Biomedical Software Engineering: BMI2002

Fall 2017
Mount Sinai School of Medicine
Prof. Arthur Goldberg

Assignment 2: Design and Document the Architecture of a System

Due Sun., Oct. 22

Logistics

Hand in via email to the instructor. Maximum length is 8 pages, including figures and tables. Type must be at least 11 pt., and all margins must be at least 1 inch.

Introduction

Design and document a system's architecture. Write no code.
The system will analyze and visualize protein-protein interaction (PPI) networks, so I've named it AVPPIN.

AVPPIN's Requirements

This section documents AVPPIN's requirements.

Users: AVPPIN will be used by one type of user:

1. Bioinformaticians who assist research geneticists with their analyses.

Inputs: AVPPIN inputs three types of data.

1. Protein classifications. This data classifies proteins as members of certain genetic pathways or biological activities, e.g., as provided by the Gene Ontology system. These are public data that rarely change.
2. AVPPIN inputs PPI networks. A PPI network represents each protein as a node labeled with its name, and a relationship between a pair of proteins as an edge between their nodes. The relationships are categorical. One kind of relationship is 'interact', which is represented as an undirected edge. Other relationship are up-regulates and down-regulates, which are represented as directed edges from the regulatory protein to its target.

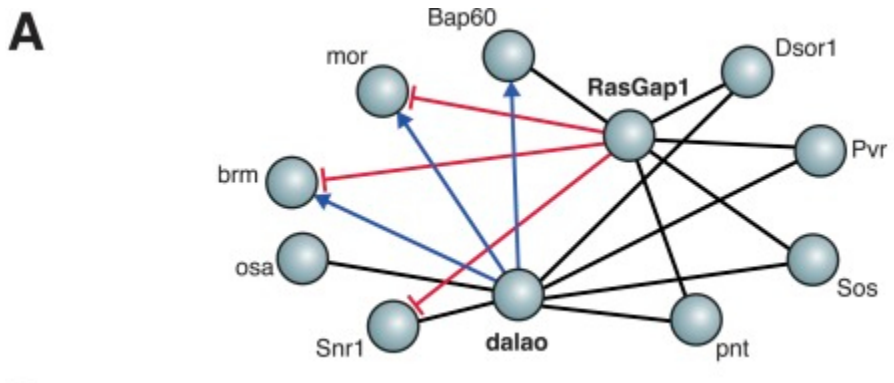

Fig 1. This network represents protein protein interactions in all three categories [Fischer 2015].

Assume that the PPI networks contain no HIPAA 'Protected Health Information'.
Two types of networks are used:

a. Reference PPI networks for particular human tissues obtained from populations of unaffected individuals, which I call 'unaffected networks'. These are public data that rarely change.
b. One or more PPI networks for the same tissue obtained from samples of subjects who have an illness -- affected individuals. These are called 'affected networks'. These are private data that belong to individual users of AVPPIN.

3. AVPPIN also inputs metadata about users and their use of AVPPIN. This includes user credentials, and the names of the studies they perform and the files that they input.

Features: AVPPIN will provide these functional features.

1. Users access AVPPIN from browsers. They can create user accounts, log in and out, upload input data, and perform, save and view analyses.
2. Detect errors in uploaded PPI networks, such as improperly formatted files and unknown proteins.
3. Perform population genetics with PPI networks: Compare affected networks with unaffected networks for the same tissue.
    a. Statistical analyses: Does an affected network differ statistically from an unaffected network? Is a sub-network of proteins that differs between an affected network and an unaffected network over or under-represented in protein classification categories? Other analyses will be added as needed.
    b. Visualization: Networks can be visualized in a user's browser.
4. The analyses and visualizations can be viewed in the browser, and downloaded in standard formats for networks, and images.

Unfortunately, more detailed requirements are not available at this time.

Architecture document

The only deliverable for this assignment is an architecture document. It should contain the following sections:

- Functional overview
  Summarize the key functions of the system, answering these questions:
  1. What does the system do?
  2. Who are the important users (what is their role) and how does the system satisfy their needs?
- Software architecture
  Address these specific architectural topics:
  1. Describe the overall organization of AVPPIN. What are its major components? Explain how they support the inputs and features described in the requirements.
  2. Which components can reuse existing software, and which should be custom-built? Why? What high-level functions are supported by the custom-built component(s)? Propose specific existing Python packages that would be reused.
  3. If a database is needed, what tables and foreign key relationships should it support? (The remaining fields for each table need not be provided.)
  4. What security should AVPPIN provide?
  5. Provide a block diagram of architectural components and their interactions. Identify each component at two levels: an abstract role and a specific tool. E. g., a web server might be identified as "Webserver: Apache version X".

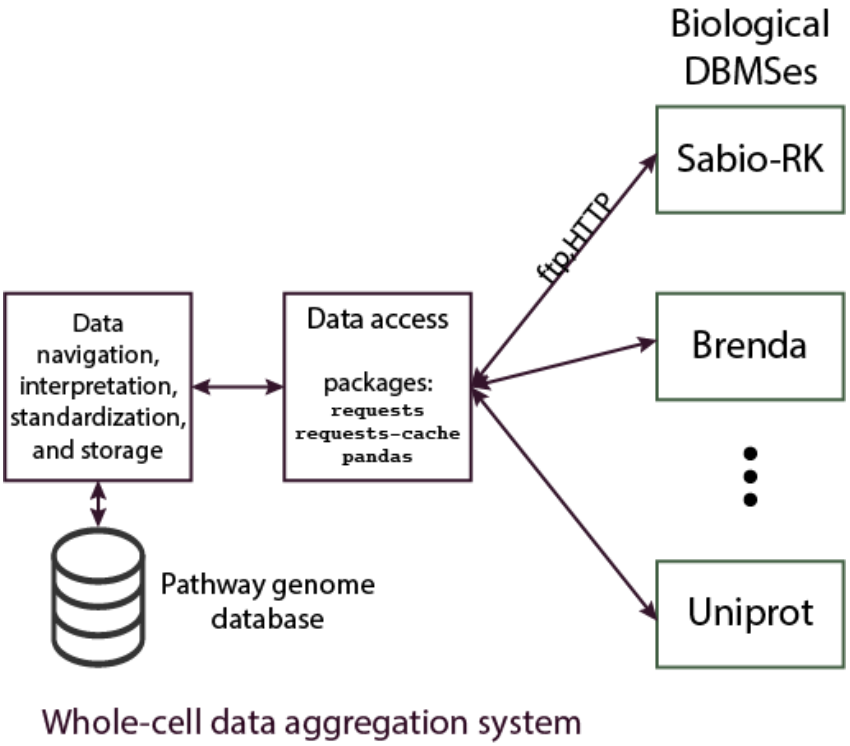

Fig. 1: Example architecture diagram. This diagram presents the architecture of a data aggregation system that downloads data used by whole cell modeling. The data is obtained from a set of biological databases accessible on the Internet, via ftp and/or HTTP. A 'Data access' module uses the listed packages and custom code to request and download data. A custom module (left) uses this data to navigate the databases, request more data, and interpret

and standardize the downloaded data, and then store it in a Pathway Genome database, for use later by whole-cell models. This custom module uses bioservices, numpy, openbabel, openpyxl, pandas, pronto, and pubchempy.

> Discuss these high-level and quality issues:
>
> 1. Which areas of the Requirements should be refined or clarified to enable more specific architectural decisions?
> 2. What non-functional requirements, beyond security needs, would be helpful to add to the Requirements?

Guidelines

With respect to the statistical analyses, the architecture does not need to include the design or implementation of the analyses. You may just assume that custom code which uses a well-known statistical library performs the analyses.
With respect to network visualization, select a widely used protein network visualization library to perform the visualization.

Further reading

Martin Fowler's thought provoking article, *Who needs an architect?*, quotes a definition of software architecture proposed by Ralph Johnson that I like:

> In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.' This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers.

But this type of architecture cannot be documented until after the code has been written.

Notes

This course's **Collaboration rules**, as written in the course description, apply to this assignment.

References

Fowler, M., 2003. Who needs an architect?. IEEE Software, 20(5), pp.11-13. pdf
Fischer, B., Sandmann, T., Horn, T., Billmann, M., Chaudhary, V., Huber, W. and Boutros, M., 2015. A map of directional genetic interactions in a metazoan cell. Elife, 4, p.e05464.