. 생명정보 분석 도구 입출력 양식을 위한 OWL 파일

<그림 17>은 Jena API를 사용하여 정의된 OWL 파일을 파싱하여 생명정보 데이터 양식에 대한 계층적 트리 구조를 생성하고 Java 트리 뷰를 통해 보여주고 있다. Jena API를 사용하면 생명정보 분석도구 간의 입출력 연계 시 데이터 양식의 계층적 호환을 쉽게 검사할 수 있다. 가령, ProteinFASTASequences와 DNAFASTASequences가 온톨로지 계층 상에서 모두 FASTASequences의 하위 클래스라고 할 때, ProteinFASTASequences또는 DNAFASTASequences 형식의 출력값은 FASTASequences 형식의 입력값으로 대입되어 질 수 있다. 이 때 Jena API의 OntClass 클래스의 hasSubClass 메소드를 사용한다.

### 2.3. XML 기반 생명정보 분석 도구 통합

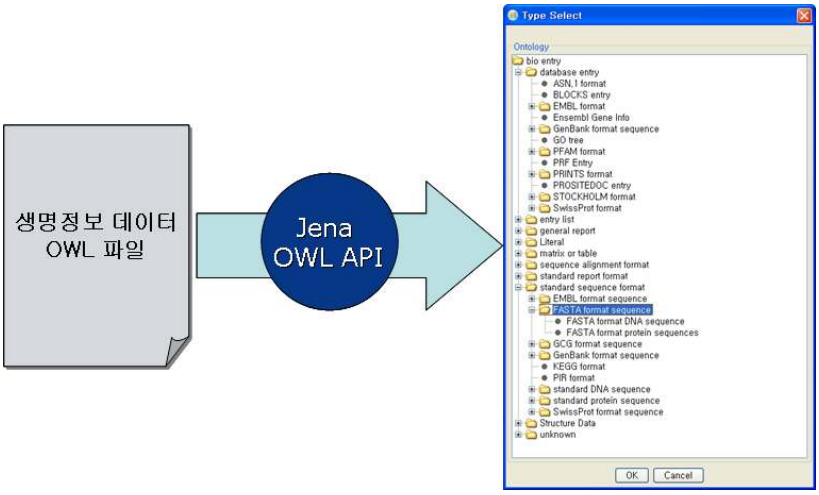


그림 17 Jena API를 통한 생명정보 데이터 OWL 파싱 및 트리뷰

생명정보 분석 도구들을 효과적으로 통합하기 위해 가장 먼저 고려해야 할 점은 같은 동작을 하는 분석 도구라 할지라도 실행방식이 여러 가지라는 것이다. 다시 말해서 유전자 데이터베이스에서 하나의 서열정보를 얻는 프로그램의 경우 명령행 실행, 스크립트, 그리고 웹서비시스, XML-RPC, REST등의 원격 호출 방식 등 다양한 실행 방식이 있을 수가 있다. 따라서 효과적인 생명정보 분석 도구 통합을 위해서는 이러한 다양한 실행 방식을 고려하여 유연한 통합 스키마를 구현해야만 한다. <그림 18>은 다양한 실행방식을 고려한 생명정보 분석 도구 클래스 계층 다이어그램이다.

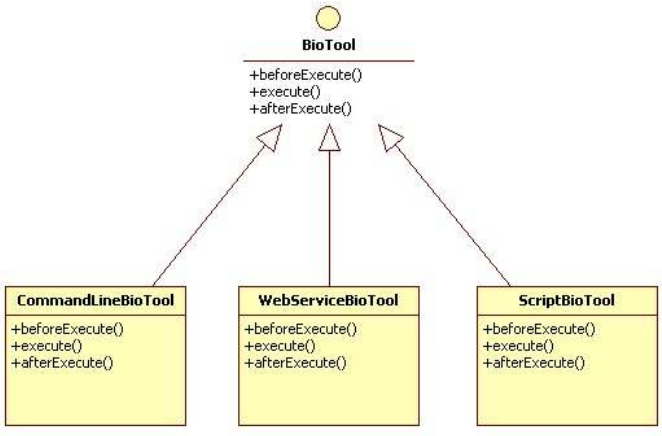


그림 18. 생명정보 분석도구 클래스 다이어그램

각 도구의 실행 방식 별로 BioTool 클래스의 하위클래스(CommandLineBioTool, WebServicesBioTool, ScriptBioTool 등)이 정의되며 각각의 하위클래스에서 주어진 입력 값을 사용하여 beforeExecute, execute, afterExecute 추상 메소드를 구현한다.

메소드 이름에서 알 수 있듯이 각각의 메소드는 실행 전처리, 도구 실행, 실행 후처리 방법을 각각 정의한다.

생명정보 분석 도구 통합을 위해 다른 한 가지 고려해야 할 점은 확장성과 유연성이다. 앞서 말한 바와 같이 전 세계 생명 과학 분야 기업 또는 연구기관들은 각기 다른 형태의 포맷으로 생명정보 분석 도구들을 개발하고 배포해 왔기 때문에 이러한 이질적인 분석도구들의 통합을 위해서는 확장적이고 유연한 통합 스키마가 필요하다. XML은 이식성, 재사용성, 확장성, 효율적인 데이터 교환 등의 이점을 갖고 있어 이질적은 생명정보 분석 도구 통합을 위해 적합하다.

XML(eXtensible Markup Language)[22]는 데이터의 구조화, 저장, 상호교환을 위한 W3C의 공개 표준 규약으로서 그 이름으로부터 XML의 정의 및 특징을 말할 수 있다. 즉, Extensible은 사용자 정의 태그와 속성을 사용할 수 있어 추상화할 데이터 모델에 따라 마음대로 정의 가능하고, Markup은 중첩된 태그로 문서를 구조화하고 각 항목을 마크업 할 수 있다는 것이며, Language는 정해진 문법 규칙을 갖고 있다는 것이다. XML의 주요 장점을 살펴보면, 첫째 XML은 일반 텍스트 문서로서 플랫폼, 개발언어에 독립적이다. 따라서 시스템 또는 응용 프로그램 간의 상호 데이터 교환이 용이하다. 둘째 문서의 내용과 포맷팅을 분리하여 하나의 XML 문서를 다양한 형태로 Formatting 하거나 Display 할 수 있다. 셋째 XML은 확장적 모델링 언어로 확장 가능한 사용자 정의 태그를 통해 거의 모든 도메인의 개념을 추상화 할 수 있다.

생명과학 분야에서도 XML을 통한 생물학적 의미들의 추상화 연구가 활발히 이루어지고 있다. <표>는 생명과학 분야에서의 세부 도메인별 XML 정의들을 정리한 것이다.

| 분류 | 이름 | 정의내용 |
|---|---|---|
| Sequence/ Annotation | BIOML | DNA와 단백질 서열 |
| | DAS | DAS 시스템에서 사용되며, 데이터소스, 서열과 주석 유형, feature 등을 정의 |
| | GAME | 유전체 주석정보 |
| | RNAML | RNA 분자 구조 및 서열 |
| Protein | InterPro | 단백질 family, domain, functional site 등 |
| | PROXIML | 단백질에 대한 전반적인 정보 |
| | SP-ML | SwissProt 데이터베이스 레코드 |
| Analysis | BlastXML | NCBI Blast output |
| | Seq-entry | NCBI Entrez 데이터베이스 레코드 |
| Physical | CML | |
| | Petri Net Markup Language | Petri net의 대사경로 |
| Expression | GEML | 유전자 발현 데이터 |
| | MAGE-ML | 마이크로어레이 유전자 발현 실험 스펙 및 데이터 |

표 2. 생명과학 분야에서의 XML 정의 스키마

<그림 19>는 생명정보 분석도구 통합을 위한 XML 표현에 대한 하나의 예를 보여준다. 주목할 점은 각각의 입출력 데이터 타입을 생명정보 데이터 양식에 대한 OWL 표현 파일에서의 온톨로지 클래스 식별자(OntClass URI)로 한다는 것이다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
<!-- cai  -->

   <bean id="cai " class="org.bioworks.domain.bioservice.CommandBioServiceDescriptor">
       <property name="name" value="cai"/>
       <property name ="description"><value>CAI codon adaptation index .</value></property>
       <property name ="copyLevel" value="1"/>
       <property name ="editable" value="true"/>
       <property name ="referenceLocation" value="underconstruction.html"/>
       <property name ="available" value="true"/>
       <property name ="commandPath"><value>${bioworks.tool.path}/cai.run</value></property>
       <property name="inputDescriptors">
           <list>
               <bean class="org.bioworks.domain.bioservice.CommandBioServiceInputDescriptor">
                   <constructor-arg>
                       <value>seqall</value>
                   </constructor-arg>
                   <constructor-arg>
                       <value>${bioworks.ontclass.ns}#FastaSequences</value>
                   </constructor-arg>
                   <constructor-arg>
                       <ref bean="cai"/>
                   </constructor-arg>
                   <property name="description">
                       <value>Nucleotide sequence(s)</value>
                   </property>
                   <property name="required" value="true"/>
                   <property name="valueSeparator" value=" "/>
                   <property name="nonValue" value="false"/>
                   <property name="asFileOnExecution" value="true"/>
               </bean>

               <bean class="org.bioworks.domain.bioservice.CommandBioServiceInputDescriptor">
                   <constructor-arg>
                       <value>cfile</value>
                   </constructor-arg>
                   <constructor-arg>
                       <value>${bioworks.ontclass.ns}#CodonUsageTable</value>
                   </constructor-arg>
                   <constructor-arg>
                       <ref bean="cai"/>
                   </constructor-arg>
                   <property name="description">
                       <value>Codon usage table name</value>
                   </property>
                   <property name="required" value="true"/>
                   <property name="valueSeparator" value=" "/>
                   <property name="nonValue" value="false"/>
                   <property name="asFileOnExecution" value="true"/>
                   <property name="defaultValueAsText" value="Eyesst_cat.cut"/>
               </bean>
           </list>
           ...
       </property>
   </bean>
</beans>
```

그림 19. 생명정보 분석도구에 대한 XML 정의 예

## 2.4. 워크플로우 실행 엔진 모델

<그림 20>은 하나의 생명정보 분석 과정에 대한 워크플로우 실행 개념 모델이다. 하나의 워크플로우는 Processor를 Vertex로 하고 Processor간 입출력 연계 링크를 Edge로 하는 'Directed Graph'로 표현되어 질 수 있다. 초기 실행 Process는 최상위 Vertex에 해당하는 Processor가 되며 각각의 Processor는 Link에 의해 연결되어 부모 Process와 자식 Processor를 갖게 된다. 하나의 Processor는 상위 Process들이 모두 종료되어 그것의 Output이 해당 Processor의 Input으로 적합하게 설정되었을 때 비동기적으로 실행된다. Processor 간의 Link에 의한 데이터 연계가 일어나는 시점에서 특정 전이 조건에 대한 검사와 데이터 변환이 이루어진다. 이러한 워크플로우

실행 개념 모델 분석을 통하여 전체 시스템에서 사용될 워크플로우 공통 모델과 세

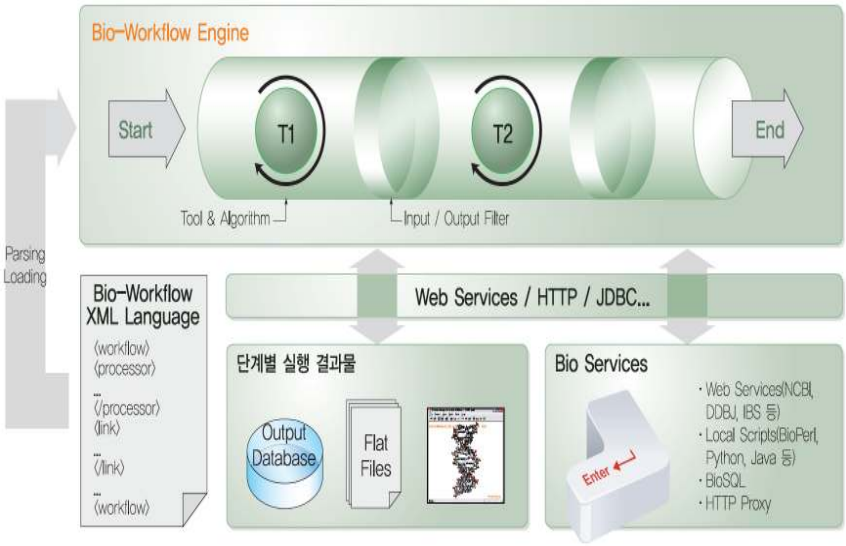


그림 20. 생명정보 분석 워크플로우 실행개념도

부 클래스 모델을 구현해 낼 수 있다. <그림 21>은 생명정보 분석 과정에 대한 워크플로우 모델과 세부 클래스 모델의 계층적 다이어그램을 보여주고 <표 3>는 각 클래스의 세부적인 기능에 대해 설명하고 있다.

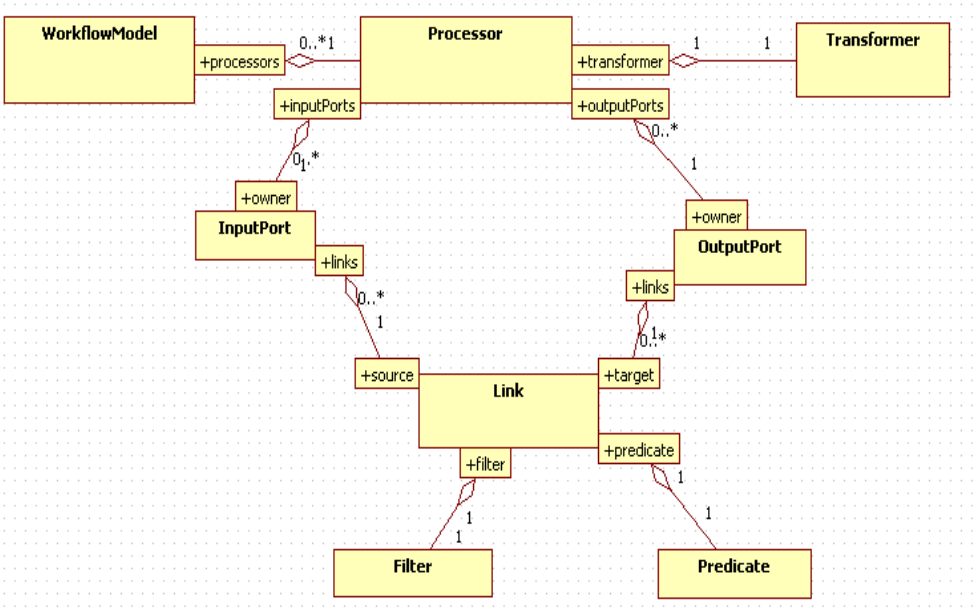


그림 21. 생명정보 분석 워크플로우 클래스 계층 모델

워크플로우 객체 모델을 데이터 저장 매체에 저장하는 방법에는 파일, 데이터베이스 등의 여러 가지 방법이 있을 수 있다. 이 중 데이터베이스에 객체모델을 영구적으로 저장할 수 있는 방법을 살펴보도록 한다. 대부분의 IT 시스템 개발에서는 데

| 클래스 명 | 설명 |
|---|---|
| WorkflowModel | Workflow에 대한 추상화 모델로서 Processor를 Vertex로 하고 Link를 Edge로하는 'Directed Graph" |
| Processor | 단위 분석도구에 대한 프로세스를 실행하며 상위 Linked Process들이 정상적으로 종료되었을 때 실행 |
| Transformer | 각 분석도구를 실행하는 Delegated 인터페이스 클래스 |
| InputPort/OutputPort | Processor의 입출력 데이터에 대한 추상화 모델 |
| Link | Workflow Graph 모델에서 Edge에 해당하며 Processor 간 데이터 흐름을 표현 |
| Filter | Link 객체에서의 데이터 필터링을 처리 |
| Predicate | Link 객체의 데이터 전이 시의 전이 조건을 판단하여 Processor간 데이터 흐름 제어함 |

표 3. 생명정보 분석 워크플로우의 세부 클래스 모델 세부사항

이터베이스와 연동하여 다양한 비즈니스 로직을 수행할 데이터베이스 계층을 만들고, 유지 보수하는 작업에 상당한 시간과 노력을 투자하게 된다. 만일 데이터베이스의 스키마가 바뀌는 경우에는 애플리케이션의 나머지 부분도 크게 변경해야 한다. 특히, 앞서 제시한 워크플로우 모델은 객체지향 기술을 사용하고 있는 데 반해, 데이터베이스는 관계 형 데이터 모델을 이용하게 되면 이로 인해 객체 모델링과 관계형 데이터 모델링 사이에 개념적 불일치가 존재하게 되고, 거의 모든 시스템 구현 및 유지보수에 있어 심각한 수준의 복잡도 부담을 떠안게 된다. 이러한 문제들을 해결하기 위해 객체 수준의 데이터베이스 모델을 OR(Object-Relation)맵핑을 통해 자동화해 주는 다양한 IT 기술들이 등장했다. 이 중 Hibernate는 현재 세계적으로 가장 많은 관심을 받고 있는 Java 기반의 OR 맵핑 엔진이다. IBTS는 객체지향 적인 데이터베이스를 구현하기 위해 Hibernate 엔진을 사용하였다. Hibernate는 XML로 표현된 객체지향 데이터 모델을 관계 형 데이터베이스와 맵핑될 수 있도록 한다. 이로써 생명정보 서비스 프레임워크의 모든 비즈니스 컴포넌트는 워크플로우의 객체지향 데이터 모델을 참조하여 구현되어 질 수 있으며, 결과적으로 데이터베이스 계층을 포함한 전체 소프트웨어 아키텍처는 객체지향적으로 구성되어 질 수 있다. <그림 22>은 앞서 제시한 워크플로우 클래스 모델에 대한 Hibernate XML 맵핑 파일의 예를 보여주고 있다.

## 2.5. 슈퍼컴퓨팅 연계를 통한 대규모 분석처리 지원

Sequencing, Bootstrapping, 그리고 다중서열분석 등과 같은 생명정보 분석 업무는 높은 컴퓨팅 능력을 필요로 한다. 게다가 폭발적으로 증가하는 생명정보 데이터를 빠르고 정확하게 분석하여 유용한 생물학적 정보를 도출하기 위해서는 슈퍼컴퓨팅 자원이 필수적이다. <그림 23>은 세포의 수, 즉 분석 데이터의 사이즈가 커질수록 요구되는 컴퓨팅 자원이 기하급수적으로 증가되는 것을 보여주고 있다. 결국 이

러한 대용량 생명정보 데이터를 누가 얼마나 빠르고 정확하게 분석하여 유용한 생

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping package="org.bioworks.domain.workflow">

    <joined-subclass name="BioServiceWorkflow" extends="org.bioworks.domain.AbstractUserData" table="BIOSERVICE_WORKFLOW">
        <key column="WORKFLOW_ID"/>

        <component name="status" class="WorkflowStatus">
            <property name="status"
                      type="integer"
                      column="STATUS"
                      not-null="true"/>
            <property name="statusText"
                      type="string"
                      column="STATUS_TEXT"
                      not-null="true"/>
        </component>

        <property name="startedDate"
                  type="timestamp"
                  column="STARTED_DATE"
                  not-null="false"/>

        <property name="finishedDate"
                  type="timestamp"
                  column="FINISHED_DATE"
                  not-null="false"/>

        <list name="tasks"
              cascade="all,delete-orphan"
              lazy="true">
            <key column="WORKFLOW_ID"/>
            <list-index column="TASK_INDEX"/>
            <one-to-many class="BioServiceTask"/>
        </list>
        <property name="shareInputBioEntryValues"
                  type="boolean"
                  column="SHARE_INPUT_VALUES"
                  not-null="false"/>
    </joined-subclass>
    ...
</hibernate-mapping>
```

그림 22. 워크플로우 클래스 모델에 대한 Hibernate OR 맵핑 예

물학적 데이터를 얻어내는가가 핵심 경쟁력인 것이다 .

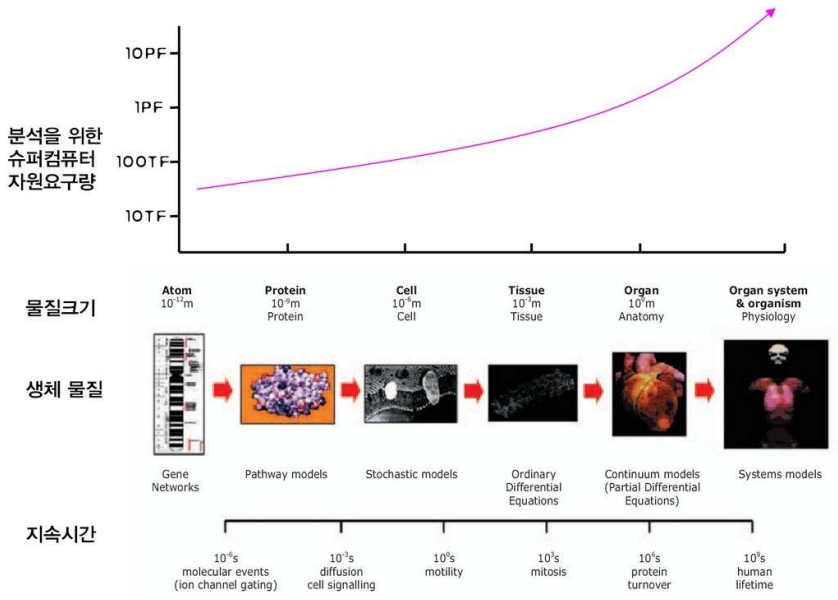


그림 23. 세포 사이즈 증가에 따른 컴퓨팅 자원 요구량

이러한 생명정보 분석에서의 대규모 분석 요구를 지원은 BeoBLAST[23] 프로젝트에서 첫 번째로 시도되었다. 그러나 이 시스템은 오직 BLAST 프로그램에만 한정된 것이었다. <그림 24>은 하나의 생명정보 워크플로우 모델에 포함된 각 분석 도구들이 슈퍼컴퓨팅 기반 환경에서 실행되는 과정을 도식화 한 것이다. 클라이언트에 의해 작성된 워크플로우 모델은 워크플로우 실행 스케즐러에 의해 실행되고 각각의 중간 결과물은 데이터베이스에 저장된다. 워크플로우 실행 스케즐러는 워크플로우 모델을 구성하는 Processor에서 참조하는 분석도구들에 대한 실행을 PBS(Portable Batch System)과 같은 배치큐 시스템의 큐에 순서대로 배치작업으로 등록한다. 배치큐 시스템은 각 배치작업을 연계된 클러스터 환경에서 순차적으로 실행하고 각각의 작업 실행 상태와 결과물들은 JobMonitor에 의해 수집되고 데이터베이스에 저장•관리된다.

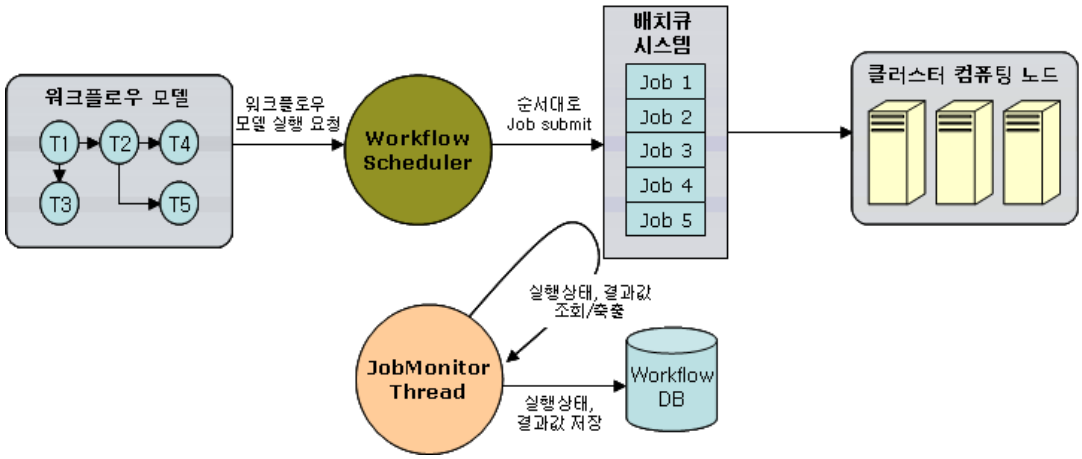


그림 24. 슈퍼컴퓨팅 환경에서의 워크플로우 실행 개념도

사용자 정의 스크립트 분석도구 및 분석도구 간 입출력 연계 시 데이터 변환 스크립트의 동적 실행을 위해서는 스크립트 언어 별 실행 엔진이 필요하다. 스크립트 실행 엔진과 관련된 라이브러리는 다양한 프로그래밍 언어로 구현된 것들이 많겠지만 여기에서는 Java 언어로 구현된 스크립트 실행엔진 라이브러리만을 다루고자 한다. 먼저 Java 언어로 작성된 스크립트를 실행하기 위한 대표적인 라이브러리로는 BeanShell[24]이 있다. BeanShell은 경량의 Java 스크립팅 엔진으로서 Java 언어로 작성된 스크립트를 JVM(Java Virtual Machine) 환경에서 동적으로 실행한다. BeanShell로 작성된 스크립트는 별도의 문법 추가나 확장 코드 없이 그 자체로 JVM 위해서 동적으로 실행되어 질 수 있기 때문에 기존의 Java 언어로 작성된 프로그램도 코드 변경이 거의 없이 쉽게 BeanShell 스크립팅 엔진에 의해 실행되어 질 수 있다. 다음으로 Ruby 언어로 작성된 스크립트를 Java 기반으로 동적 실행하는 스크립팅 엔진 라이브러리로는 JRuby[25]가 있다. JRuby는 Ruby 인터프리터를 JVM에서 실행할 수 있도록 Java 기반으로 구현한 것으로 Ruby 코드와 Java 코드를 서로 연동하여 두 언어의 장점을 최대한 발휘할 수 있게 설계되어 있다. Ruby 코드

를 Java 프로그램 내에서 사용하기 위해서는 먼저 Java 언어로 Interface 클래스를 만들고 이것에 대한 구현 클래스를 Ruby 언어로 작성하여 Java 클라이언트 프로그램에서 JRuby 스크립팅 엔진을 통해 해당 구현 클래스의 객체 인스턴스를 만들어 사용한다. 마지막으로 Python 언어로 작성된 스크립트 코드를 Java 기반으로 동적 실행하는 자바 기반 스크립팅 엔진으로는 Jython[26]이 있다. Jython은 Python의 자바 구현으로 이전에는 Python 표준 구현 이름에 따라 JPython라는 이름으로 개발되다가 2000년 이후부터 오픈소스 프로젝트 기반으로 Jython이라는 이름으로 개발이 진행되고 있다. Jython을 이용하여 자바 프로그램 안에서 Python 코드를 동적으로 실행하기 위해서는 Jython 패키지에 포함된 PythonInterpreter 클래스를 이용한다. PythonInterpreter 객체를 자바 코드 내에서 생성하고 여기에 Python 코드를 적재하고 실행한 후 결과 값을 얻어 낸다.

다양한 스크립트 언어(Java, Ruby, Python 등)를 자바 기반의 동적 스크립트 실행을 위해 무엇보다 고려해야 할 점은 시스템 보안에 대한 것이다. 가령 사용자 정의 스크립트에 서버를 종료하거나 중요한 보안파일에 접근하는 코드가 포함된다면 매우 심각한 일이 아닐 수 없다. 따라서, 사용자 정의 스크립트의 동적 실행 시에는 시스템 차원에서의 보안 설정이 필수적이다. 자바 기반의 시스템 환경에서는 기본적으로 자바보안아키텍처에서 지원하는 Security Manager를 사용하여 파일, 네트워크 등의 대부분의 리소스에 대한 클라이언트 코드의 접근을 제어한다. Security Manager는 java.lang.SecurityManager 클래스(혹은 이를 확장한 클래스)로서 특정 클라이언트 코드에서 행하는 작업들이 실행 시간에 특정 리소스에 대한 접근이 허용되는지 여부를 검사하는 역할을 한다. 클라이언트 코드가 일단 Security Manager의 제어 하에서 실행되면 관련 보안 정책(Security Policy)에 의해 허용된 작업들만 수행될 수 있으며 이 때 정책(policy)은 기본적으로 일반 텍스트 문서로 된 정책파일(policy file)로 기술된다. 클라이언트 코드가 Security Manager 하에서 돌아가게 하는 건 그저 간단한 클라이언트 코드를 Java 인터프리터로 실행할 때 -Djava.security.manager 옵션을 설정하는 것만으로도 가능하며 정책 파일 역시 어느 텍스트 에디터로도 쉽게 만들 수 있다. 다음은 JRuby로 작성된 클라이언트 Ruby 코드의 실행 시 특정 리소스에만 접근이 가능하도록 설정해 놓은 정책파일의 일부분이다(jruby.jar는 JRuby 스크립팅 엔진이 포함된 아카이브 파일임). 정책파일을 클라이언트 코드에 적용하기 위해서는 Java 인터프리터를 실행할 때 -Djava.security.policy=policy.txt 옵션을 설정한다.

```
grant codeBase "file:/home/bioworks/works/bioworks_server/site/WEB-INF/lib/jruby.jar"
{
        permission java.util.PropertyPermission "*", "read";
        permission java.lang.RuntimePermission "accessDeclaredMembers";
        permission java.lang.RuntimePermission "createClassLoader";
        permission java.lang.RuntimePermission "defineClassInPackage.java.lang";
        permission java.lang.RuntimePermission "getenv.*";
        permission java.util.PropertyPermission "*", "read,write";
        permission java.io.FilePermission "${user.home}/.jruby", "read,write";
        permission java.io.FilePermission
        "file:/home/bioworks/works/bioworks_server/site/WEB-INF/lib/jruby.jar!/-", "read";
        permission java.lang.reflect.ReflectPermission "suppressAccessChecks", "public";
};
```

## 2.6. 클라이언트-서버 간 데이터 동기화 최적화

앞서 말한 바와 같이 대용량의 생명정보 데이터를 처리하고 시•공간의 제약 없이 워크플로우의 실행 현황을 모니터링 하기 위해서는 클라이언트-서버 아키텍처로 워크플로우 기반 생명정보 서비스 프레임워크를 구현해야 한다. 클라이언트 프로그램에서 서버 측에서 실행되는 워크플로우의 실행현황을 실시간으로 모니터링 하기 위한 방법으로는 일정 시간 간격으로 클라이언트에서 서버 측으로 워크플로우 데이터를 요청하여 처리하는 클라이언트 폴링(Polling) 방식과 워크플로우 상태 변경 시 서버측에서 클라이언트 측으로 워크플로우 데이터를 밀어 넣는 서버 푸쉬(Push)방법이 있다. 클라이언트 폴링 방식의 가장 큰 단점은 많은 클라이언트가 서버 측으로 데이터를 동시 요청할 경우 생성되는 트래픽 부하와 서버 측 리소스 부하에 있다. 최악의 경우 실행시간이 긴 워크플로우의 데이터 동기화를 클라이언트 폴링하면 거의 매번 동일한 워크플로우 상태를 서버 측에서 가져오게 되는 결과를 초래한다. 이는 클라이언트 측이나 서버 측 모두 심각한 자원의 낭비라 할 수 있겠다. 물론 서버 측 부하는 폴링 간격을 늘림으로써 경감될 수 있으나, 서버 측과 클라이언트 측 간의 데이터 동기화의 지연을 초래한다. 따라서 생명정보 분석과정에 대한 워크플로우 작업이 비교적 긴 시간을 요구한다고 가정할 때 클라이언트-서버 간 데이터 동기화는 서버 푸쉬 기술을 적용하는 것이 바람직하다. 서버 푸쉬 기술은 초기 웹브라우징 환경을 선도했던 넷스케이프 사에서 최초로 제안한 이래로 최근의 Web 2.0 환경에서의 AJAX[27] 등의 RIA(Rich Internet Application)기술의 발전과 더불어 Comet이라는 기술로 각광받고 있다.

Comet이란 웹 클라이언트(보통 웹 브라우저)의 명시적인 요청이 없어도 서버 푸

쉬 방식으로 동작하는 웹 프로그래밍 모델을 말하며 차세대 웹의 근간이 될 HTML5[28] 표준에서 WebSocket 기술을 통해 지원되어 주류 기술로 인정받고 있다. Comet 기술의 동작방식은 일반적으로 Long Polling 방식으로 클라이언트가 서버에 접속 하면 서버는 계속 접속을 유지하고 있다가 서버 측 이벤트가 발생하면 클라이언트로 이를 전송하고 HTTP 트랜잭션을 마친다. 서버 측은 클라이언트 접속을 유지하며 이러한 과정을 반복한다. Comet 을 효율적으로 지원하기 위해서는 하나의 스레드가 여러 클라이언트 접속을 유지하며 각 접속에서 여러 개의 요청을 처리할 수 있어야 한다. 현재의 자바 기반 웹애플리케이션 서버 환경(Java Servlet 환경)에서는 이것이 불가능하다. 따라서 이에 대한 대안이 여럿 등장했다. 자바 서블릿 서버 환경 중에는 Jetty 서블릿 컨테이너가 최초로 Continuation 방식의 Comet 기술을 구현했다. 현재 초기 드래프트 리뷰 상태에 있는 자바 서블릿 3.0에서 Jetty의 Continuation과 비슷한 방식으로 지원이 논의되고 있다. Continuation[29]은 scheme 등의 언어에서 지원하던 개념인데, Jetty 서블릿 컨테이너에서의 Continuation 지원 방식은 이와 유사한 메커니즘으로 버전 6.0부터 Continuation API를 제공하고 있다. Jetty Continuation의 동작방식은 서버에 접속한 클라이언트의 요청 처리를 중지(suspend)시킨 후 재개(resume)하면 바로 중지시킨 그 지점에서 다시 진행하는 것이 아니라 요청 처리 체인(FilterChain)을 다시 처음부터 진행한다는 점에서 기존 Continuation과 약간의 차이점이 있다. <표 4>는 일반적인 웹 서비스(Web 1.0), AJAX와 같은 Web 2.0 기술과 Comet 기술을 적용한 웹 서비스, 그리고 Jetty Continuation 기술을 함께 적용 했을 시의 웹 서비스의 성능 개선 내용을 보여준다.

| | Formula | Web 1.0 | Web 2.0 + Comet | Web 2.0 + Comet + Continuations |
|---|---|---|---|---|
| **Users** | u | 10000 | 10000 | 10000 |
| **Requests/Burst** | b | 5 | 2 | 2 |
| **Burst period (s)** | p | 20 | 5 | 5 |
| **Request Duration (s)** | d | 0.200 | 0.150 | 0.175 |
| **Poll Duration (s)** | D | 0 | 10 | 10 |
| | | | | |
| **Request rate (req/s)** | rr=u*b/20 | 2500 | 4000 | 4000 |
| **Poll rate (req/s)** | pr=u/d | 0 | 1000 | 1000 |
| **Total (req/s)** | r=rr+pr | 2500 | 5000 | 5000 |
| | | | | |
| **Concurrent requests** | c=rr*d+pr*D | 500 | 10600 | 10700 |
| **Min Threads** | T=c<br>T=r*d | 500<br>- | 10600<br>- | -<br>**875** |
| **Stack memory** | S=64*1024*T | 32MB | 694MB | **57MB** |

표 4 Continuation 기반 Comet 기술 적용시 서버 자원 소요량(출처: http://www.webtide.com/downloads/whitePaperAjaxJetty.html)

앞서 제시한 것처럼 Jetty Continuation 기반의 Comet 기술을 활용하면 웹서비시스 기반의 클라이언트-서버 환경에서의 서버의 부하를 최소화하면서 클라이언트에서 실시간으로 생명정보 분석 워크플로우의 실행 상태 및 결과물을 동기화 할 수 있다. <그림 25>은 Jetty Comet 기반의 생명정보 워크플로우 데이터 동기화에 대한 실행 모델을 도식화 한 것이다.

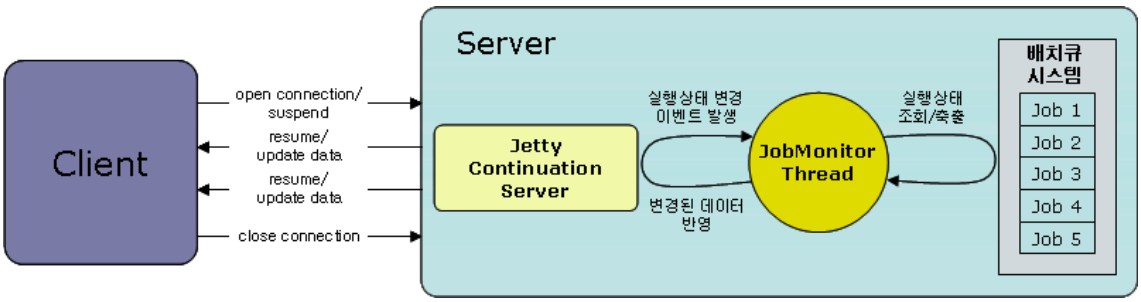


그림 25. Jetty Comet 기반 생명정보 워크플로우 데이터 동기화 실행모델

<그림>에서 볼 수 있듯이 초기에 클라이언트가 서버에 접속하면 접속 상태는 유지되면서 요청처리는 중지된 상태로 남아 있다. 이 때 Continuation 객체 내에서 요청처리에 대한 객체는 유효한 상태로 남아 있다. 서버 측의 JobMonitor는 일정 간격으로 배치 큐에 적재되어 있는 워크플로우에 포함된 분석 도구 작업을 모니터링하다가 각 실행 작업의 상태가 변경되면 Jetty 서버에 워크플로우 실행 상태 변경에 대한 이벤트를 발생시키고 변경된 워크플로우 데이터를 전달하게 된다. Jetty 서버가 워크플로우 상태 변경 이벤트를 받으면 Continuation 객체는 중지되었던 요청처리를 resume하고 변경된 워크플로우 데이터를 클라이언트 측으로 반환하게 된다.

## 2.7. 협력연구를 위한 워크플로우 공유 환경 구현

생명정보라는 말이 의미하듯이 생명정보 분석 과정은 생명과학과 IT 기술이 융합된 과정이다. 최적의 생명정보 분석과정을 모델링하고 이를 활용하기 위해서는 생명과학, 생명정보학, IT 분야의 전문가 간의 협력 연구가 가장 효율적이다. 서로 다른 분야의 연구자(또는 개발자)의 원활한 협력 연구를 위해서는 상호간의 커뮤니케이션을 위한 일관된 매개체, 즉 접점 인터페이스를 활용하는 것이 효과적이다. 이러한 점에서 생명정보 분석 과정에 대하 워크플로우는 연구자 간 협업을 위한 최상의 접점 인터페이스라고 할 수 있다.

하나의 워크플로우를 연구자 간에 공유하는 방법은 워크플로우 게재, 검색, 복사의 순환 과정을 거친다. 워크플로우 게재 단계는 자신이 소유한 워크플로우에 대한 공유 수준(None, ReadOnly, ReadWrite)을 설정하여 다른 연구자가 복사하여 사용할 수 있도록 공개하는 것이다. 이 때 워크플로우를 구성하는 각 분석 도구의 입력값을 공유할 것인가를 설정할 수 있어야 한다. 왜냐하면 워크플로우 그 자체뿐만 아니라 각 입력 값도 매우 중요한 자료일 수 있기 때문이다. 특정 사용자에 의해

워크플로우가 게재되면 다른 사용자가 이를 효과적으로 검색할 수 있게 하는 것도 매우 중요하다. 왜냐하면 초기에 생명정보 분석과정에 대한 워크플로우를 디자인 하는 일은 대부분의 사용자에게 있어 매우 부담스러운 작업일 수 있기 때문이다. 따라서 사용자가 작성하고자 하는 워크플로우와 유사한 공개 워크플로우를 간단한 검색 키워드만으로도 쉽고 빠르게 검색할 수 있는 기능을 구현해야 한다. 검색된 워크플로우를 복사하여 자신의 워크플로우로 재 작성하는 과정은 윈도우즈 탐색기에서 문서 파일을 복사하여 편집하는 과정과 유사하다. 복사된 워크플로우에 대한 사용자(또는 소유자)는 해당 워크플로우를 재편집하여 다시 다른 사용자에게 게재할 수 있다.

지금까지 제시한 워크플로우 공유방법은 하나의 동일한 기능을 수행하는 워크플로우의 복사물이 지나치게 많아져 자칫 서버 저장소의 부하를 초래할 수도 있다. 그러나 워크플로우의 공유 및 편집의 자유도가 증가하여 보다 많은 사용자가 초기 시드 워크플로우에 대한 활용이 많아질수록 생명정보 분석 과정에 보다 최적화된 워크플로우가 재생산될 가능성이 커진다. 이것은 현재 IT 분야에서 각광받고 있는 오픈소스 프로젝트의 기본 철학과도 일맥상통하는 개념으로도 볼 수 있다.

## 3. 결론

생물정보학 분야는 최근 급격히 세계시장이 증가하고 있는 분야이며, 향후 국가 전략 산업으로 육성되고 있는 분야이다. 특히 생물정보 분석 결과는 유전체학, 전사체학, 단백질체학, 대사체학, 약리유전체학 등 분자생물학의 모든 분야에서의 질문들에 대한 잠재적 의미 있는 답을 줄 수 있다. 따라서 유전자 구조 및 기능, 진화상 관계 등 생명정보 분야에서의 중요한 문제들에 대해 발견되는 새로운 지식을 종합적으로 신속하게 분석하여 생명과학 연구에 활용하는 것은 매우 중요한 일이다. 그러나 이러한 생물정보 분석 기술의 발전에도 불구하고, 생명과학 연구자들이 자신의 연구에 활용하기에는 몇 가지 문제점이 있다. 첫째, 생물정보 데이터 및 도구의 이질성이다. 실로 다양하고 이질적인 생물정보 분석도구 들이 서로 다른 입출력 포맷을 가지고 분산되어 있어, 연구자들은 자신의 연구 환경에 맞는 분석도구를 선택하고 그것을 활용하는 데 있어 많은 어려움을 겪게 된다. 둘째, 여러 생물 정보 데이터베이스를 검색하여 다양한 정보를 축출하고 이에 대한 다양한 분석도구의 적용 및 결과물 분석 등의 여러 단계의 단위 분석 도구의 입출력 연계로 이루어진다. 그러나 이질적인 분석도구의 입출력 양식 때문에 도구 간 데이터 연계는 매우 어려운 일이다. 셋째, 생명과학 연구자, IT 개발자, 생명정보 연구자 들 간의 협력 연구 환경이 미흡하다는 것이다. 생명정보라는 말이 의미하듯이 생명정보 분석 과정은 생명과학과 IT 기술이 융합된 과정이다.  최적의 생명정보 분석과정을 모델링하고 이를 활용하기 위해서는 생명과학, 생명정보학, IT 분야의 전문가 간의 협력 연구가 매우 중요하며 이를 위한 일관성 있고 통합된 협업연구 환경이 절실히 요구된다. 넷째, 대용량, 대규모 분석 환경이 미흡하다는 것이다. 최근 선진국들을 중심으로

빠르게 발달하고 있는 첨단 생명과학 연구는 기존의 소규모 연구방식에서 벗어나서 점차적으로 대용량의 생물학 정보들을 대상으로 복잡한 계산과정을 요구하기 때문에 고도의 컴퓨팅 인프라가 필요시 된다. 그러나 이와 같은 조건에 부합하는 전산 자원들을 개개의 연구자들이 사용하기에는 비용, 운영인력 등의 제약사항 때문에 현실적으로 불가능한 일이다.

본 연구보고서에서는 생명정보 분석 기술의 활용에 있어서의 이러한 문제점들을 해결하기 위하여 이질적인 생명정보 데이터와 도구를 확장적으로 통합하고 생명정보 분석과정을 효과적으로 모델링하고 실행할 수 있는 워크플로우 기반의 서비스 프레임워크의 구현 방법 및 세부 기술 요소에 대해 제시하였다. 가장 먼저 생명정보 분석과정의 특징과 국내외 사례분석을 통하여 워크플로우가 생명정보 분석 과정에 대한 가장 효과적인 모델임을 제시하였다. 또한, 웹서비시스 기반 클라이언트-서버 아키텍처, 온톨로지 기반 이질적 생명정보 데이터 통합, XML 기반 생명정보 분석 도구 통합, 슈퍼컴퓨팅 연계를 통한 대규모 분석처리를 지원하는 워크플로우 실행엔진 모델, 클라이언트-서버 간 데이터 동기화 최적화, 그리고 협력연구를 위한 워크플로우 공유 등 최적의 워크플로우 기반 생명정보 서비스 프레임워크 구현방법 및 세부 기술요소를 제시하였다.

# 참고문헌


1. Tom Oinn, et al., Taverna: *A tool for the composition and enactment of bioinformatics workflows*, Bioinformatics, 20, 17:3045-3054, 2004.

2. Life Sciences Practice Team, *BioWBI and WEE: Tools for Bioinformatics Analysis Workflows*, IBM Business Consulting Services-AIS, 2004.

3. Dowell, R. D., Jokerst, R. M., Day, A., Eddy, S. R. & Stein, L. *The distributed annotation system.* BMC Bioinformatics, 2:7, 2001.

4. Francis Tang, et al., *Wildfire: distributed, Grid-enabled workflow construction and execution*, BMC Bioinformatics, 6:69, 2005.

5. Rice P, Longden I, Bleasby A: *EMBOSS: The European Molecular Biology Open Software Suite*, Trends in Genetics, 16, pp. 276-277, 2000.

6. Chua Ching Lian , Tang F, Issac P, Krishnan A, *GEL: Grid Execution Language*, J Parallel and Distributed Computing, in press.

7. Pegasys web page, *http://bioinformatics.ubc.ca/pegasys*

8. Ezio Bartocci, et al., *BioWMS: a web-based Workflow Management System for bioinformatics*, BMC Bioinformatics, 8, 2007.

9. Romano P, et al., *Biowep: a workflow enactment portal for bioinformatics applications*, BMC Bioinformatics, 8, 2007.

10.BioPipe homepage, *http://www.biopipe.net*

11.D. Booth et al., *Web Services Architecture*, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211, 2004.

12.T. R. Gruber, *A translation approach to portable ontologies*, Knowledge Acquisition, 5(2):199-220, 1993.

13.이재윤, *온톨로지 기반 과제관리 및 분석체제 구축 방안 연구*, 한국문헌정보학회지, 2006

14.W3C RDF web page, *http://www.w3.org/RDF/*

15.W3C OWL web page, *http://www.w3.org/2004/OWL/*

16.W3C SWRL web page, *http://www.w3.org/Submission/SWRL/*

17.Gene Ontology web page, *http://www.geneontology.org/*

18.EcoCyc Ontology web page, *http://ecocyc.PangeaSstems.com/ecocyc/ecocyc.html*

19.Stevens R, et al., *TAMBIS: transparent access to multiple bioinformatics information sources.*, Bioinformatics. 16(2):184-5, 2000

20.Stoeckert CJ, et al., *The MGED Ontology: A Framework for Describing Functional Genomics Experiments,* Comp Funct Genomics. 4(1):127-32, 2003

21.BioPAX web page, *http://www.biopax.org*

22.T. Bray et al., *Extensible Markup Language (XML) 1.0,* http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

23.Grant, J. D., et al., *BeoBLAST: distributed BLAST and PSI-BLAST on a*

*Beowulf cluster*, Bioinformatics, 18, 765-766, 2001.

24.BeanShell web page, *http://www.beanshell.org/*

25.JRuby web page, *http://jruby.org/*

26.Jython web page, *http://jython.org/*

27.AJAX web wiki page, *http://ko.wikipedia.org/wiki/Ajax*

28.W3C HTML5 web page, *http://dev.w3.org/html5/spec/Overview.html*

29.Gerald Jay Sussman and Guy L. Steele, Jr. *Scheme: An interpreter for extended lambda calculus,* AI Memo, 349:19, 1975