Data Mining for Unidentified Protein Sequences

Leif Blaese

Potsdam University, Potsdam, D-14482, Germany
blaese@uni-potsdam.de

Abstract. Through the use of next generation sequencing (NGS) technology, a lot of newly sequenced organisms are now available. Annotating those genes is one of the most challenging tasks in sequence biology. Here, we present an automated workflow to find homologue proteins, annotate sequences according to function and create a three-dimensional model.

1 Introduction

The advance in DNA sequencing technology over the last years has enormously increased the number of newly sequenced genes. With the advent of the '-omics' technologies as genomics, metagenomics, proteomics, bioinformatics has discovered a number of problems to deal with - one of the most important to effectively manage these data and create biological context for them by creating links between known and unknown data to infer (e.g. in case of protein sequences) evolutionary history, function or even structure of the unknown proteins.

Sequences are usually stored in international Databases, for example GenBank [4], the Nucleic Acid Database of the National Institute of Health (NIH). It has more than 135 million sequence records (as of July 2012). In contrast, the PDB [6], the RCSB Protein Data Bank that stores known structure of proteins, has only about 81 thousand structures (as of July 2012). This shows the imbalance between known sequences and the amount of information *known about* these sequences.

The workflow presented here tries to address this problem. It can be used to automatically mine for information about an unknown protein sequence. Consider for example a metagenomics experiment, where the complete genomic content of an environment sample is analyzed to examine the proteins present (for the sake of this example) [44]. After preprocessing of the data one has a number of protein sequences that have to be assigned a biological context - where they are from, what their function is, etc.

2 Service Analysis

This section introduces the methods and the tools that have been used for the workflow. Note that only online services providing their functionality have been used for the actual realization. The reason for that is, that online services provide lots of computational power while keeping the cost of running for the user small.

A.-L. Lamprecht et al. (Eds.): Process Design for Natural Scientists, CCIS 500, pp. 73–87, 2014.
DOI: 10.1007/978-3-662-45006-2_6

Blast, MaFFT and InterProScan are available at the EBI [16], Phyre is available at the website of the structural bioinformatics group of the Imperial College London [14]. The EBI provides RESTful as well as SOAP-Services. Here, only the REST-services are used.

Technically, the services are accessed from jABC via an *ExecuteCommand*-SIBs that executes cURL within a shell script. The results are downloaded and stored on the user's PC. As shell scripts are used, the user has to run Linux with cURL already installed. cURL is open-source and available at `http://curl.haxx.se/download.html`. The shell scripts should be copied to a folder called 'nat2' in your jABC-directory. In general, the online shell scripts use the default settings of the respective tool. This should give good results in general, but users should be advised that changing these options in the scripts may improve them further. The default values can be found on the web-pages of the EBI. Phyre does not give the option to change the default values.

2.1 Sequence Alignments

Two sequences are homologous, if they are derived from a common (sequence) ancestor. They are further called orthologs, if they share their ancestry due to a speciation event or paralogs if they share it due to gene duplication. These homologous sequences provide lots of information about the unknown gene or protein at hand. The idea is that the genetic code changes over time and the parts of the code that stay the same must hold some valuable information. Consider a speciation event, where two new species emerge from a common ancestor. At first their set of genes will be similar, but after a while (on an evolutionary time scale) their genetic code will drift apart due to accumulation of mutations. Most of those mutations do not occur in protein coding regions and even if they do, they have a certain probability of not changing the protein sequence due to the degeneracy of the genetic code. The protein sequences are thus in general more similar than the gene sequences. Furthermore, a lot of mutations will not change the form or function of the protein (at least not by much) insertions or deletions for example are often in loop regions whereas the core region of a protein or its secondary structure is more conserved. Also, amino acids that have a special role in the proteins function (e.g. binding sites for substrates in enzymes or docking sites for cofactors) are usually highly conserved, they cannot be changed without disturbing the function of the protein. If you have an unknown protein sequence you can infer the structure and function from its homologous' proteins provided those are known. Because of the higher conservation of the amino acids one generally compares proteins on the amino acid-level instead of the gene level. When mentioning sequences, here I refer to protein sequence unless mentioned otherwise.

2.2 Optimal Pairwise Alignment

Homologous sequences are usually found by a pair wise comparison (alignment) between the sequences. They are scored and if the score is above a certain threshold

and the finding is statistically significant, the genes are said to be homolog. There are various tools to achieve a pairwise sequence alignment, the probably best known tool are based on the Needleman-Wunsch algorithm [34], which produces an optimal global alignment of two proteins, and on the BLAST algorithm [2], which is usually faster but relies on heuristics.

The Needleman-Wunsch algorithm was developed in 1970 by S. Needleman and C. Wunsch. It uses a dynamic programming approach to find an optimal global alignment of two proteins. The algorithm consists of three main parts: Creating the dynamic programming table, filling it and backtracking to find the optimal alignment. The dynamic programming table is a two-dimensional matrix of the size $m + 1 \times n + 1$ with m, n being the length of the sequences. The value of a field (i, j) in this matrix is computed as the maximum of either $(i-1, j-1)$ plus some weight w, $(i-1, j)$ or $(i, j-1)$. The values are thus computed from the top left to the bottom right field of the table. On Backtracking, the Needleman-Wunsch starts in the bottom right field and goes the other way back to the upper left field of the table; always choosing the path that maximizes the overall achieved score. The chosen path then represents the alignment of the two sequences. If there are multiple equal paths possible, all those alignments are equal.

2.3 Substitution Matrices and Gap Penalties

The weight w that is added is determined by a substitution matrix, for example the BLOSUM or the PAM matrix. Those substitution matrices score the probability of a mutation of one amino acid to another. Due to the different shape and chemical nature of the amino acids not all mutations are even likely. Glycine for example is the smallest amino acid; its side chain consists of only one hydrogen atom. It is very agile and often found in loop regions on the outside of a protein, for example between two alpha helices. Phenylalanine on the other hand is a considerably large amino acid that has a benzene-derivate as side chain. A mutation that would lead to a change from glycine to phenylalanine would have dramatic impacts on the structure of a protein. However, because of the different nature of the two amino acids, this mutation is very unlikely and has a low score in the substitution matrix. The BLOSUM-Matrix (Blocks of Amino Acid Substitution Matrix) calculates the scores as the logarithmic ratio of the probability of the two amino acids appearing in a protein together and the probability of either of the amino acids appearing in the proteins alone. There are different BLOSUM matrices labeled by the percentage of identity of the sequences used to create the matrix. The default value for a lot of programs and services is the BLOSUM62 matrix, which was computed from proteins having at least 68% sequence identity.

The PAM matrix (Point Mutation Accepted, accepted mutations per 100 residues) is a similar substitution matrix, but it was created differently. Here, a phylogenetic tree was build using highly related sequences. The ratio of the number of changes of each type of amino acid to the total number of occurrences of that amino acid in the set of sequence is used as the score [40].

Sequences often change their length with increasing evolutionary distance. This is the result of insertions or deletions of amino acids. In an alignment, those mutations are seen as gaps in one of the sequences. To limit the introduction of gaps, a gap penalty is used. In the Needleman-Wunsch algorithm, this is implicitly done, when the maximum value of a field is equal to the value of the field to the left or top of it. The gap penalties used to score gaps are important; if a gap is given a lower score, more gaps are likely to be found, if a higher score is given, less will be found. Usually, different gap scores are given for creating, extending and closing a gap.

A progression of the Needleman-Wunsch algorithm is the Smith-Waterman algorithm that produces optimal local alignments of two proteins. It has a slightly changed backtracking, so it does not only find one optimal global alignment, but many optimal aligned pieces.

These algorithms produce good results; however, they are slow as they have to compute a full alignment of two sequences. The runtime of the (serial) Needleman-Wunsch Algorithm is $\mathbf{O}(n^2)$. Consider a database with millions or sequences that one unknown sequence has to be compared against. This is not a trivial task and requires an algorithm with maximal speed to be done in an acceptable time.

2.4 Blast

BLAST uses a different approach and is usually faster than Needleman-Wunsch. As it uses heuristics, it cannot provide an optimal alignment, statistics can be used to counteract and score the found alignment based on the significance of appearance in the searched database.

Blast searches in its initial state for small identical words in the database. A word in this respect is a k-mer of identical residues of the two sequences that would score above a certain threshold T if it was aligned with the query k-mer. It uses finite-state automata to search these words. The input of the automaton is the sequence splitted in its symbols; for each possible input (each combination of amino acids) there is a defined response from the automaton. The score is computed using a substitution matrix such as BLOSUM62 and explicit gap penalties. The query sequence is divided in all possible k-mers and all words in the complete database are found. Originally, these words were then extended in both directions until the score drops off by some amount. Newer version of the Blast-Algorithm use a so called two-hit method: The idea behind that is that the significant alignments have probably more than one word in their alignment. The algorithm searches for a second word within the realm of a certain distance. If a second word is found, it is ungapped extended. If this extension does not yield a score above a threshold S_x, it is discarded. If it does, a dynamic programming approach is used to determine a gapped alignment.

If a DNA sequence is to be compared to an amino acid sequence, one has to pay attention to the reading frame. There are six possible ways to translated proteins; if there are additional insertions or deletions due to sequencing error, frame shifts can occur. For the purpose of this work, I consider only one reading frame and assume that all errors are eliminated.

As mentioned earlier, the Blast-Score can be used to find a significant alignment in the given database. The total Score S of an alignment is the sum of its pairwise scores. From that, the so called bit-score S can be computed, a normalized score that is a function of the raw score, the scoring matrix, and the amino acid frequency in the database $S = \frac{(\lambda S) ln(K)}{ln(2)}$ With λ being a constant depending on the substitution matrix, K being a constant depending on the amino acid frequency in the database. The expected value of S (so called E-Value) is then $E = Kmne^{\lambda S}$ with m, n being the length of the sequences. This is the expected number of database hits with a score at least S, the observed bit-score. The smaller the E-Value, the better: Alignments with a small E-Value are not likely to be found by chance in the database; this indicates that biological significance might be the crucial factor to explain why an alignment scores so well. Typically, alignments that achieve an E-Value below 1^{-10} are said to be homologous.

2.5 MaFFT

Instead of incorporating information of just two sequences, one could look at multiple sequences to find conserved regions. It seems intuitive that a residue that is conserved in multiple sequences must be valuable to the protein, whereas a residue that is only conserved in two of the many sequences may not be as valuable something might not be detected when looking at only two sequences. There are many programs that compute a multiple sequence alignment. The program used here is called MaFFT [13], Multiple alignment by Fast-Fourier Transformation.

MaFFT represents each amino acid in the sequence as a vector consisting of the polarity and the volume of the amino acid. The vectors are then normalized and the correlation between two sequences based on those vectors is computed. This computed correlation peaks, if the sequences have homologous regions. These peaks are used to find homologous regions, which guide a dynamic programming approach to find an alignment: With the information that the alignment must pass some explicit regions, the dynamic programming table can be reduced and the computation time can be improved. Reducing the dynamic programming table means dividing it into sub-matrices as indicated by the homologous regions and only aligning those regions. After computing this initial alignment, it is iteratively refined [5]

MaFFT uses a scoring matrix based on the previously discussed PAM substitution matrix, but normalized by the frequency of occurrence of the amino acids. To determine the order of the alignments that are to be aligned, a distance is calculated based on the number of 6-tuples shared by two sequences. From this distance matrix, a guide tree is built. This guide tree is computed by using the UPGMA method (Unweighted Pair Group Method with Arithmetic mean)[42]. Building such a tree is fast, but it infers a uniform rate of evolution, an assumption which is not necessarily true. Still, MaFFT performs very well and is faster than most of its competitor programs [12].

2.6 InterProScan

The information known about protein sequences is stored in the databases like the UniProt-Database, especially the manually curated UniProt Knowledgebase (UniProtKB). UniProt is the central resource for protein-related information. Protein sequences can be used to query this database to find information for example about its function, substrate specificity (in case of an enzyme), location, GO-Terms, etc.

A more sensitive approach is not to search every single protein but create protein signatures, mathematical descriptions of the sequence or structure of the protein, and search with the help of that signature for protein families. These signatures can be regular expressions used in Prosite [41], build from multiple sequence alignments [9] or based on Hidden Marko Models [15]. Although there are many databases, they all have their main focus on different things - using only one in a query could lead to bad results because the analyzed protein did not lie in the focus area of the database. This problem was addressed by InterPro, a database combining now more than ten of these signature based databases [10]. The InterPro database holds various information about the information, one of them a list of precomputed mathes to the UniProtKB database. Using the InterProScan tool (IPS) [46], a sequence can be scanned against the Interpro database to find the matching InterPro entry and thus the UniProtKB entries.

2.7 Phyre

Proteins consist of four stages of protein sequences. The primary structure is the linear chain of amino acid, the secondary structure is this chain ordered in secondary structure motifs, the most prominent being alpha-helices and beta-sheets, the tertiary structure is a three-dimensional arrangement of those motifs as one would expect in vivo. The quaternary structure of a protein consists of several tertiary structures together as a complex.

Although Anfinsen's Paradigm [3] states that all information needed to create the tertiary structure lies within the sequence of the protein itself, finding this sequence is not trivial. The process of folding lowers the Gibbs free energy of the protein and the folded protein will have the lowest possible free energy. In other words, finding an energetic favorable structure for a given sequence will yield the structure of the folded protein. Based on this thought, many molecular dynamic (MD) simulations were constructed, to find the structure with the lowest energy by computing the force on the atoms of the protein and thus the change of their position over a given time frame. This yields good results in general but it consumes too much time and computational power. Typical MD-Simulations only simulate a few hundred nanoseconds, while a complete folding of a protein can take several microseconds.

Although a lot of sequences are known, the respective structures remain unsolved. As of July 2012, there were about 75,000 solved protein structures and about 6,000 solved nucleic acid and protein/nucleic acid complexes. The number of unique folds on the other hand is comparably slow and not as fast-growing as

the total number structures. A lot of programs exploit this fact. They are based on the idea that although the sequence does not fully determine the structure of the protein, it biases it into the right direction. It should therefore be possible to infer the structure of a query protein by looking at the structure of its known homologues. This is referred to as homology modeling. Prominent example of this method are for example Rosetta [37] or the here used Phyre [14].

A submitted query protein us searched against a database of non-redundant protein sequences using the tool PSI-Blast [1]. This is an development of the original BLAST algorithm, that uses so-called profiles, to iteratively find more homologous sequences. The secondary structure of the found sequences is computed using the tools PsiPred [11], SSPro [35] and JNet [7], also the ordered and disordered parts of the proteins are computed using Disopred [25]. The profiles found by PSI-BLAST and the secondary structure predictions are then scanned against a database of known folds to achieve an E-Value similar to that obtained in a traditional BLAST-search. The highest scoring known folds are then combined to yield a resulting fold for the query protein.

3 Workflow Realization

The jABC workflow now does several things. First, it tries to find homologous proteins via a Blast search. With those homologous proteins, a multiple sequence alignment is created using MaFFT. Simultaneously, the sequence is classified structural and functional using an InterProScan. The third part is finding information about the overall fold and structure. This is done using Phyre.

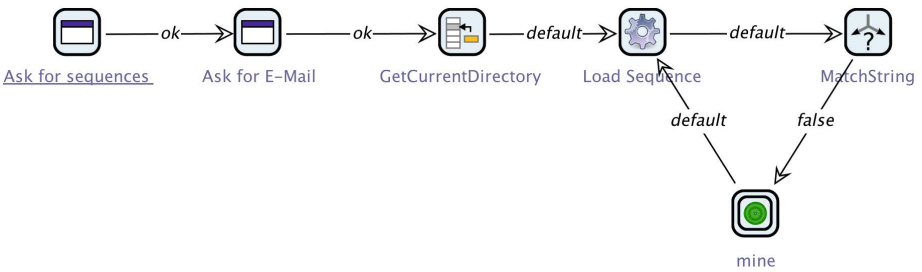

Fig. 1. Main procedure of this jABC workflow. It repeatedly loads a sequence from a file and sends it to the second procedure, here called *mine*, until the file is empty.

The workflow consists of two parts: The main part (see Figure 1) that is only used to set settings and load the sequences from the specified files, and the sub-workflow that realizes the actual gene annotation (see Figure 2).

The main procedure asks for the path of the sequence file. That is, a plain text file consisting of unknown sequences, one line each. It proceeds to ask for a valid e-mail address. This is necessary, since most of the online tools need it

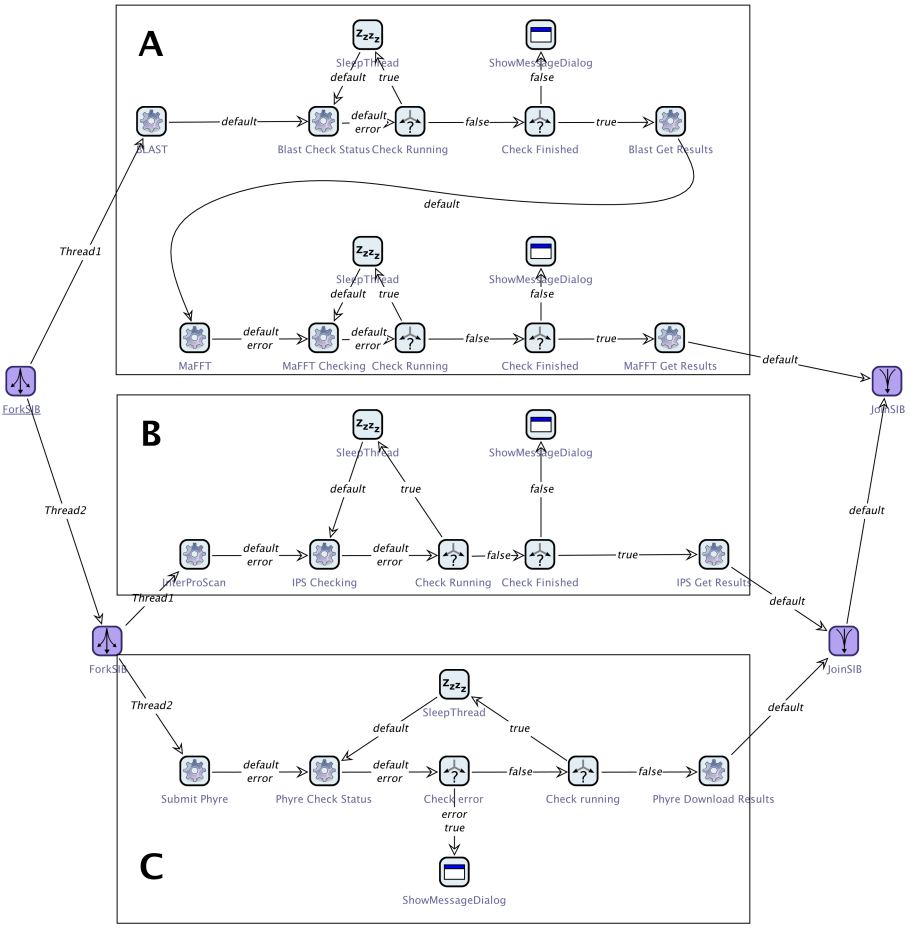

Fig. 2. Actual gene annotation workflow

to function properly. You must not leave this dialog box empty. It continues to determine the current directory. It is assumed that the directory is the jABC main directory, the shell scripts are assumed to be in jABC/Nat2/. A shell script is executed that splits off the first sequence of the sequence file and starts the mine-procedure.

The sub-workflow that realizes the actual gene annotation is then divided into three parts: part A creates the multiple alignment, part B searches the InterPro database using Interproscan, and part C searches for structural homologous using Phyre. These parts run in parallel in different threads created by a *Fork-SIBs*. At the end, they are all joined by *Join-SIBs*. This is necessary to prevent the main procedure from beginning with the next sequence, when the first has not completely finished. This drags down the speed of the procedure due to the

different speeds of the Alignment, IPS and Phyre tools. This is mainly due to the small capacity of the Phyre-Server; in peak-times the queue is comparably long so that a job may have to wait for more than an hour. This is a problem and should be addressed in further versions of this workflow.

3.1 Blast and Multiple Alignment

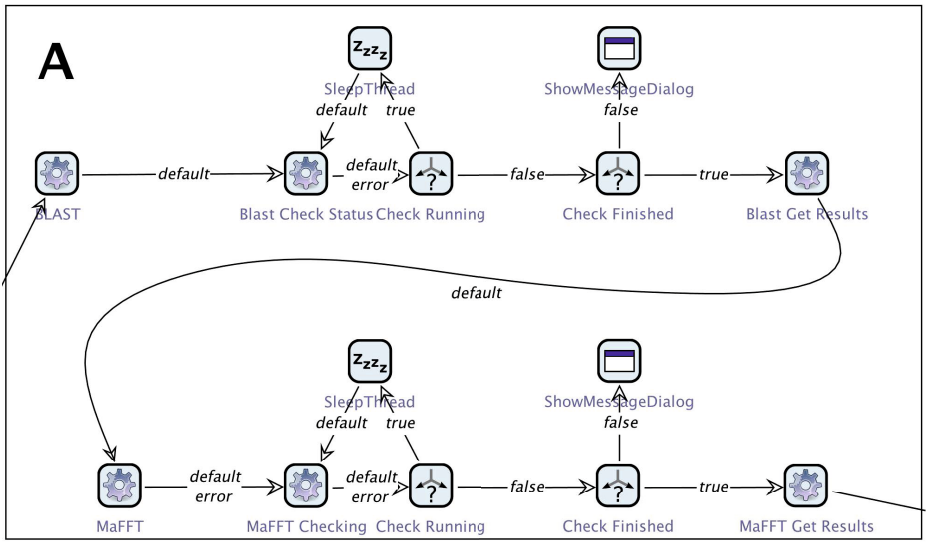

Fig. 3. Part A of the workflow: Blast and multiple alignment

In Figure 3 you see the workflow for the Blast and MSA procedures. The implementation is very straightforward. The first *ExecuteCommand*-SIB (termed BLAST) sends the sequence to the BLAST web service of the EBI. It returns the Job-ID, and identifier that can be used to check the status of the Job and download the results. The former task is done by the second *ExecuteCommand*-SIB (Check Running). It checks, if the job has already finished. If it hasn't, it waits for ten seconds, then checks again. After the job has finished, the third *ExecuteCommand*-SIB (Blast Get Results) downloads the results and stores them in the folder jABC/nat2/blast-results/[Job-ID]/.

The Blast results are then used as an input for the multiple sequence alignment. The first *ExecuteCommand*-SIB, MaFFT, loads the found homologous and sends them to the MaFFT service at the EBI. It returns a Job-ID, that the second *ExecuteCommand*-SIB (MaFFT Checking) uses to access the status of the job. If it is stilled running of queued, the workflow waits for thirty seconds, than queries the status again. After the Alignment is done, the third *ExecuteCommand*-SIB (MaFFT Get Results) downloads the results and stores them at jABC/nat2/mafft-results/[Job-ID]/.

3.2 InterProScan

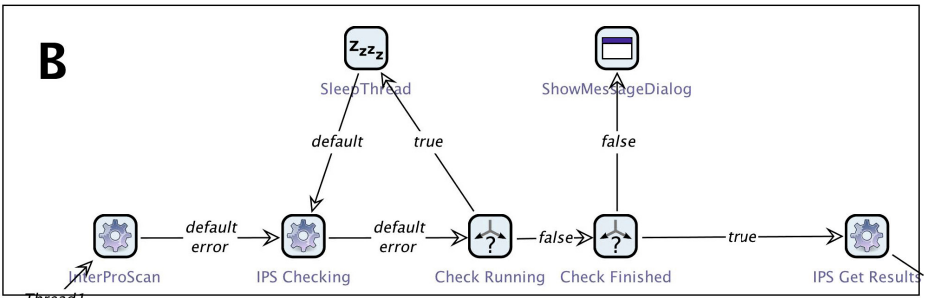

Fig. 4. Part B of the workflow: InterproScan of the query sequences

Figure 4 shows the second part of the workflow. As mentioned before, Interpro is being searched by IPS. The sequence is given to the first *ExecuteCommand*-SIB called InterProScan. It starts the online-search in the InerPro database. It uses the default values and returns a Job-ID. The second *ExecuteCommand*-SIB, IPS Checking, uses this ID to check the status of the job. Once it has finished, the third *ExecuteCommand*-SIB, IPS Get Results, downloads all available results.

3.3 Phyre

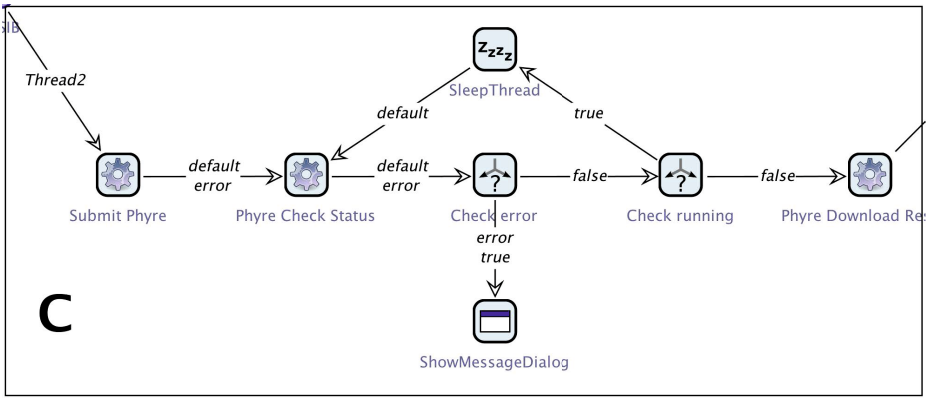

Fig. 5. Part C of the workflow: Structural modeling and homology search with Phyre

The last part of the workflow is depicted in Figure 5. Phyre is used to assess the overall fold of the protein. The implementation is as straightforward as in

the case of the IPS or MSA. Phyre is run on a server of the Imperial College London. It does not provide an explicit API and thus does not meet all criteria for a traditional web service. However, as Phyres web-page structure is very simple, it is easy to access it via cURL.

The first *ExecuteCommand*-SIB (Submit Phyre) submits the query sequence to the Phyre sever. The second *ExecuteCommand*-SIB checks the status of the Job. If the job is running, the process halts for thirty seconds and then queries the status again. When the job is finished, it is downloaded by the third *ExecuteCommand*-SIB, Phyre Download Results. The results are saved at jABC/nat2/phyre-results/[Job-ID]/.

4 Conclusion

With the advent of Next Generation Sequencing mechanism, Big Data in Biology is now a highly discussed topic. One of the problems is still the gap between the number of known sequences and the amount of information known about these sequences.

The goal of this jABC project was to - at least partially - automate the steps to annotate an unknown protein sequence and thus put them in a meaningful biological context. This goal was reached - the workflow presented here is an easy way to mine the databases for information about an unknown protein. The use of online services guarantees huge databases that can searched very performant, while keeping the requirements for the user in terms of hardware and cost low. Also, the workflow is easily expandable to incorporate new services or new options for the existing services as they arise.

Still, it should be noted that the jABC workflow was not as extensively tested, as one would do for a program for the productive use. The results should be treated with extreme caution. Also, the use of online-services may offer some disadvantages in terms of privacy, as the results of the queries are openly available.

There are various ways how this project could be extended. First, the operability could be improved by displaying the expected running time of the services, so the user always knows if and which program is running. Secondly, more services could be used to gain more information. An interesting idea would be to create a 3D-model in Rosetta, as this is a tool known to be very performant and accurate. Unfortunately, the Rosetta server that handles automatic model generation requires a log-in and it can not be expected of all users to have an account. Still, a local version of Rosetta could be used to locally compute the model of the protein. A third way to further improve the protein would be an automatic upload of the protein to UniProt along with all known information about the protein. A drawback of this workflow is the running time - especially the phyre-search runs very slowly and can take for longer proteins at peak-times up to some hours. A simple way to address this problem would be to rearrange the submit- and download-scripts. If at first all queries are submitted before the first download of the results is begun, the workflow will probably run a lot faster,

as the download scripts of InterProScan and MaFFT would not have to wait for Phyre to finish. This asynchronous work could improve the running time by a many-fold but could not be tested due to time concerns.

Another drawback of the current workflow is its strong platform dependency due to the use of the ExecuteCommand SIBs and shell scripts in a fashion that requires a particular directory structure. Here, platform independency could be achieved by using appropriate REST or jETI SIBs instead.

This article is part of a larger evaluation [18], which aimed at illustrating the power of simplicity-oriented development [30] by validating the claim that process modeling can indeed be handed over to the domain experts by providing them with a graphical modeling framework [43] that covers low-level details in a service-oriented fashion [32], integrates high-level modeling in the overall development process in a way that user-level models become directly executable [31,28], and supports ad-hoc adaptations and evolution [27,29].

The project described in this article can be characterized as follows:

- Scientific domain: bioinformatics
- Number of models: 2
- Number of hierarchy levels: 2
- Total number of SIBs: 38
- SIB libraries used (cf. [23]): common-sibs
- Service technologies used: REST web services

The bioinformatics part of this volume contains five other articles on workflow applications in this domain [36,24,38,39,45]. Further examples of workflow projects with he bioinformatics-specific incarnation of the jABC framework, called Bio-jETI [20], have been described, for example, in [19,21,8]. As shown in [20,22,17], bioinformatics is also a suitable field for the application of semantics-based (semi-) automatic workflow composition techniques (as provided by, e.g., [33]) to support the workflow design process.

References

1. Altschul, S.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acid research 25, 3389–3402 (1997)
2. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. J. Mol. Biol. 215(3), 403–410 (1990)
3. Anfinsen, C., Haber, E., Sela, M., White. Jr., F.: The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. PNAS 47(9), 1309–1314 (1961)
4. Benson, D., Karsch-Mizrachi, I., Lipman, D., Ostell, J., Sayers, E.: Genbank. Nucleic Acids Res. 7, D32–D37 (2011)

5. Berger, M., Muson, M.: A novel randomized iterative strategy for aligning multiple protein sequences. Comput. Appl. Biosci. 7, 479–484 (1994)
6. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., Bourne, P.: The Protein Data Bank. Nucleic Acids Research 28, 235–242 (2000)
7. Cole, C., Barber, J., Barton, G.: The Jpred 3 secondary structure prediction server. Nucleic Acids Res. 36(Web server issue), W197–W201 (2008)
8. Ebert, B.E., Lamprecht, A.-L., Steffen, B., Blank, L.M.: Flux-P: Automating Metabolic Flux Analysis. Metabolites 2(4), 872–890 (2012)
9. Gribskov, M., Luthy, R., Eisenberg, D.: Profile Analysis. Methods in Enzymology 183, 146–159 (1990)
10. Hunter, S., Jones, P., Mitchell, A., et al.: InterPro in 2011: new developments in the family and domain prediction database. Nucleic Acids Research 40, D306–D312 (2011)
11. Jones, D.: Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. 292, 195–202 (1999)
12. Katoh, K., Misawa, K., et al.: MAFFT version 5: Improvement in accuracy of multiple sequence alignment. KNucleic Acids Research 33(2), 411–518 (2005)
13. Katoh, K., Misawa, K., Kuma, K., Miyata, T.: MaFFT: a novel method for rapid multiple sequence alignment based on fast Furier transform. Nucleic Acids Res. 30(14), 3059–3066 (2002)
14. Kelley, L., Sternberg, M.: Protein structure prediction on the web: a case study using the Phyre server. Nature Protocols 4, 363–371 (2009)
15. Krogh, A., Brown, M., Mian, I., Sjolander, K., Haussle, D.: Hidden Markov models in computational biology. Applications to protein modeling. Journal of Molecular Biology 235(5), 1501–1531 (1994)
16. Labarga, A., Valentin, F., Anderson, M., Lopez, R.: Web services at the European bioinformatics institute. Nucleic Acids Research 35(Web Server issue), W6–W11 (2007)
17. Lamprecht, A.-L.: User-Level Workflow Design. LNCS, vol. 8311. Springer, Heidelberg (2013)
18. Lamprecht, A.-L., Margaria, T. (eds.): Process Design for Natural Scientists: An Agile Model-Driven Approach. CCIS, vol. 500. Springer, Heidelberg (2014)
19. Lamprecht, A.-L., Margaria, T., Steffen, B.: Seven variations of an alignment workflow - an illustration of agile process design and management in bio-jETI. In: Măndoiu, I., Wang, S.-L., Zelikovsky, A. (eds.) ISBRA 2008. LNCS (LNBI), vol. 4983, pp. 445–456. Springer, Heidelberg (2008)
20. Lamprecht, A.-L., Margaria, T., Steffen, B.: Bio-jETI: a framework for semantics-based service composition. BMC Bioinformatics 10(Suppl 10), S8 (2009)
21. Lamprecht, A.-L., Margaria, T., Steffen, B., Sczyrba, A., Hartmeier, S., Giegerich, R.: GeneFisher-P: variations of GeneFisher as processes in Bio-jETI. BMC Bioinformatics 9(Suppl 4), S13 (2008)
22. Lamprecht, A.-L., Naujokat, S., Margaria, T., Steffen, B.: Semantics-based composition of EMBOSS services. Journal of Biomedical Semantics 2(Suppl 1), S5 (2011)
23. Lamprecht, A.-L., Wickert, A.: The Course's SIB Libraries. In: Lamprecht, A.-L., Margaria, T. (eds.) Process Design for Natural Scientists. CCIS, vol. 500, pp. 30–44. Springer, Heidelberg (2014)
24. Lis, M.: Constructing a Phylogenetic Tree. In: Lamprecht, A.-L., Margaria, T. (eds.) Process Design for Natural Scientists. CCIS, vol. 500, pp. 101–109. Springer, Heidelberg (2014)

25. Marchler-Bauer, A., et al.: CDD: a conserved domain database for interactive domain family analysis. Nucleic Acids Res. 35(Database issue), D237–D240 (2007)
26. Margaria, T., Nagel, R., Steffen, B.: jETI: A tool for remote tool integration. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 557–562. Springer, Heidelberg (2005)
27. Margaria, T., Steffen, B.: Agile IT: Thinking in User-Centric Models. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. CCIS, vol. 17, pp. 490–502. Springer, Heidelberg (2009)
28. Margaria, T., Steffen, B.: Business Process Modelling in the jABC: The One-Thing-Approach. In: Cardoso, J., van der Aalst, W. (eds.) Handbook of Research on Business Process Modeling. IGI Global (2009)
29. Margaria, T., Steffen, B.: Continuous Model-Driven Engineering. IEEE Computer 42(10), 106–109 (2009)
30. Margaria, T., Steffen, B.: Simplicity as a Driver for Agile Innovation. Computer 43(6), 90–92 (2010)
31. Margaria, T., Steffen, B.: Service-Orientation: Conquering Complexity with XMDD. In: Hinchey, M., Coyle, L. (eds.) Conquering Complexity, pp. 217–236. Springer, London (2012)
32. Margaria, T., Steffen, B., Reitenspieß, M.: Service-oriented design: The roots. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 450–464. Springer, Heidelberg (2005)
33. Naujokat, S., Lamprecht, A.-L., Steffen, B.: Loose programming with PROPHETS. In: de Lara, J., Zisman, A. (eds.) Fundamental Approaches to Software Engineering. LNCS, vol. 7212, pp. 94–98. Springer, Heidelberg (2012)
34. Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biologie 48, 443–453 (1970)
35. Pollastri, G., Przybylski, D., Rost, B., Baldi, P.: Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. Proteins 47, 228–235 (2002)
36. Reso, J.: Protein Classification Workflow. In: Lamprecht, A.-L., Margaria, T. (eds.) Process Design for Natural Scientists. CCIS, vol. 500, pp. 65–72. Springer, Heidelberg (2014)
37. Rohl, C., Strauss, C., Misura, K.: DBaker. Protein structure prediction using rosetta. Methods in Enzymology 383, 66–93 (2004)
38. Schulze, G.: Workflow for Rapid Metagenome Analysis. In: Lamprecht, A.-L., Margaria, T. (eds.) Process Design for Natural Scientists. CCIS, vol. 500, pp. 88–100. Springer, Heidelberg (2014)
39. Schütt, C.: Identification of Differentially Expressed Genes. In: Lamprecht, A.-L., Margaria, T. (eds.) Process Design for Natural Scientists. CCIS, vol. 500, pp. 127–139. Springer, Heidelberg (2014)
40. Zvelebil, M., Baum, J.: Understanding Bioinformatics. Garland Science (2008)
41. Sigrist, C., Cerutti, L., Hulo, N., Gattiker, A., Falquet, L., Pagni, M., Bairoch, A., Bucher, P.: PROSITE: A documented database using patterns and profiles as motif descriptos. Briefigs in Bioinformatics 3, 265–275 (2002)
42. Sokal, R., Michener, C.: A statistical method for evaluation systematic relationships. The University of Kansas science bulletin 28, 1409–1438 (1958)
43. Steffen, B., Margaria, T., Nagel, R., Jörges, S., Kubczak, C.: Model-driven development with the jABC. In: Bin, E., Ziv, A., Ur, S. (eds.) HVC 2006. LNCS, vol. 4383, pp. 92–108. Springer, Heidelberg (2007)

44. Thomas, T., Gilbert, J., Meyer, F.: Metagenomics - a guide from sampling to data analysis. Microbial Informatics and Experimentation 2(3) (2012)
45. Vierheller, J.: Exploratory Data Analysis. In: Lamprecht, A.-L., Margaria, T. (eds.) Process Design for Natural Scientists. CCIS, vol. 500, pp. 110–126. Springer, Heidelberg (2014)
46. Zdobnov, E., Apweiler, R.: InterProScan - an integration platform for the signature-recognition methods in InterPro. Bioinformatics 17(9), 847–848 (2001)