# DISBi: A flexible framework for integrating systems biology data

Rüdiger Busche[1][0000−0002−9593−8696], Henning Dannheim[2], and Dietmar Schomburg[2][0000−0002−3354−822X]

[1] Osnabrück University
Institute of Cognitive Science
rbusche@uos.de
[2] Technische Universität Braunschweig
Department for Bioinformatics and Biochemistry
{h.dannheimm, d.schomburg}@tu-braunschweig.de

**Abstract.** Systems biology aims at understanding an organism in its entirety. This objective can only be achieved with the joint effort of specialized work groups. These collaborating groups need a centralized platform for data exchange. Instead data is often uncoordinatedly managed using heterogeneous data formats. Such circumstances present a major hindrance to gaining a global understanding of the data and to automating analysis routines.
DISBi is a framework for creating an integrated online environment that solves these problems. It enables researches to filter, integrate and analyze data directly in the browser. A DISBi application dynamically adapts to its data model. Thus DISBi offers a solution for a wide range of systems biology projects.
An example installation is available at disbi.org. Source code and documentation are available from https://github.com/DISBi/django-disbi.

**Keywords:** Systems biology · Data integration · Data exchange.

## 1 Introduction

In systems biology, researchers try to gain a system-level understanding of an entire organism. This objective requires investigating organisms from different perspectives involving diverse experimental and computational approaches [6]. As the different approaches usually require specialization, work groups bundle their efforts in consortia, each work group investigating a different level of biological information. The huge amounts of data generated in different experimental and simulation approaches need to be integrated to reveal the underlying patterns. While data exchange is a crucial aspect of this effort, it is often hindered by a lack of standardized data formats and a centralized data storage. Though different standards exist [10], they are often not used consistently throughout the project due to inflexibility of the formats or diverging preferences of different work groups [7]. These circumstances greatly hinder the progress towards

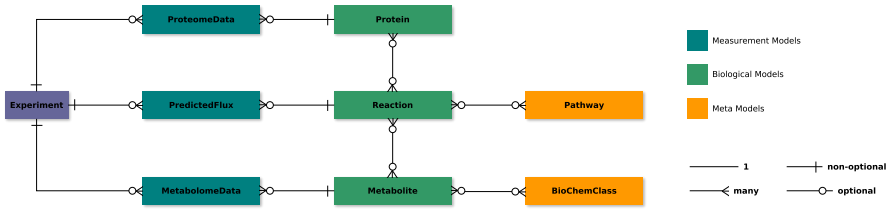


**Fig. 1. Exemplary layout of the abstract data model**. A possible data model for integrating proteome, flux balance analysis and metabolome data. Reactions are grouped in pathways and metabolites in biochemical classes by utilizing meta models. Every measurement model has a non-optional relation to one *Experiment* and one biological model. Note that this is only one possible data model for a DISBi application. The abstract data model can accommodate varying numbers biological and measurement models with more complex relationships.

system-level understanding, as complete data are seldomly available and often need to be integrated manually.

With DISBi, we present a framework to easily construct an online environment for ***D**ata **I**ntegration* in ***S**ystems **Bi**ology*. It enables users to manage, organize and integrate heterogeneous data generated in systems biology projects. Existing systems like *Aureolib* [3] rely on fixed data models, which makes the applications useful only for a narrow range of projects. Often these applications are limited to manage only data from a single organism or a single experimental method. Instead, DISBi's approach is to generalize the problem of data integration in systems biology. Thus DISBi is a framework rather than an application that allows users to set up a web app customized for the data of a particular project. Programmers can focus on the logic relevant for the project, rather then implementing details. Researchers profit from the availability of integrated data throughout the course of a project.

## 2 Data Model

When setting up a new instance of a DISBi application for a systems biology project, only a new data model has to be defined. This data model should capture all information relevant to the project. The defined data model needs to adhere to DISBi's *abstract data model*, which is a relational data model with additional constraints. These constraints serve the purpose of imposing some structure on the data model that can be exploited by the framework, while giving the developer the greatest possible freedom in modeling the domain of the project. In the abstract data model, data is split into three categories: *biological models*, *measurement models*, *meta models* (Fig. 1). While these are abstract categories a concrete *Experiment* model is necessarily part of the data model as well. Every data source (simulation or wet-lab experiment) is stored in the Experiment model. The Experiment model should include all meaningful experimental

parameters, but can also accommodate free input labels. What exactly these parameters need to be ultimately depends on the design of the investigation. We find it useful to include only experimental parameters that were varied across the investigation, e.g. carbon source, and meta data such as the experimental method used to generate the data.

Measurement models store data points generated in an experiments, such as the response from a mass spectrometer. Biological models store the biological entities the data points map to, such as proteins or metabolites. Meta models store information about these biological entities, such as pathways or functional groups. The data model needs to be designed such that each instance of a measurement model can uniquely be identified by its relation to an experiment and a biological object. Thus it is representing data about a certain biological object measured in a certain experiment. The biological models can be related in arbitrary ways, but must not create cyclic relations. For example a protein maps to a reaction that maps to a metabolite. These relations define an implicit relation between proteins and metabolites, but an additional explicit relation between a reaction and a metabolite would introduce the possibility of inconsistency and is therefore forbidden by the abstract data model.

## 3 Data Integration

Data of biological models and their relations have to be determined and uploaded to the system before it is used for integrating experimental data. These data form the backbone of the data integration process and can be conveniently uploaded through the admin interface in tabular formats. Relations of the biological models are included in the tabular data by including columns of unique identifiers of related biological objects. The naming and relations of the biological objects should be agreed upon by all partners in the project as it is only possible to upload measurements that map to biological objects that are included in the system. Thus the biological objects serve as a kind of controlled vocabulary. Adapting the data model during the project can be achieved by using Django's built-in migration framework.

Data integration is done based on the relations of the biological models by combining the results from all matched experiments in a dynamic table. The relations between the models are dynamically inspected at runtime. Data points from different experiments that map to the same biological object as well as data points that map to related biological entities are combined in one row. For example, transcriptome and proteome data will be presented together, if the respective protein derives from the respective gene. This format makes it easy to analyze the data for correlations between related biological entities and to test predictions from simulations. Meta models play no role in data integration, but simply function as a container for data that cannot be put on the biological models due to normalization constraints. That means that while the information from meta models is included in the dynamic table, biological objects are not joined together based on information in the meta models.

## 4 Usage

A DISBi application is split into two main interfaces. The *Filter View* provides the user with a tool to find experiments of interest based on experimental parameters. It is automatically constructed from the underlying data model. The user is shown a preview of the matched experiments, which allows for interactive exploration of available experiments. Once a set of experiments of interest is determined, the user is taken to the *Data View*. In this view, the integrated experimental data is presented in a table and can be further filtered, exported and analysed. The analysis capabilities include calculating fold changes, plotting distributions of single columns and comparing two experiments in a scatter plot. These functionalities provide the users with a tool to quickly get an overview of the datasets and determine which data are worthwhile for in-depth analysis.

The DISBi framework supports an easy to use, customizable admin interface that facilitates uploads of large datasets from common file formats such as Excel, CSV or JSON.

Extended version of Django model classes are used to define the data model. This provides a high-level interface for specifying the data base scheme that does not require a deep understanding of the underlying database structures and can be accomplished by everyone with basic Python proficiency. As the abstract data model puts no constraint on the number of models or fields, it can accommodate a great variety of different project outlines.

## 5 Experiences

DISBi was successfully applied to internal projects in the Department for Bioinformatics and Biochemistry at the Technische Universität Braunschweig for integrating data from the organisms *Chlostridium difficile*, *Aromatoleum aromaticum* EbN1 and *Dinoroseobacter shibae* as well as for public data from *Sulfolobus solfataricus* [12]. The integrated data sources include transcriptome, proteome, metabolome and predicted metabolic flux data. The integrated methods range from RNA sequencing and mass spectrometry to microarray data and simulations. This demonstrates the applicability of DISBi to a wide range of different data.

## 6 Implementation

The DISBi framework is designed with Python 3.6 (python.org) as backend language based on the Django web development framework (v1.11 djangoproject.com). PostgreSQL (postgresql.org) is used as database backend. Interactive data visualization is implemented with NumPy [11] and matplotlib [4].

The jQuery (jquery.com) JavaScript library and SASS (sass-lang.com) preprocessor are used to provide a responsive user interface with a consistent visual appearance.

By using only open source technologies, we ensure that the DISBi framework is freely available and extendable. Moreover, by using Python future developers of DISBi will have access to the vast ecosystem of scientific software available in Python [5] for extending DISBi with new features.

## 7 Related Work

Many systems exist that tackle integrating heterogenous data in multi work group system biology projects [14]. The most prominent systems are DERIVA [2], openBIS [1] and the SEEK platform [13] together with the ISA toolchain [8]. These platforms are well established and go far beyond the current scope of DISBi. They offer functionality for automatically uploading data to the system when they are produced at the measurement device and ultimately enable the user to deploy integrated data in public repositories. In contrast to DISBi they focus on *asset management*, i.e. attaching meta data to arbitrary data sources and storing these data sources in a unified fashion. DISBi is more focused on establishing correspondences between single data points. Hence, larger systems such as DERIVA can accommodate more heterogeneous data and are therefore applicable to very large projects, while DISBi is restricted to tabular data.

DISBi should therefore be seen as a lightweight complementary approach that can be used to integrate managed assets in a more fine-grained manner.

The importance of having the system dynamically adapt to its data model is highlighted in the design of DERIVA [9]. This ensures that the system is applicable to a wide variety of different projects. DISBi follows the same design philosophy.

## 8 Conclusion

With DISBi, we present a powerful framework for the construction of custom data integration platforms for systems biology projects. Its flexibility and customizability render it an applicable solution for managing data in small to mid-sized projects and making data publicly available. The source code is freely available under terms of the MIT license, which allows other developers to modify and evolve the software. We will continue to explore possible ways of automating data integration and analysis based on the abstract data model.

## Acknowledgements

The authors thank Meina Neumann-Schaal for critical reading of the manuscript and four anonymous reviewer for their instructive comments. Rüdiger Busche thanks Pascal Nieters for support in the publication process.

This work was supported by the Federal State of Lower Saxony, Niedersächsisches Vorab (VWZN2889)/3215.

## References


1. Bauch, A., Adamczyk, I., Buczek, P., Elmer, F.J., Enimanev, K., Glyzewski, P., Kohler, M., Pylak, T., Quandt, A., Ramakrishnan, C., Beisel, C., Malmstrom, L., Aebersold, R., Rinn, B.: openBIS: a flexible framework for managing and analyzing complex data in biology research. BMC Bioinformatics **12**(1), 468 (2011). https://doi.org/10.1186/1471-2105-12-468
2. Bugacov, A., Czajkowski, K., Kesselman, C., Kumar, A., Schuler, R.E., Tangmunarunkit, H.: Experiences with DERIVA: An asset management platform for accelerating eScience. In: Proceedings - 13th IEEE International Conference on eScience, eScience 2017. pp. 79–88 (2017). https://doi.org/10.1109/eScience.2017.20
3. Fuchs, S., Zühlke, D., Pané-Farré, J., Kusch, H., Wolf, C., Reiß, S., Binh, L.T.N., Albrecht, D., Riedel, K., Hecker, M., Engelmann, S.: Aureolib — A Proteome Signature Library: Towards an Understanding of *Staphylococcus aureus* Pathophysiology. PLoS ONE **8**(8), e70669 (2013). https://doi.org/10.1371/journal.pone.0070669
4. Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing in Science and Engineering **9**(3), 99–104 (2007). https://doi.org/10.1109/MCSE.2007.55
5. Jones, E., Oliphant, T., Peterson, P.: SciPy: open source scientific tools for Python (2014)
6. Kitano, H.: Systems Biology: A Brief Overview. Science **295**(5560), 1662–1664 (2002). https://doi.org/10.1126/science.1069492
7. Lubitz, T., Hahn, J., Bergmann, F.T., Noor, E., Klipp, E., Liebermeister, W.: SBtab: A flexible table format for data exchange in Systems Biology. Bioinformatics **32**(April), btw179– (2016). https://doi.org/10.1093/bioinformatics/btw179
8. Rocca-Serra, P., Brandizi, M., Maguire, E., Sklyar, N., Taylor, C., Begley, K., Field, D., Harris, S., Hide, W., Hofmann, O., Neumann, S., Sterk, P., Tong, W., Sansone, S.A., Wren, J.: ISA software suite: Supporting standards-compliant experimental annotation and enabling curation at the community level. In: Bioinformatics. vol. 27, pp. 2354–2356. Oxford University Press (2011). https://doi.org/10.1093/bioinformatics/btq415
9. Schuler, R.E., Kesselman, C., Czajkowski, K.: Accelerating data-driven discovery with scientific asset management. In: 2016 IEEE 12th International Conference on e-Science (e-Science). pp. 31–40. IEEE (2016). https://doi.org/10.1109/eScience.2016.7870883
10. Taylor, C.F., Field, D., Sansone, S.A., Aerts, J., Apweiler, R., Ashburner, M., Ball, C.A., Binz, P.A., Bogue, M., Booth, T., Brazma, A., Brinkman, R.R., Michael Clark, A., Deutsch, E.W., Fiehn, O., Fostel, J., Ghazal, P., Gibson, F., Gray, T., Grimes, G., Hancock, J.M., Hardy, N.W., Hermjakob, H., Julian, R.K., Kane, M., Kettner, C., Kinsinger, C., Kolker, E., Kuiper, M., Novère, N.L., Leebens-Mack, J., Lewis, S.E., Lord, P., Mallon, A.M., Marthandan, N., Masuya, H., McNally, R., Mehrle, A., Morrison, N., Orchard, S., Quackenbush, J., Reecy, J.M., Robertson, D.G., Rocca-Serra, P., Rodriguez, H., Rosenfelder, H., Santoyo-Lopez, J., Scheuermann, R.H., Schober, D., Smith, B., Snape, J., Stoeckert, C.J., Tipton, K., Sterk, P., Untergasser, A., Vandesompele, J., Wiemann, S.: Promoting coherent minimum reporting guidelines for biological and biomedical investigations: The MIBBI project (2008). https://doi.org/10.1038/nbt.1411
11. Van Der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: A structure for efficient numerical computation. Computing in Science and Engineering **13**(2), 22–30 (2011). https://doi.org/10.1109/MCSE.2011.37

12. Wolf, J., Stark, H., Fafenrot, K., Albersmeier, A., Pham, T.K., Müller, K.B., Meyer, B., Hoffmann, L., Shen, L., Albaum, S.P., Kouril, T., Schmidt-Hohagen, K., Neumann-Schaal, M., Bräsen, C., Kalinowski, J., Wright, P.C., Albers, S.V., Schomburg, D., Siebers, B.: A systems biology approach reveals major metabolic changes in the thermoacidophilic archaeon Sulfolobus solfataricus in response to the carbon source L-fucose versus D-glucose. Molecular Microbiology (2016). https://doi.org/10.1111/mmi.13498
13. Wolstencroft, K., Owen, S., Du Preez, F., Krebs, O., Mueller, W., Goble, C., Snoep, J.L.: The SEEK: A platform for sharing data and models in systems biology. Methods in Enzymology **500**, 629–655 (2011). https://doi.org/10.1016/B978-0-12-385118-5.00029-3
14. Wruck, W., Peuker, M., Regenbrecht, C.R.A.: Data management strategies for multinational large-scale systems biology projects. Briefings in Bioinformatics **15**(1), 65–78 (2014). https://doi.org/10.1093/bib/bbs064