Cross Web Application Integration through a Shared Database

Weijian Xuan[1,4], Terry Weymouth[4], Glenn Tarcea[4], Alex Ade[4], Zach Wright[4], Jim Cavalcoli[4], Barbara Mirel[2,4], H.V. Jagadish[3,4], Fan Meng[1,4]

[1]Psychiatry Department and Molecular and Behavioral Neuroscience Institute, [2]School of Education, [3] Computer Science and Engineering

[4]National Center for Integrative Biomedical Informatics, University of Michigan

Abstract

A large number of web applications are designed to deal with different aspects of biomedical data analysis requirements. A comprehensive data analysis task often requires functions from multiple applications and enabling different web applications to work seamless with each other is a major challenge. While there are different ways to achieve various levels of cross-application integration, we choose to use a common database for the sharing of intermediate data from multiple applications.

Our solution offers a number of unique advantages including 1) central location of intermediate archiving and project management, including sharing of intermediate results among pre-defined group members. 2) shared functions for manipulating intermediate data such as gene/protein id mapping from different applications, union, intersect and subtraction of different data lists 3) different applications only need to communicate with a single API and database rather than implementing application-specific solution for each new application that needs integration.

Web Application Integration Overview

From architecture perspective:

- Web system integration
  - Sub web systems into one system
  - Sub web systems function together
- Web application data integration
  - Multiple online data sources
  - Online (session, dynamic) and offline
- Service-oriented architecture
  - Around task-oriented processes
  - Package into interoperable services
  - Loose coupling of services

From functionality perspective:

- Presentation Layer
  - A variety of user interface across applications
  - Browser-based GUI, in various frameworks
- Data Layer
  - Simple ones: access to remote data sources
  - Advanced: integrate/analyze data sources
- Functional Layer
  - Often lightweight app, focus on specific areas,
  - One-application does not fit all
  - A cross app integration pipeline is desirable

Considerations & Requirements in Designing Integration

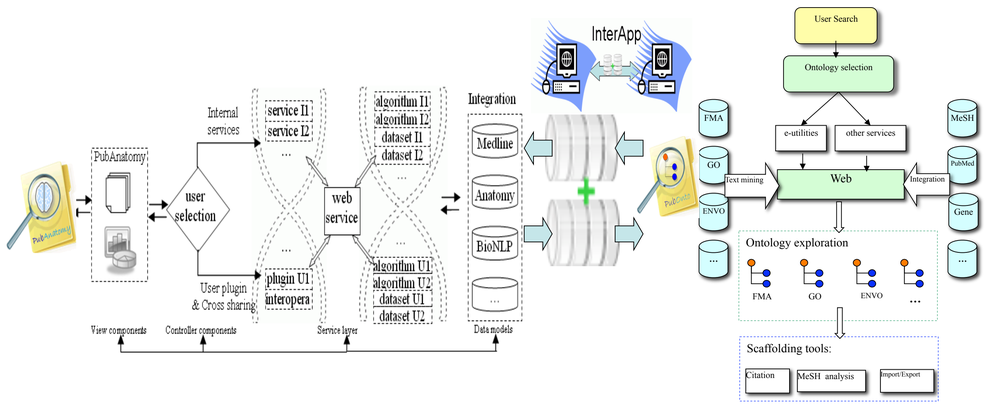
Considerations

- Internal
  - Implementation, technical complexity
  - Specific applications developed by each development group
  - Pipeline across different NCIBI groups
  - Different development priorities
- External:
  - Utilization, functional, usability
  - Across whole Internet in general
  - Large number of parties and users
  - Unexpected settings:
  - Network, permission, firewall

Requirements

- Use cases:
  - Nonlinear, explorative, repetitive
  - Diversified functional requirements
  - Long time spanning, incremental knowledge
- Data types
  - Simple (GeneIDs, PMIDs, MeSH, Scores, etc)
  - Reusable (data common to many applications)
- Development setup:
  - Heterogeneous and changing (prototyping)
- Application scope:
  - Target on different type of research and analysis

Fig. 1. Integration Architecture Example



Acknowledgement: W. Xuan and F. Meng are members of the Pritzker Neuropsychiatric Disorders Research Consortium, which is supported by the Pritzker Neuropsychiatric Disorders Research Fund L.L.C. This work is also partly supported by the National Center for Integrated Biomedical Informatics through NIH grant 1U54DA021519-01A1.

Integration Architecture: Our Choice

✓ **Integration through shared database and core web services**

Pro: Low dependency, high flexibility, no bottleneck, computable datasets, friendly to typical multi-session research use cases, loose coupling, low complexity, and expandable

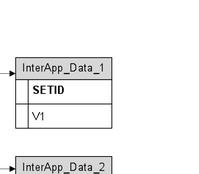
Con: may require more user actions but often just a few clicks

We chose it over the following approaches because:

× Session-based techniques

Pro: integrated operation, easy to use (if work, often times not)

Con: unstable, lose data when session ends



× Embedded approaches

Pro: integrated UI, better usability for certain type of tasks

Con: Only fit in certain type of tasks, higher level of dependency

× Web-services only architecture

Pro: standard, loose coupling

Con: pre-fixed scenarios, bottleneck in collaborative development environment, fragmented datasets, not analysis friendly, hard to adapt in heterogeneous development environment

Design and Implementation

Principles:

- ✓Agreement on data sharing approach
- ✓Loose Coupling among apps
- ✓Encapsulation of core function
- ✓Composability: built larger system
- ✓Abstraction of underlying services
- ✓Performance: minimize overhead
- ✓Usability: simple service calls
- ✓Flexibility: adaptive to various NCIBI data sharing requirements

Components:

- ❖Database schema is developed and deployed
- ❖Syntactic/semantic interoperability among NCIBI web applications
- ❖Centralized database repository for sharing
- ❖Application-independent dataset operations
- ❖Shared core dataset operating web services, e.g., read, write, and some dataset operations
- ❖Group level and individual level dataset sharing
- ❖Interoperability: app pipeline

We have created a web-based API for controlled access to the server supporting these functions: save data set, list all saved data sets, review the content of a data set. NCIBI is developing a common set of core services for external and internal integration of tools. We also have a demo session for a workflow that involves Gene2MeSH, PubAnatomy and MiMI database developed in NCIBI.

Fig. 2. Core Part of the Sharing Schema

Fig. 3. Research Pipeline Using Multiple NCIBI Tools through Data Sharing

Example Pipeline: Gene2Mesh ⇔PubAnatomy ⇔ PubPath⇔ PubIO ⇔ MiMi-Cytoscape