UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Ingeniería del Software e Inteligencia Artificial

TESIS DOCTORAL

Arquitectura de un sistema multi-agente para el soporte de la anotación funcional
(MASSA: Multi-agent system to support functional annotation)

MEMORIA PARA OPTAR AL GRADO DE DOCTORA

PRESENTADA POR

Daniela Dias Xavier

Directores

Rubén Fuentes Fernández

Madrid, 2016

Arquitectura de un sistema multi-agente para el soporte de la anotación funcional (*MASSA: Multi-Agent System to Support functional Annotation*)

TESIS DOCTORAL

Daniela Dias Xavier

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Arquitectura de un sistema multi-agente para el soporte de la anotación funcional (*MASSA: Multi-Agent System to Support functional Annotation*)

Memoria que presenta para optar al título de Doctor en Informática

Daniela Dias Xavier

Dirigida por el Doctor

Rubén Fuentes Fernández

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

To all who never stopped
believing in me.

Deep into that darkness peering,
long I stood there, wondering,
fearing, doubting, dreaming dreams
no mortal ever dared to dream before.
Edgar Allan Poe

Acknowledgements

Success is a science; if you have the conditions,
you get the result.

Oscar Wilde

There is no such thing as a one-person show and this work could not be different. I would never have been able to accomplish this thesis without the knowledge, guidance, help, and support of my advisor, family, friends, and co-workers.

First of all, I would like to express my sincere thanks and appreciation to my advisor, Rubén Fuentes Fernández, for his guidance, motivation, and patience throughout this research. He not only gave me the knowledge and tools required to succeed, but also emotional support during this rough paved road.

I would also like to express all my gratitude to Berta Crespo, who became a dearest friend after all these years, and whose biological knowledge and collaboration were vital to this thesis.

My family played a very important role in this work as well. My deepest thanks to my amazing parents, Eduardo and Tânia Xavier, who always believed in me. Without their love, encouragement, and support, nothing of this would have been possible. A very special thanks to my dear aunt, TT, and "cousin", Lia, who wisely helped me to find strength to overcome each obstacle and to see the bright side of everything. Another big thanks to my awesome sister, Fernanda, who spent so many nights helping me to improve my English. Thank you sis for each lesson and book you gave me. You have no idea how important they were to me.

I would like to thank all my friends, specially Laura Cáceres, Verónica González, Clara Higuera, Arturo Marín, Alberto García Yoldi, José Manuel Rodriguez Carrasco, Miguelángel Cuenca, Laura Liles, Kelsea Boran, and Carl Broadbent, who worked directly with me, in different parts of the world, and gave me academic and emotional support throughout these years. I also need to thank my friend, Miguel Conde, who supported me immensely in the first years of this journey; my dear friend Maria Padilha, who was always there for me, illuminating my path and looking after me; and my loved friends Maria Pia Morelli, Loreto and Yoyo Calderón, Vanessa Pérez, Jerry Robinson, Sean Ralph, and Vivek Sud, who I met at the end of this road, but all of them were a great source of wisdom, support, and motivation. I am also very grateful to Kristina Ibañez, who in the

last years, when I was not living in Spain any more, helped me with all the paperwork to renew my registration as a student.

I cannot forget to thank my past advisors, Gonzalo Pajares and Maria Emília Machado Telles Walter, whose teaching and guidance were crucial not only to this thesis but to make me the professional I am.

Last but not least, I would like to extend my appreciation to everybody who had contributed directed or indirectly to this work.

Abstract

Predicting the biological function of Deoxyribonucleic Acid (DNA) sequences is one of the many challenges faced by Bioinformatics. This task is called *functional annotation*, and it is a complex, labor-intensive, and time-consuming process. This annotation has to be as accurate and reliable as possible given its impact in further researches and annotations.

In order to guarantee a high-quality outcome, each sequence should be manually studied and annotated by an expert. Although desirable, the manual annotation is only feasible for small datasets or reference genomes. As the volume of genomic data has been increasing, specially after the advent of Next Generation Sequencing techniques, automatic implementations of this process are a necessity. The automatic annotation can handle a huge amount of data and produce consistent analyses. Besides, it is faster and less expensive than the manual approach. However, its outcome is less precise than the one predicted manually and often has to be curated by an expert. Although collaborative processes of *community annotation* could address this expert bottleneck in automatic annotation, these efforts have failed until now. Moreover, the annotation problem, as many others in this domain, has to deal with heterogeneous information that is distributed and constantly evolving.

A possible way to overcome these hurdles is with a shift in the focus of the process from individual experts to communities, and with a design of tools that facilitates the management of knowledge and resources. This work follows this approach proposing MASSA, an architecture for a Multi-Agent System (MAS) to Support functional Annotation.

MASSA integrates two approaches from Artificial Intelligence: Rule-Based Expert Systems (RBESs) and MASs. The first one helps to mimic the expert reasoning and enhance the quality of the annotation, while avoiding the drawbacks of the manual approach. The RBES has been populated with knowledge from Biology and heuristics in Bioinformatics. The second approach deals with the different features of the available resources and takes advantage of distributed computing. This combination makes MASSA able to cope with the domain limitations and infer accurate annotations.

In order to keep pace with the constant changes of the domain and the limited human resources available, MASSA has been developed with a special focus on adaptability, evolution, and support to teamwork. For this purpose, it adopts state-of-the-art practices and technologies. For instance, the expert information from the domain was acquired

following knowledge elicitation techniques and modeled using the CommonKADS methodology, and the requirements of community annotation were analyzed using the Activity Theory framework.

MASSA performance was assessed using the benchmark proposed by the first large-scale community-based Critical Assessment of protein Function Annotation (CAFA). MASSA processed and annotated the 866 phylogenetically diverse sequences (11 species) from this experiment. The results were compared to the 10 top-performing algorithms described in CAFA, and MASSA outperformed them. Other features of MASSA were evaluated through expert surveys, also with positive results. This overall evaluation is highly encouraging regarding the benefits of the proposed architecture and considered expert knowledge.

Keywords: Functional annotation, Community annotation, Bioinformatics, Multi-Agent System, Ruled-Based Expert System.

Resumen

Predecir la función biológica de secuencias de Ácido Desoxirribonucleico (ADN) es unos de los mayores desafíos a los que se enfrenta la Bioinformática. Esta tarea se denomina *anotación funcional* y es un proceso complejo, laborioso y que requiere mucho tiempo. Dado su impacto en investigaciones y anotaciones futuras, la anotación debe ser lo más fiable y precisa posible.

Idealmente, las secuencias deberían ser estudiadas y anotadas manualmente por un experto, garantizando así resultados precisos y de calidad. Sin embargo, la anotación manual solo es factible para pequeños conjuntos de datos o genomas de referencia. Con la llegada de las nuevas tecnologías de secuenciación, el volumen de datos ha crecido significativamente, haciendo aún más crítica la necesidad de implementaciones automáticas del proceso. Por su parte, la anotación automática es capaz de manejar grandes cantidades de datos y producir un análisis consistente. Otra ventaja de esta aproximación es su rapidez y bajo coste en relación a la manual. Sin embargo, sus resultados son menos precisos que los manuales y, en general, deben ser revisados ("curados") por un experto. Aunque los procesos colaborativos de la *anotación en comunidad* pueden ser utilizados para reducir este cuello de botella, los esfuerzos en esta línea no han tenido hasta ahora el éxito esperado. Además, el problema de la anotación, como muchos otros en el dominio de la Bioinformática, abarca información heterogénea, distribuida y en constante evolución.

Una posible aproximación para superar estos problemas consiste en cambiar el foco del proceso de los expertos individuales a su comunidad, y diseñar las herramientas de manera que faciliten la gestión del conocimiento y los recursos. Este trabajo adopta esta línea y propone MASSA (*Multi-Agent System to Support functional Annotation*), una arquitectura de Sistema Multi-Agente (SMA) para Soportar la Anotación funcional.

MASSA integra dos aproximaciones de la Inteligencia Artificial: los Sistemas Expertos Basados en Reglas (SEBR) y los SMA. La primera ayuda a reproducir el razonamiento del experto, mejorando la calidad de la anotación y evitando los inconvenientes en coste de la anotación manual. El SEBR contiene conocimiento en Biología y heurísticas en Bioinformática. La segunda aproximación facilita el manejo de recursos heterogéneos y aprovecha la posibilidad de computación distribuída. Esta combinación permite a MASSA abordar las limitaciones del dominio e inferir anotaciones precisas.

Con el objetivo de poder mantener el sistema al día con los constantes cambios del dominio contando con un limitado número de expertos, MASSA ha sido desarrollado con

un énfasis especial en su adaptabilidad, evolución, y soporte al trabajo en equipo. Con este propósito, su desarrollo adopta metodologías, prácticas y tecnologías de vanguardia. Por ejemplo, la información de los expertos sobre el dominio se obtuvo mediante técnicas de elicitación del conocimiento y se modeló utilizando la metodología CommonKADS, y los requisitos para la *anotación en comunidad* se analizaron usando el marco de la Teoría de Actividad.

El rendimiento de MASSA fue evaluado con el procedimiento propuesto en la primera Evaluación Crítica de la Anotación Funcional de proteinas (CAFA, *Critical Assessment of protein Function Annotation*). Fue una comparativa gestionada por la comunidad de expertos y realizada a gran escala. MASSA procesó y anotó las 866 secuencias filogenéticamente diversas (11 especies) de este experimento. Sus anotaciones se compararon con las producidas por los 10 mejores algoritmos presentados en CAFA, siendo las de MASSA más precisas. Otras características de MASSA fueron evaluadas mediante encuestas a expertos, también con resultados positivos. Estos resultados globales apuntan a los beneficios de la arquitectura propuesta y el conocimiento experto considerado.

Palabras clave: Anotación funcional, Anotación en comunidad, Bioinformática, Sistema Multi-Agente, Sistema Experto Basado en Reglas.

About this document

Knowledge has to be improved, challenged, and increased constantly, or it vanishes.

Peter Drucker

This thesis is presented as a compilation of publications. The papers presented here are:

- Xavier, D., Morán, F., Fuentes-Fernández, R. and Pajares, G. Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology. *Expert Systems with Applications*, vol. 40(10), pages 3943-3952, 2013.

  Impact factor on 2013: 1.854, First Third (T1) in the category "Artificial Intelligence" (31/115); and First Quartile (Q1) in the category "Engineering, Electrical & Electronic" (56/243).

- Xavier, D., Crespo, B. and Fuentes-Fernández, R. A rule-based expert system for inferring functional annotation. *Applied Soft Computing*, vol. 35, pages 373-385, 2015.

  Impact factor in 2014: 2.810, First Quartile (Q1) in the category "Computer Science, Artificial Intelligence" (17/123); and First Quartile (Q1) in the category "Computer Science, Interdisciplinary Applications" (14/102).

- An analysis of tool requirements for collaborative annotation. Daniela Xavier, Rubén Fuentes-Fernández. Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Calabria, 6-8 May 2015.

  CORE B in 2014 (last version available).

- Xavier, D., Crespo, B., Fuentes-Fernández, R. and Gómez-Sanz, J. J. MASSA: Multi-Agent System to Support Functional Annotation. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection* (edited by Y. Demazeau, F. Zambonelli, J. M. Corchado and J. Bajo), vol. 8473 of *Lecture Notes in Computer Science*, pages 291-302. Springer-Verlag, Berlin-Heidelberg, Germany, 2014.

CORE C in 2014 (last version available).

Impact factor in 2009: 0.56. 52/701 (top 7,42 %) according to the Computer Science Conference Ranking, category Topics II: Artificial Intelligence / Machine Learning / Robotics / Human Computer Interaction.

This document comprises an introduction and an overview of the biological domain and some concepts required to understand the problem of functional annotation being tackled. It also contains the study of the tools and systems developed so far to support the experts in the process, together with the description of a methodology used to evaluate the system performance and in use in the domain. The main contribution of this thesis is the architecture for a system proposed to solve support experts in functional annotation. This document includes the description of this architecture and its functioning, and evaluation of its performance. In addition to that, the papers published during the research are presented as well. The document also includes the conclusions and future lines of work. As appendixes, there is some additional information regarding file formats and methodologies applied in the context of functional annotation and this research. Finally, the bibliography containing the references used in this thesis and in the papers is described.

Index

Figures Index

Tables Index

Chapter 1

Introduction

The beginning is the most important part of the work.

Plato

This chapter presents the motivation that inspired this thesis, as well as its goals and the work plan traced to accomplish it. The chapter ends describing the structure and content of the document.

1.1. Motivation

The first draft of the human genome was released in 2001, and after two years the first human genome was completed. It took thirteen years, three billion dollars, and a work force from different institutions to finish this project. More than a decade has passed, and the advances in sequencing technologies have drastically dropped the sequencing cost and time. Currently, a small genome can be sequenced in a few hours and a big one in some days at more affordable prices. In fact, in January 2014, Illumina announced its new system, HiSeq X Ten, which is capable of sequencing the human genome for less than $1,000. The advent of Next Generation Sequencing has not just affected the cost and time curves, it also has a remarkable impact on the volume of data produced, shifting the problem from generating genomic information to processing and analyzing it.

Bioinformatics is the discipline in charge of the management and analysis of biological data (Boguski, 1998). This field, where Biology, Computer Science, and Information Technology meet, aims to develop computational methods and tools for storing, retrieving, maintaining, processing, analyzing, and relating biological information. Bioinformatics can be applied in a range of biological and medical areas, such as Genomics, Proteomics, and Epigenetics.

Predicting the biological function of Deoxyribonucleic Acid (DNA) sequences is one of the many challenges faced by Bioinformatics. This complex, labor-intensive, and time-consuming process, called *functional annotation*, aims to characterize biologically the

genes (see Section 2.1), and predicting the biological role they and their protein products play (see Section 2.3). This process, also called *protein prediction*, comprises tasks such as: comparing sequences to huge data sets, accessing online or local resources, and dealing with different data formats and heterogeneous databases, employing in all these tasks proper knowledge and literature to infer and support the biological function. These tasks require expertise to handle computational tools and outputs and biological databases and information, and make pertinent decisions. Once the annotation is done, its outcome can be used in further researches and even in future annotations. Consequently, it has to be as reliable and accurate as possible in order to avoid misleading works or error propagation.

Ideally, the functional annotation should be carried out by an expert who detains the knowledge to assign the proper features to a sequence. The *manual annotation* yields high-quality annotations, but it is a slow process only feasible for small data sets or reference genomes. Besides, it can produce some conflicting outputs due to annotators' divergent interpretations (Potter et al., 2004). The *automatic annotation*, on the other hand, is able to process large data sets consistently without (almost any) user intervention. Although this approach is cheaper and faster than the manual one, its results are less accurate and should be verified ("curated") manually, which creates the expert bottleneck.

The functional annotation can be carried out through different approaches, but the majority of them ends up being very laborious and expensive or unfeasible computationally. The most practicable method is "inheriting through homology" (see Section 2.5), which takes advantage of evolutionary relationships among sequences of genes or proteins. In order to do it, it is essential to compare the sequences of interest with a set of known sequences. Then, if a good match is found, the annotation is transferred from the already annotated sequences to the query.

There is a variety of publicly available tools that support the different steps of functional annotation through homology. Among them, BLAST (Altschul et al., 1997) (see Section 3.1.1) has a leading role. This comparison tool is the first step of many annotation pipelines (e.g., (Bryson et al., 2001; Cadag et al., 2007; Cozzetto et al., 2013; Decker et al., 2002; Domselaar et al., 2005; Falda et al., 2012; Götz et al., 2011; Gouret et al., 2005; Koski et al., 2005; Koskinen et al., 2015; Potter et al., 2004)), and it is also wrapped as part of some stand-alone software applications that support annotation (e.g., (Storm and Sonnhammer, 2002)). The different flavors of BLAST (see Section 3.1.1) have distinct applications that go from verifying DNA similarity to revealing distant evolutionary relationships. Despite BLAST popularity, some tools like InterProScan (Jones et al., 2014) (3.1.3.1), PfamScan (Finn et al., 2014) (see Section 3.1.3.2), and HMMER (Eddy, 1998) (see Section 3.1.2) have their own comparison algorithms. Regardless their algorithms, the majority of the supporting tools compares the input sequence(s) to one or more local or remote data sources.

Supporting tools can provide important insights in the annotation process, but their use presents some constraints. First, these tools focus on just part of the annotation process. Thereby, in order to obtain a precise prediction, the expert needs to combine

them. Second, they are generally stand-alone programs that are not integrated with each other, so the expert has to perform each stage manually and then combine the outputs properly (Bryson et al., 2001). Third, although these tools generally have the same type of input, a FASTA file (see Appendix A.2.1), they often produce different output formats. Fourth, some of these tools and their outputs are not quite intuitive for the final users, so they require long practice to master them. Fifth, many of these tools employ biological information that is stored in large data sets of heterogeneous databases that are in different locations and constantly being updated. This requires continuously checking sources and hinders performance. Sixth, since their output may contain spurious data, it should be curated by an expert who detains the knowledge to extract the relevant information.

Aiming to infer more reliable annotations and to address the aforementioned constraints, annotation systems have been developed. These systems combine supporting tools (that tackle different biological knowledge) with some expertise in the domain and heuristics. Although these systems share a common goal, they differ in regards to the knowledge they comprise, their output type and quality, and their design approaches.

These tools have been validated through their use to annotate different types of organisms (e.g., apple and grape (Falda et al., 2012); human, mouse, rat, and zebra fish (Potter et al., 2004); and herpes viruses (Decker et al., 2002)). However, their current application faces some important problems in different aspects, mainly regarding their use of knowledge and design. Given that these applications are intended to support knowledge-intensive tasks, what information they consider and how they manage it are key aspects. Their design must also be consistent with their use by experts and the resources they need to employ.

Regarding the applied knowledge, systems tend to encapsulate limited expertise. Most of them (e.g., (Cozzetto et al., 2013; Decker et al., 2002; Gouret et al., 2005; Potter et al., 2004)) integrates tools to study homology (see Section 2.3) and for protein patterns identification (e.g., domain and motifs) (see Section 2.3), using a controlled vocabulary (see Section 2.5.1.5). Nevertheless, just a few incorporate further relevant knowledge like transmembrane topology prediction (e.g., (Decker et al., 2002; Domselaar et al., 2005; Möller et al., 1999; Potter et al., 2004)) or ortholog relationships (e.g., (Cozzetto et al., 2013; Domselaar et al., 2005; Gouret et al., 2005)), features that increase the accuracy of the annotation. Furthermore, there are systems that do not even apply all the "basic knowledge" (i.e., homology, domain prediction, and ontology) (e.g., (Koskinen et al., 2015)), and others that were implemented to deal only with specific data sources (e.g., (Chen et al., 2012; Falda et al., 2012; Gouret et al., 2005)).

It is also important to have a good understanding of the domain where the problem lies. The majority of systems seems to be problem-oriented, and just a few of them (e.g., (Bryson et al., 2001; Decker et al., 2002)) consider some particularities of the domain (e.g., data that are heterogeneous, distributed, and constantly evolving). Moreover, in general, they focus on the individual annotator in a given setting, not taking into account collaborative incentives in the biological community, such in *community annotation*.

The previous considerations affect the design of tools. However, they are frequently

not addressed properly. In particular, there are important issues regarding how experts can use tools in integrated ways to overcome difficulties that were not considered in their initial requirements.

Some tools present restrictions regarding the kind of query sequences they support. For example, the Ensembl Analysis Pipeline (EAP) (Potter et al., 2004) (see Section 3.2.1.2) only handles sequences inside the genomic context, while the Bacterial Annotation System (BASys) (Domselaar et al., 2005) (see Section 3.2.4.1) was designed specifically for bacterial input.

There are also issues regarding the outcome of tools. The quality of the annotation is crucial, and assessing it requires plenty of information about the process. Nevertheless, some systems (e.g., (Chen et al., 2012; Conesa et al., 2005; Domselaar et al., 2005; Koski et al., 2005)) work merely as pipelines that combine tools, rather than focusing on the inference process. So, they omit further relevant information on the annotation. Another outcome issue is the format of the output. There is a lack of standards, as no annotation format has been set so far. Systems present their outcomes in multiple ways with different shortcomings: some offer very visual outputs (e.g., (Conesa et al., 2005; Domselaar et al., 2005)), which can be beneficial for the user but prevents computational processing; others discard potentially relevant information (e.g., (Cozzetto et al., 2013; Falda et al., 2012) just make a list of related ontology terms, but do not include evidences to support them).

A final issue, affecting both support tools and annotation systems, is the lack of documented applications of good practices in their development. First, literature does not report the use of any Software and Knowledge Engineering methodologies, so key decisions are neither explained nor documented. This makes difficult improving the design of tools based on previous experiences, and even the exchange of more abstract knowledge regarding the annotation process. For instance, there is no documentation on the development approach of EAP (Potter et al., 2004). Second, there is also a lack of maintenance guidelines regarding knowledge and integrated tools. For instance, EAP (Potter et al., 2004) and Figenix Gouret et al. (2005) integrate other tools, but only consider dataflow management. There are no guidelines on how to deal with the data of these systems as biological information or to integrate new tools into them. Moreover, some systems rely on poorly supported *ad-hoc* technologies (e.g., (Potter et al., 2004)) or less popular languages (e.g., (Möller et al., 1999)). This way of working makes systems difficult to maintain and evolve.

This thesis looks for a general solution to provide annotation systems that address the previous limitations. This solution needs to consider several issues:

- How to elicit the relevant knowledge for annotation, and describing it in a form suitable for automated processing and human reading.
- What the data sources and support tools that experts consider useful are, for what purposes, and how they use them.
- The kind of support needed by annotators, both at the individual and group levels.
- Designing a suitable system architecture to meet the requirements identified pre-

viously. This architecture, and its proof-of-concept implementation, are called MAS-SA (Multi-Agent System to Support functional Annotation). It should consider the knowledge to manage, the integration of distributed and heterogeneous data sources and support tools, and the usual tasks of experts. It should also be well-suited for evolutive maintenance regarding knowledge, data sources, and support tools.

- Developing, supporting and documenting the previous elements, when possible using well-established practices of Requirements and Software Engineering. This would facilitate understanding the design and the experience got with the solution.

1.2. Objectives

Based on the context discussed previously in Section 1.1 the goals of this work are:

- *To decrease the workload of experts without detriment to the quality of the annotation.* Automating the annotation problem makes possible to process the ever growing amount of genomic data. This thesis proposes to combine the expertise in the domain with computational knowledge to develop a functional annotation system that produces reliable and precise annotations with limited expert participation.

- *To propose a system architecture able do deal with the inherent particularities of the domain.* The knowledge experts apply is heterogeneous, coming from different and not always consistent sources, and continually evolving. The information in the domain shares the features of that knowledge, and it is also characterized by high volumes of data and distributed data sources. Finally, support tools offer quite different features, even for the same tasks. Considering that experts have optimized their processes for this context, the architecture should be based on key steps of the manual annotation process, being close to the expert experience and intuitive for the user. Thus, the system is going to simulate the expert reasoning process at annotation inference, and make use of available resources and tools when possible.

- *To develop the system with a special focus on evolution.* Systems that are well-designed, easy to maintain, and capable to evolve are more prone to succeed. In a constantly changing environment, as the one studied, these features should be considered and supported by well-established methods.

- *To assess the fulfillment of the architecture requirements.* Validating the system against its requirements is crucial in any development. In this case, there are several design requirements (e.g., usability, ease of maintenance, or support to community annotation) that are checked through a qualitative analysis of the system development and use. Besides these requirements, there is also need to evaluate the actual performance of the system in annotation. This work proposes to assess this following the benchmark used by the first large-scale community-based Critical Assessment of protein Function Annotation (CAFA) (see Section 3.2.5).

1.3. Research methodology

In order to address the previous goals, the following work plan has been followed:

1. *Studying the domain* focusing on:

   a) *Obtaining the knowledge required to understand the studied problem.* This knowledge was acquired through two different research approaches:

   - *Primary research* was carried out in order to get a deep understanding of the problem and its domain. Here, attention was paid to the particularities and limitations in knowledge processing and tool operation.
   - *Knowledge Elicitation through structured interviews* was also conducted to acquire expert knowledge not described in the literature, and to comprehend the reasoning process and different approaches used by experts.

   b) *A study of the state-of-the-art in tools* was made with special focus on:

   - *The supporting tools publicly available.* This study allowed understanding the role of the tools used in the different steps of the annotation process, their usage, input, output, parameters, and the data sources they consult. Based on it, supporting tools were reviewed, installed, and tested.
   - *The annotation systems that have been developed.* This investigation provided insights on the strengths and weaknesses of these systems with regards to: the knowledge and tools they comprise, their goals, output, computational paradigms, architecture, and development model, and the way they are made available.

   c) *An analysis of community annotation.* In order to propose a system able to meet the main needs of the biological community, a specific study centered in *community annotation* efforts was carried out. This aims to have a better understanding of the domain under an ill-explored but growingly relevant perspective.

   The results of all these studies were combined to define the requirements of the proposed solution. At this point, the focus was on the key steps of the manual annotation, the knowledge applied at different stages of the process, and the data sources (and their formats) and support tools that provide relevant information. These aspects set the core requirements of the solution.

2. *Modeling the knowledge* using Knowledge Engineering methodologies. This step was crucial to understand the type of problem being tackled, to propose the solution approach that fits better the problem, and to properly structure the knowledge. This step also provided guidelines for the future adaptation of the knowledge considered in the solution to new findings in the domain.

3. *Designing the system architecture* following Software Engineering methodologies. Identifying the requirements of the system and the knowledge it has to process

allows determining the type of computational solutions best-suited for it. Then, the development was accomplished following suitable methodologies for the chosen paradigms.

4. *Implementing the system* using well-supported languages and frameworks. This task adopted technologies that facilitate the implementation of the proposed architecture, but also the maintenance of the system by providing development resources (e.g., tools, libraries, and documentation).

5. *Evaluating the system regarding its requirements.* In the case of annotation performance, evaluation followed the benchmark proposed in CAFA (Radivojac et al., 2013). The results were compared to the ones obtained by the systems presented in that experiment. Other requirements were evaluated qualitatively with expert and user surveys.

1.4. Document structure

The results of the domain research, the system development and assessment, and all the publications related to this work are gathered in the rest of the document. They are structured in five chapters and two appendices as follows.

Chapter 2 introduces the biological concepts required to understand the annotation problem. It also describes this problem, explaining some of the approaches used to tackle with it. This chapter also presents a brief review of some publicly available databases which knowledge is applied to aid in the annotation.

Chapter 3 compiles some publicly available tools that support the different steps of the annotation process. These tools are grouped based on the aspect of the problem they address. The same chapter presents some of the annotation systems that have been developed so far. Although the main focus is on systems that apply Artificial Intelligence (AI) approaches, non-AI systems are also described. This chapter also introduces and explains the CAFA experiment, presenting its three top-performing systems. At the end of the chapter, a comparative analysis of the state-of-the-art tools is carried out, which provides the basis to define some system requirements.

The proposed solution is presented in Chapter 4. The chapter starts giving an overview of the main aspects considered for the system: how its knowledge was acquired, the tools it needs to work with, and the support required for community annotation. Then, it describes the system architecture that implements these requirements. Finally, there is an evaluation of the system according to the metrics introduced in Chapter 3.

Chapter 5 gathers the papers published during this research. The first article describes how the knowledge was elicited and modeled. The second one introduces the system architecture, while the third focus on the sub-system directly related to the inference process. The last one, presents the analysis carried out of collaborative annotation.

The conclusions of this work are presented in Chapter 6. In it, some system improvements and future lines of research are proposed.

This work also contains two appendices. Appendix A describes and exemplifies some of the most popular Bioinformatics file formats. It also presents the format used by MASSA for its configuration files. Appendix B gives an overview of the methodologies applied in the different aspects of the development of MASSA: to model the system knowledge, to carry out the social analysis of the community annotation requirements, and to design the system.

Chapter 2

Biological Background

DNA is like a computer program but far, far more advanced than any software ever created.

Bill Gates

This chapter introduces biological and bioinformatics notions and knowledge required to understand the problem addressed in this work, the *functional annotation*. It also gives some insights of how this knowledge is applied in the development of the architecture proposed to address the annotation problem, MASSA. Sections 2.1, 2.2, and 2.3 explain basic biological concepts related to DNA and genes, Ribonucleic Acid (RNA), and proteins respectively. The functional annotation aims to infer the biological function (and other features) of genes through the proteins they encode. The methods used to predict the protein function are briefly described in Section 2.4. Then, Section 2.5 discusses the functional annotation problem, introducing data sources and methods to tackle it. Finally, Section 2.6 sums up the key knowledge presented.

2.1. From the cell to the DNA

The cell is the basic structural and functional unit of all living kinds. Cells of all organisms share some structural features. The *plasma membrane* (see Figure 2.1) is a thin and tough barrier around the cell that separates its content from the surroundings and controls the exchange of substances between the cell and the external environment. This compartment encloses the *cytoplasm*, an aqueous solution that gives support to the cell and facilitates the intra-cellular transport. The *cytoplasm* contains a range of suspended particles and *organelles* that have different functionalities.

One of the most important *organelles* is the one that houses the complete genetic endowment of an organism (i.e., the *genome*). In *prokaryotes* (i.e., bacterias and archeas), it is called *nucleoid* and is immersed inside the *cytoplasm* without any membrane to separate them. Conversely, in *eukaryotes*, the *nucleus* wraps the nuclear material with a

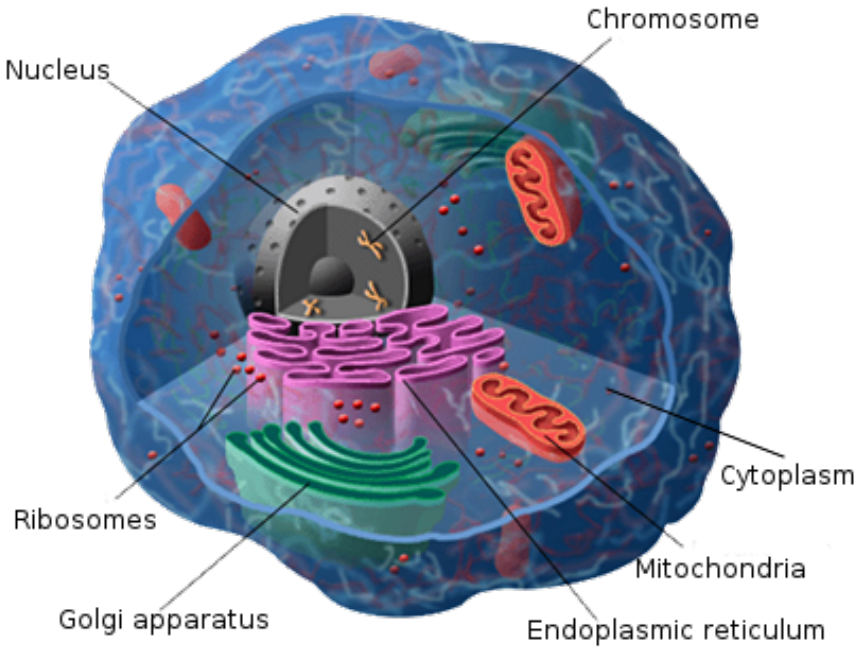


Figure 2.1: Anatomy of the eukaryotic cell.

Source: CRG - Council for Responsible Genetics (2015)

double membrane. In this work, the biological concepts are going to be explained in the light of eukaryote cells.

Cells from eukaryotes are larger than those of prokaryotes, and besides the *nucleus* they contain a variety of *organelles* not found in the last ones, like the *mitochondria*, the *endoplasmatic reticulum*, and the *Golgi apparatus*. The first is in charge of producing energy to the cell, the second works as a transport network and storage for some cellular substances, and the third modifies, sorts, and packs synthesized macromolecules for cell secretion or internal use. These macromolecules are synthesized in an *organelle*, common to both eukaryotes and prokaryotes, called *ribosome*.

Inside the *nucleus* the hereditary information is stored in DNA macromolecules. The DNA is composed of nitrogen-rich bases (i.e., *nucleotides*), deoxyribose sugars, and phosphates (see Figure 2.2.a). There are four types of nucleotides: Adenine (A), Thymine (T), Guanine (G), and Cytosine (C). Each of them bonds only with a specific base: A with T, and C with G. In the DNA, these bases are organized in two side-by-side complementary polynucleotide strands held together by hydrogen bonds. These two long strands entwine in a double helix.

The DNA is organized and compacted within one or more *chromosomes*. *Chromosomes* are essential to pass genetic traits between generations. In a very simplistic way,

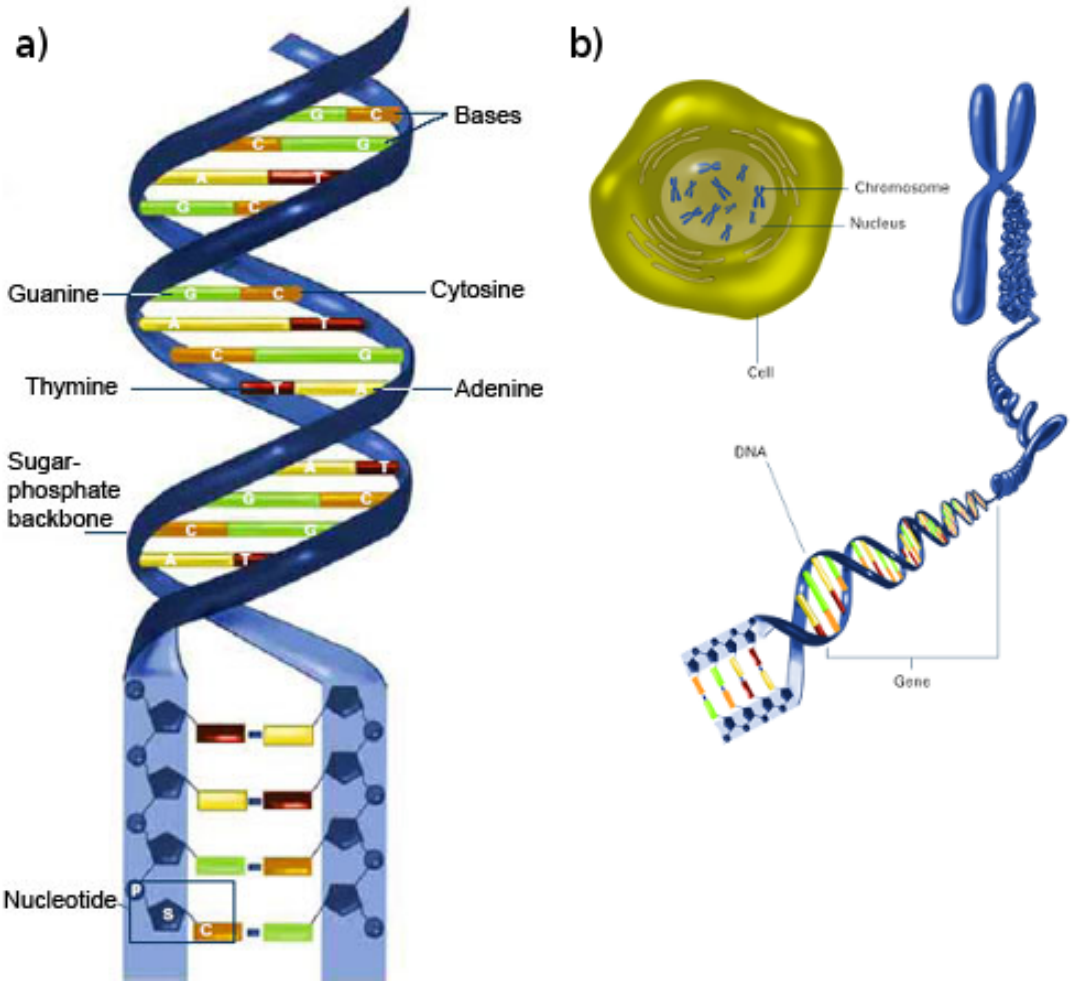


Figure 2.2: a) DNA composition and structure. b) From the cell to the gene.

Source: CRG - Council for Responsible Genetics (2015).

they can be visualized as a stick-like form (see Figure 2.2.b), although they only take this shape during cell division. This compact form consists of a pair of sister *chromatids* (composed of DNA and histones, which are structural proteins) united by a *centromere*. Each organism has a different number of *chromosomes*. For instance, humans (*Homo sapiens*) have 46 *chromosomes*, while mouses (*Mus musculus*) have 40, and dogs (*Canis familiaris*) 78 (Pevsner, 2009).

In a chromosome, along the DNA strand, there are sequences of nitrogen-rich bases that detain the information to control inherited traits. These sequences are the *genes*. All the cells of an organism have the same genetic information, that means, they have the same *genes* independently of the cell's function. However, in each cell just some genes are expressed and this expression can be temporary. A *gene* can be composed of non-coding

and coding regions, known as *introns* and *exons* respectively (see Figure 2.4). In higher eukaryotes, the genes are mostly formed by *introns*. For instance, 85 % of the genes that code the "single pclypeptide chain of ovalbumin" consists of *introns* (Lehninger et al., 2008). Despite being the majority in a gene, the *intron's* functions remained uncertain for years. Currently, it is known that *introns* are involved in a range of functions from the regulation of *transcription* (see Section 2.2) to *genome* organization (Chorev and Carmel, 2012).

A DNA sequence can be represented as a chain of nucleotide letters and its size denoted by the number of bases in the chain, known as *base pair* (bp). The size of the *genome* depends on the organism. Although the number of genes does not influence the genome size, the number of chromosomes might. Prokaryotes have genomes in the order of a few million bp whereas the genome size of a eukaryote can vary from 10^7 to 10^{11} bp (Cooper, 2000). The human genome has 3.2 Gbp and from 25 to 30 thousand protein-coding genes (Pevsner, 2009).

In genetics, genomics is the field that focus particular attention on the genome's function and structure. Nowadays, genomics, together with other areas of genetics, is directly linked to Bioinformatics. Bioinformatics is a fusion of Biotechnology and Information Technology that aims at the management and analysis of biological data through computational approaches (Boguski, 1998; Pevsner, 2009).

2.2. The power of the RNA

RNA is a macromolecule of ribonucleic acid, similar to the DNA. Like DNA, it is composed of complementary nitrogen-rich bases, but it is single-stranded and the Uracile (U) replaces the T. There is a variety of RNAs, though three of them play major roles during the protein synthesis: messenger RNA (mRNA), transfer RNA (tRNA), and ribosomal RNA (rRNA).

The protein synthesis starts in the nucleus with the *transcription* (see Figure 2.3.a), where the DNA double helix is unwound and one of its strands used as a template to generate a complementary mRNA molecule. This is then modified and its message edited. Although the entire gene (i.e., introns and exons) is transcribed, all introns are removed from the mRNA during the editing step. At this point, exons can also be deleted from the mRNA, generating sequences with distinct combinations of exons. This phenomenon is called *alternative splicing* (see Figure 2.4) and brings a great versatility to the gene, allowing it to be expressed in different ways. After these modifications, the mRNA abandons the nucleus, as the protein is synthesized outside it.

The mature mRNA migrates to the *cytoplasm*, more precisely to the *ribosome* (made of rRNA). This *organelle* is responsible for the *translation* (see Figure 2.3.b), where the mRNA is recognized and its message read by consecutive nucleotide triplets, known as *codons*. The *genetic code* comprehends 64 *codons*, formed by the combination of the four bases (i.e., A, C, G, and U), being 61 in charge of coding specific *Amino Acids* (AAs) and 3 capable of ceasing the *translation* process (Table 2.1).

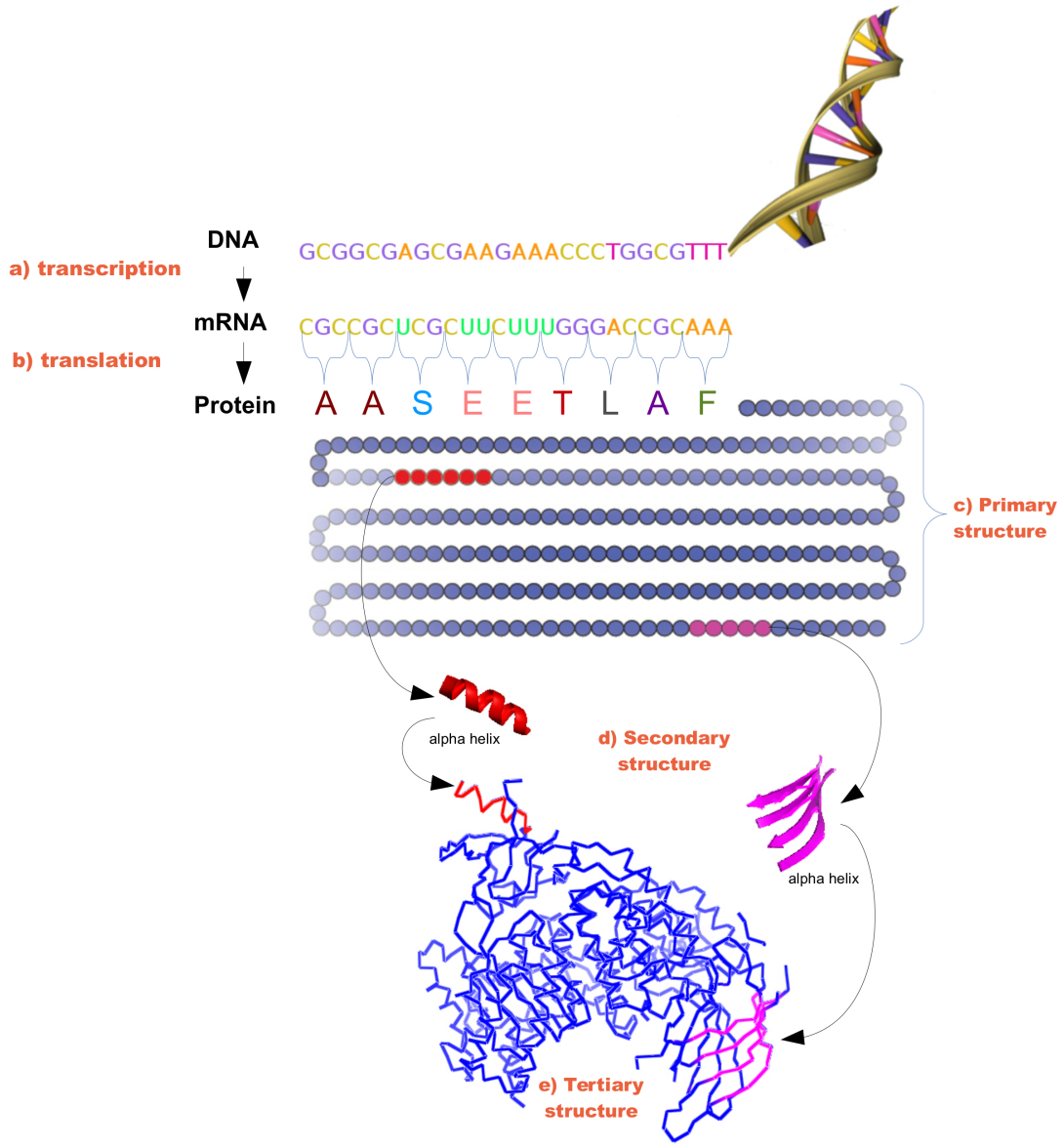


Figure 2.3: Part of the central dogma of Biology (*transcription* and *translation*) along with the different levels of protein organization.

There are 20 AAs present in nature. They are composed of an amino group, a carboxyl group, and a radical group (see Figure 2.5). When linked together in chains they form *polypeptides*. The tRNA is the one which transports these compounds to the ribosome for their assembly into polypeptides. When the ribosome reads the STOP codon, the translation is finished and its product, the *protein*, is released.

As depicted in Figure 2.6, it is possible to read the mRNA beginning from different

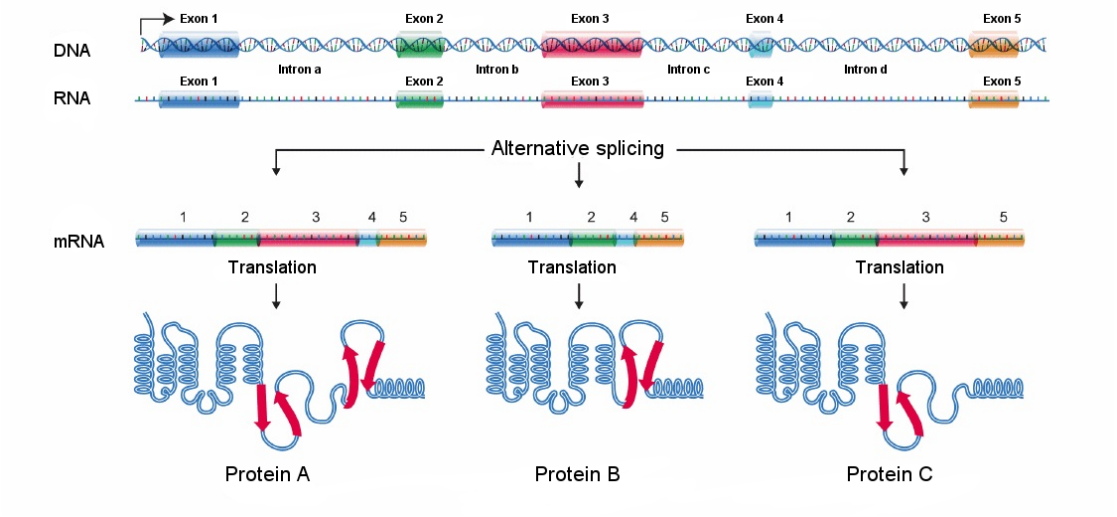


Figure 2.4: Organization of a *gene* into *introns* and *exons* and an example of *alternative splicing*.

Source: Wikipedia (2015).

nucleotides in a triplet, that is, using distinct *reading frames*. This event increases the complexity of the DNA, making it possible to read each strand in three different frames: 1 (first base of the codon), 2 (second base of the codon), and 3 (third base of the codon). Since the DNA is a double helix containing two paired strands in opposite directions —forward (+) and reverse (-)—, if we randomly (without knowing its strand and frame) pick a DNA sequence, it is possible to read it in six different ways: +1, +2, +3, -1, -2, and -3.

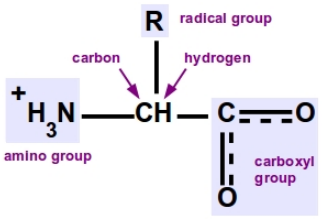


Figure 2.5: Amino acids chemical composition.

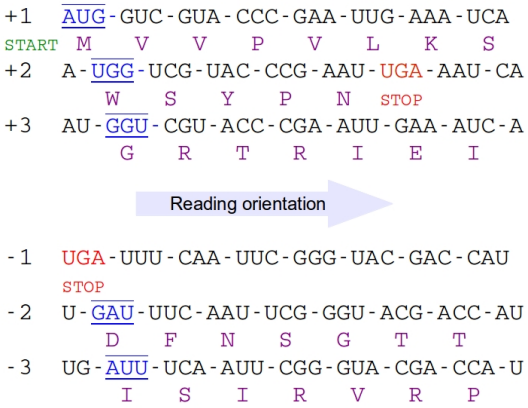


Figure 2.6: Reading frames for both strands in mRNA.

First Letter ↓	Second Letter U	Second Letter C	Second Letter A	Second Letter G	Third Letter ↓
U	phenylalanine (Phe)	serine (Ser)	tyrosine (Tyr)	cysteine (Cys)	U
	phenylalanine (Phe)	serine (Ser)	tyrosine (Tyr)	cysteine (Cys)	C
	leucine (Leu)	serine (Ser)	STOP	STOP	A
	leucine (Leu)	serine (Ser)	STOP	tryptophan (Trp)	G
C	leucine (Leu)	proline (Pro)	histidine (His)	arginine (Arg)	U
	leucine (Leu)	proline (Pro)	histidine (His)	arginine (Arg)	C
	leucine (Leu)	proline (Pro)	glutamine (Gln)	arginine (Arg)	A
	leucine (Leu)	threonine (Thr)	lysine (Lys)	arginine (Arg)	G
A	isoleucine (Ile)	threonine (Thr)	asparagine (Asn)	serine (Ser)	U
	isoleucine (Ile)	threonine (Thr)	asparagine (Asn)	serine (Ser)	C
	isoleucine (Ile)	threonine (Thr)	lysine (Lys)	arginine (Arg)	A
	methionine (Met) & START	threonine (Thr)	lysine (Lys)	arginine (Arg)	G
G	valine (Val)	alanine (Ala)	aspartate (Asp)	glycine (Gly)	U
	valine (Val)	alanine (Ala)	aspartate (Asp)	glycine (Gly)	C
	valine (Val)	alanine (Ala)	glutamate (Glu)	glycine (Gly)	A
	valine (Val)	alanine (Ala)	glutamate (Glu)	glycine (Gly)	G

Table 2.1: Genetic code.

2.3. The protein

Proteins are long chains of AA residues (i.e., each AA component group) generated through the *translation* process. They are probably the most versatile of all biomolecules (Lehninger et al., 2008) due to their broad array of functions, including serving as structural elements and signal receptors. They play a major role in life maintenance, being involved in all metabolic processes.

Proteins have a biomolecular structure which size and form vary according to its purpose. The protein structure can be decomposed into different levels of complexity. The *primary structure* consists of a linear chain of AA and, similarly to DNA, it can be represented as a string of AA letters. The *primary structure* folds into itself, forming two-dimension structures (i.e., the *secondary structure*), such as turns, helixes, and sheets (see Figure 2.3.d). The connectivity of two or more of these elements forms a folding pattern called *supersecondary structure* or *motif*. Elements of the secondary structure together with motifs are packed in the space into compact units, generating the *tertiary structure* (see Figure 2.3.e).

The protein's biological function is directly related to the *tertiary structure*, and some regions of the *primary structure* may vary substantially without affecting its role. Nevertheless, the AA regions and *residues* that are crucial to the protein's function and structure are in general conserved over evolutionary time, like *conserved sites* and *protein domains*. *Motifs*, on the other hand, are not related to the biological function. In fact, the same pattern can be found in proteins with distinct functions.

Conserved sites (CSs) are small regions of the protein that are preserved through time. *Active sites* and *binding sites* are examples of the most highly CSs. The first ones are the places in the *enzymes* (i.e., catalytic proteins) where the chemical reaction happens, while the second ones are the region where chemical bonds to specific molecules occur.

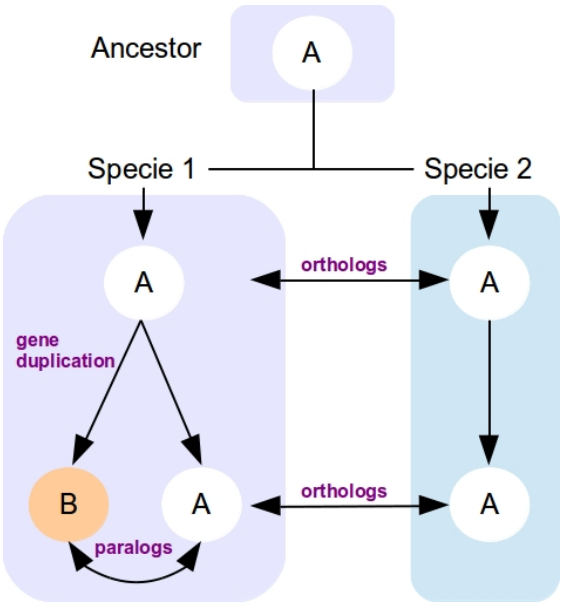


Figure 2.7: Homology relationships.

A protein *family* is a set of proteins which members descend from a common ancestor. Proteins from the same family are often produced by evolutionary-related genes known as *homologs*. The term *homolog* can also be applied to proteins. *Homolog* proteins have a significant similarity in *primary structure*, being generally identical across a 25 % or more of their sequence, and/or with similar *tertiary structure* and function (Lehninger et al., 2008). Two *homologs* are said to be *paralogs* if they are found in the same species, or *ortologs* if they belong to different species (Figure 2.7). Orthologous genes are more prone to share the same function than paralogous genes (Gouret et al., 2005). *Protein families* that share some levels of similarity can be grouped into *superfamilies*.

2.4. Predicting the protein function

The *tertiary* structure and function of a protein are tightly connected. Therefore, an intuitive way of predicting the function is finding its three-dimensional shape. It is possible to determine the protein's structure through experimental methods, like X-Ray Crystallography and Nuclear Magnetic Resonance. Although these techniques are fairly precise, they are labor-intensive, time consuming, and expensive, which prevents their extensive use. Alternatively, the three-dimensional structure can be obtained from the *primary structure*, but this is a NP-complete (Nondeterministic Polynomial time

complete) problem (Berger and Leighton, 1998; Crescenzi et al., 1998b).

A more popular and feasible approach is inheriting the function through homology (Lee et al., 2007). This method takes advantage of evolutionary relationships and uses the homology concept: if several sequences are similar enough to be homologs, they probably have similar functions (Pevsner, 2009). Based on that, a protein sequence of interest (i.e., the *query*) is compared to a set of previously studied sequences (i.e., the *targets*), and if a good match is found, the annotation is transferred to the query sequence; otherwise, the protein domains should be characterized and based on that the function is predicted (Rust et al., 2002). After that, a shared vocabulary should be used to describe the assigned function. The Gene Ontology (GO) (The Gene Ontology Consortium, 2000) is a well-recognized project that aims to create a controlled vocabulary of common terms to specify gene annotations and products (see Section 2.5.1.5).

The sequence comparison is carried out by lining up two (i.e., pairwise alignment) or more (i.e., multiple-sequence alignment) sequences. A *sequence alignment* can be done by comparing the sequences over their entire length (i.e., *global alignment*) or searching for partial homology (i.e., *local alignment*).

Currently, there is a great number of sequence comparison tools available. These tools are able to align, compare, cluster, and filter sequences, and perform different kinds of analysis. BLAST (Altschul et al., 1997) and its many derivatives (e.g., BLASTP, BLASTX, TBLASTX, and RPS-BLAST) are arguably the most widely used programs for pairwise local alignment (Edwards et al., 2009). Chapter 3 discusses this tool and other tools and systems to support the annotation process.

2.5. Functional annotation

A major challenge of the genomic and post-genomic era is detecting *genes* and determining their organization, structure, and function (Gouret et al., 2005). The *annotation* is the process where important features of a *genome* (or DNA segments) are identified and described. The annotation can be *structural* or *functional*. The first aims to identify genes, their structure, and location in the sequence, while the second tries to assign a biological function to the sequence. The current work covers only the last one.

Ideally, an expert should carry out the annotation process manually. However, the process is in this way labor-intensive and time consuming, and therefore, expensive. Thereby, this approach is only suitable for key reference genomes or relatively small regions of interest (Edwards et al., 2009). Alternatives consider automatic implementations of the annotation process (Bryson et al., 2001), since the steps involved in the annotation are almost all computer-based tasks: scanning sequence databases looking for similar sequences, collecting the matching sequences, and trying to infer the function of the query sequence from annotations of the matched proteins for which the function is already known. The automatic annotation is faster and less expensive than the manual one. Moreover, it can handle great volumes of data and keep the consistency of all analyses. Nevertheless, this approach produces less accurate annotations, as it can mix spurious information with important functional data.

The ultimate value of a genome sequence depends entirely on the quality of the accompanying annotation (Alterovitz and Ramoni, 2010). Therefore, this should be as reliable and accurate as possible. The majority of annotations are based on information derived from sequence homology (Curwen et al., 2004), and homology programs use databases that are fed mostly by automatic (sometimes manually curated) annotations. Thus, producing highly accurate annotations is crucial to prevent error propagation.

The quality of the annotation can be improved by collecting information that supports (or questions) the "inheritance through homology". This can be done through different databases (see Section 2.5.1) and tools (see Chapter 3) that can perform orthology prediction, family and domain detection, identification of conserved residues that affect function, prediction of transmembrane regions or signal peptide, and so on.

2.5.1. Databases

For both manual and automated approaches, there is a great amount of genetic information publicly available that can be used in the annotation process. Table 2.2 presents the most popular databases. These are constantly being updated, and for some of them there are available pre-formatted Search Databases (SDBs) ready to use with different BLAST flavors (Altschul et al., 1997; Marchler-Bauer et al., 2002). The National Center for Biotechnology Information (NCBI)[1], for instance, makes available SDBs (see Table 2.3) together with a script to keep them up to date[2].

There is also a variety of tools that support functional annotation and consult some biological databases for this purpose. The majority of them is based on sequence alignments and is distributed as stand-alone programs or are accessible through web servers. Besides these tools, there are some annotation systems available as well. These systems, in general, wrap some of these tools and use different approaches to infer the biological function. Both tools and systems are discussed on Chapter 3.

Despite the availability of these databases, compiling all the information about specific data is a complex task. Generally each database uses its own format (see Appendix A.1), and only in some cases these databases are linked to each other through cross-references. Some of them also store a huge amount of data. The majority of databases present statistics based on the number of sequences or bases (or AAs), instead of reporting their computational size. This makes difficult to know their real size. An approximation can be obtained from their SDBs. For instance, the CDD SDB is 6.3 GB [3], and NR is 94 GB [4]. It is important to take into account that the SDBs are formatted based on the

[1] `http://www.ncbi.nlm.nih.gov/`

[2] `http://www.ncbi.nlm.nih.gov/blast/docs/update_blastdb.pl`. [Online: accessed 4-September-2015]

[3] CDD v3.14, as of 16 January 2015. Downloaded from `ftp://ftp.ncbi.nlm.nih.gov/pub/mmdb/cdd/little_endian/Cdd_LE.tar.gz`. [Online: accessed 4-September-2015]

[4] Last updated on January 12, 2015 . Downloaded from `ftp://ftp.ncbi.nlm.nih.gov/blast/db/`. [Online: accessed 5-September-2015]

Data type	Database Name
Genome and genomic information	▪ Ensembl (Flicek et al., 2014)
Gene	▪ GenBank (Benson et al., 2013)
Transcripts	▪ Unigene (Schuler, 1997)
Protein	▪ Entrez Protein (NCBI Resource Coordinators, 2013) ▪ UniProtKB (Magrane and Consortium, 2011) (composed of SwissProt*and TrEMBL**)
Protein domain and family	▪ Conserved Domains Database (CDD) (Marchler-Bauer et al., 2013) ▪ InterPro (Hunter et al., 2012) ▪ Pfam (Finn et al., 2014) ▪ SMART (Letunic et al., 2012a) ▪ TIGRFAM (Haft et al., 2013) ▪ Protein Domain (ProDom) (Kahn et al., 2008) ▪ Protein ANalysis THrough Evolutionary Relationships (PANTHER) (Mi et al., 2013) ▪ Clusters of Orthologous Group (COG) (Wheeler et al., 2004) ▪ PRotein K(c)lusters (PRK) (Klimke et al., 2009) ▪ PROSITE (Sigrist et al., 2013)
3-D structures	▪ Protein Data Bank (PDB) (Bernstein et al., 1977b)
Ontology	▪ GO (Ashburner et al., 2000)
Enzymes	▪ ENZYME (Bairoch, 2000)
Metabolic pathways	▪ Kyoto Encyclopedia of Genes and Genomes (KEGG) (Kanehisa and Goto, 2000)

* Manually annotated and reviewed by experts.
** Automatically annotated and not reviewed by experts.

Table 2.2: Publicly available biological databases.

SDB name	Content
NR	Database of non-redundant protein sequences, with entries from GenPept (translations for each of the coding sequences within the GenBank (Benson et al., 2013)), RefSeq Proteins (Pruitt et al., 2014), PDB, SwissProt, PIR (Wu et al., 2003), and Protein Research Foundation (PRF) .*
NT	Database of nucleotide sequences, with entries from all traditional divisions of GenBank, RefSeq Nucleotides (Pruitt et al., 2014), EMBL**, DNA Data Bank of Japan (DDBJ)+, and PDB.
CDD	NCBI++ curated domains plus domain models from Smart, Pfam, TIGRFAM, and other sources.

* https://www.prf.or.jp/index-e.html
** European Molecular Biology Laboratory (EMBL) http://www.embl.de
+ http://www.ddbj.nig.ac.jp
++ NCBI National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov

Table 2.3: Examples of SDB publicly available at NCBI.

FASTA file format. Thus, these numbers just represent a part of the databases from where the sequences were retrieved. Besides this, the quality of the data can vary from source to source: some contain manually curated data and others not; there are sources more complete or accurate; and sometimes they can even present diverging information for the same entry. These issues just increase the complexity of the annotation task, and make even more necessary the involvement of experts during this process.

Although the system presented in this work can deal with any type of SDB and mine information from different sources, annotations were carried out here based on NR data and on five major databases: GenBank, UniProtKB, CDD, InterPro, and GO. These databases are briefly described next.

2.5.1.1. GenBank

GenBank, hosted by the NCBI, is the National Institutes of Health (NIH) genetic database. It stores a great collection of publicly available DNA sequences, currently having data related to almost 260.000 species (Benson et al., 2013) and 187.066.846 sequences [5]. These sequences can be directly submitted by individual laboratories or obtained through data exchanged with other databases. In order to provide a worldwide coverage, GenBank daily synchronizes information with the European Nucleotide Archive (ENA) and the DDBJ.

The data are accessible through the NCBI Entrez retrieval system, more specifically with Entrez Nucleotide [6]. This system gives access not only to the DNA and protein sequence databases, but also to biomedical literature (e.g., PubMed [7]). It is also possible to automatically retrieve information from the NCBI Entrez system through Entrez Programming Utilities (E-Utilities) (Sayers, 2010), a set of server-side programs that provide a stable interface to query Entrez databases. The Perl module Bio::DB::EUtilities is an

[5] http://www.ncbi.nlm.nih.gov/genbank/statistics. [Online: accessed 4-September-2015]
[6] http://www.ncbi.nlm.nih.gov/nucleotide/
[7] http://www.ncbi.nlm.nih.gov/pubmed

Application Programming Interface (API) to access this information.

GenBank information can be visualized in different formats. The standard is its own GenBank format (.gb) (see Appendix A.1.2).

2.5.1.2. UniProt

UniProt (Consortium, 2015) is a protein database made available by the European Bioinformatics Institute (EBI) [8] and organized in four core databases: UniProtKB (Magrane and Consortium, 2011), UniParc, UniRef (Suzek et al., 2015), and UniMes. UniProtKB is the most popular of these databases and comprehends UniProtKB/SwissProt and UniProtKB/TrEMBL. The first one contains manually annotated non-redundant protein data, whereas the second stores automatic annotated data. Together, these databases hold a great volume of data: SwissProt has 549.008 sequence entries from 13.222 species [9]; and TrEMBL 50.011.027 entries from 557.696 species [10].

UniProt data can be obtained in different formats, being the European Molecular Biology Laboratory (EMBL) format (see Appendix A.1.1) the default. Data automatic retrieval can be easily done using the Perl module LWP::UserAgent with the dataset name, the entry's accession, and the desired format [11]. For instance, it is possible to obtain the information regarding entry *P12345* from UniprotKB passing the url `http://www.uniprot.org/uniprot/P12345.txt` to the LWP::UserAgent.

2.5.1.3. CDD

CDD is a protein domain database that comprises a collection of models built up based on multiple alignments of ancient *domains* (i.e., slowly evolving and highly conserved *domains*) and full-length protein sequences. It contains *domains* from NCBI-curated domains (cd) (which are explicitly defined based on information on their 3D-structure), and from external sources (such as Pfam, SMART, COG, PRK, and TIGRFAM).

It is possible to query CDD about a given protein through CD-Search (see Section 3.1.3.3). This tool is publicly available at the CDD website[12] and performs a RSP-BLAST (see Section 3.1.1) of the query protein against the whole CDD database. This search provides the following results [13] that are taken into consideration in this work:

- **Specific hits:** it is a set of similar sequences (i.e., *hits*) sorted in descending order by *bit score*. The *bit score* is a measure that can give an indication of the statistical significance of the alignment (see Appendix A.2.2). Only those sequences which

[8] `http://www.ebi.ac.uk`

[9] `http://web.expasy.org/docs/relnotes/relstat.html`. [Online: accessed 4-September-2015]

[10] `http://www.ebi.ac.uk/uniprot/TrEMBLstats`. [Online: accessed 4-September-2015]

[11] `http://www.uniprot.org/help/programmatic_access`. Accessed on April 1s 2015.

[12] `http://www.ncbi.nlm.nih.gov/Structure/cdd/wrpsb.cgi`. [Online: accessed 4-September-2015]

[13] `http://www.ncbi.nlm.nih.gov/Structure/cdd/cdd_help.shtml`. [Online: accessed 4-September-2015]

bit score meets or exceeds a domain Threshold Bit Score (TBS) (NCBI - National Center for Biotechnology Information, 2013) are considered. A text file with the TBS information (TBSF) is provided by CDD [14]. The *specificity* indicates a very high level of confidence that the query is a member of the protein family matched. Consequently, it is possible to make a reliable inference of the query's function.

- **Non-specific hits:** it is a set of *hits* that are below the TBS or has an expectation value (*e-value*) above 0.01 (or the value the user sets). The *e-value* is a value that indicates the likelihood of a hit had happened by chance (see Section 3.1.1). In this work, non-specific hits just have a *bit score* lower than the TBS, because *hits* with an *e-value* higher than 1.0E-05 are automatically excluded.

- **Superfamily:** both *specific* and *non-specific hits* belong to a set of *domain* models called *superfamily* (i.e., a "general" *domain* created on the basis of multiple sequence alignments of related proteins that share AA patterns). A *superfamily* is a set of conserved domain models, and like the domains models, it is created based on overlapping annotations on the same protein sequences and it is supposed to cluster evolutionarily related *domains*. The *superfamily* accession number has the prefix "cl", which stands for *cluster*. The file *family_ superfamily_ links*[15] (FSLF) is the plain text that lists the members of each *superfamily*.

Besides the aforementioned, CDD also has files that store information about CSs (cddannot.dat[16]) and previous versions of the CDD database (cdd.versions [17]). Both files are used in this work in order to infer a more reliable annotation.

2.5.1.4. InterPro

InterPro (Hunter et al., 2012) is a database that includes most of the online information about protein families, domains, and functional sites. It integrates PANTHER (Mi et al., 2013), Pfam (Finn et al., 2014), TIGRFAMs (Haft et al., 2013), ProDom (Kahn et al., 2008), SMART (Letunic et al., 2012a), PIR (Wu et al., 2003) superfamily, and PROSITE (Sigrist et al., 2013) patterns and profiles.

InterPro works with *functional signatures*, that is, models obtained from conserved patterns found in multiple alignments of protein members of the same *family*. *Signatures* can also be created based on *motifs* that belong to *domains* and CSs. Some *signatures* can be specific of a whole *family* or *domain*, while others just appear in specific *domains*.

[14] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/bitscore_specific_3.14.txt`. [Online: accessed 4-September-2015]

[15] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/family_superfamily_links`. [Online: accessed 4-September-2015]

[16] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/cddannot_generic.dat.gz`. [Online: accessed 4-September-2015]

[17] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/cdd.versions`. [Online: accessed 4-September-2015]

2.5.1.5. GO

GO is a collaborative project that aims to create a common vocabulary for describing gene products across databases. Its information is organized in an ontology and structured as a directed acyclic graph divided in three main categories: Biological Process (BP), Molecular Function (MF), and Cellular Component (CC). Its terms have a name and an unique identifier, and are linked to a specific category (see Section A.3.2). These terms are associated with a list of genes, gene products, and annotations, and can also have synonyms and cross-references, along with other information.

GO is implemented with a MySQL database. It can be accessed remotely or installed locally[18]. Its information is updated frequently.

2.6. Conclusions

This chapter presents key biological concepts to understand the functional annotation problem. First, it briefly describes the cell structure, focusing on the element in charge of storing the hereditary information, the DNA. The DNA composition, structure and purpose are explained together with important concepts such as gene, RNA, protein, protein domain, and homology. The process the DNA goes through in order to synthesize proteins is also described.

In the light of this, the challenge of predicting the function of genes (i.e., *functional annotation*) is introduced. Some approaches to solve the problem are mentioned, but a special attention is given to the "inheritance through homology" approach, which is the most popular and feasible one. Following this line, the manual and functional annotations are discussed, pointing out the necessity of ally the accuracy of the first method with the automation of the second.

The chapter also gives a overview of the domain studied, doing a brief review of some of the most common databases used in the annotation process. It gives an idea of the volume of data they comprise and the variety of formats they use, both features that reinforce the need of automatic annotation.

[18] `http://geneontology.org/page/lead-database-guide`. [Online: accessed 4-September-2015]

Chapter 3

Annotation tools and systems

I do not fear computers.
I fear the lack of them.

Isaac Asimov

Many tools and systems have been developed to support functional annotation. The majority of them are publicly available through web servers, but stand-alone versions are also provided, specially for supporting tools. The tools are often based on sequence alignments and tackle just one aspect of the process (e.g., protein discovery, family/domain identification, or orthology prediction), while the systems try to combine supporting tools and make predictions based on a broader information. The systems use different computational approaches, but the knowledge they use is in general very similar. The most popular tools are described in Section 3.1, classified by their biological purpose. The annotation systems are presented grouped by their computational approach in Section 3.2. Tables 3.3 and 3.4 summarize tools and system features respectively.

3.1. Tools to support annotation

A variety of tools has been developed to support functional annotation. They generally accept a common input format, the well-known text-based file format FASTA (see Appendix A.2.1). However, they produce different types of outputs (see Appendix A.2), increasing the complexity of the annotation process. In order to work with different input and output formats, there are specific libraries in BioPerl (Stajich et al., 2002) and BioJava (Holland et al., 2008) that are able to retrieve, process, and parse these files (see Appendix A.2). Some of the most popular annotation tools are reviewed in this section.

3.1.1. BLAST, a multipurpose tool

BLAST Altschul et al. (1997) is one of the most popular tools for annotation. It searches in a SDB for segments (i.e., target sequences) that have a high homology with

the sequences from a query set, described in a FASTA file (see Appendix A.2.1). Based on the stochastic model Karlin-Altschul (Karlin and Altschul, 1990), it produces a report (see Appendix A.2.2) with the alignments found, *bit score*, the percent of the bases that match in the alignment (i.e., percent of identity), and the likelihood of a match had occurred by chance (i.e., *e-value*), along with other information. Albeit the *e-value* provides the most important measure of statistical significance, sometimes it is necessary to take into consideration the other reported values in order to produce a more accurate annotation. Besides, it is also important to use thresholds with the purpose of filtering false positive matches.

There are different flavors of BLAST that can be used for distinct purposes (see Table 3.1). Although their input is a FASTA file and a SDB, their kinds of query (i.e., DNA or protein) and SDB (i.e., nucleotide or protein) vary according to the BLAST version.

Type	**Input**	**SDB**	**Purpose**
BLASTN	DNA	Nucleotide	▪ Verifying DNA similarity
BLASTP	Protein	Protein	▪ Finding protein family members ▪ Predicting a protein function ▪ Predicting its 3-D structure
BLASTX	DNA*	Protein	▪ Discovering genes in a genome or a protein encoded in a sequence
TBLASTN	Protein	DNA*	▪ Discovering new proteins
TBLASTX	DNA*	DNA*	▪ Discovering protein and ESTs[1]
RPS-BLAST	Protein**	Protein	▪ Identifying conserved domains in a protein sequence
PSI-BLAST	Protein	Protein	▪ Revealing distant evolutionary relationships among proteins

* Translated to protein.
** It has an option to run translated searches of DNA sequences.

Table 3.1: Types of BLAST, their inputs, and purposes.

According to Table 3.1, BLASTX, RPS-BLAST, and PSI-BLAST are useful for functional annotation. However, according to (Radivojac et al., 2013), PSI-BLAST does not produce advantages over BLAST anymore.

3.1.2. HMMER

HMMER (Eddy, 1998) is a protein homology and alignment tool that uses profile Hidden Markov Models (HMMs) to search on sequence databases. Profile HMMs are statistical models of one single sequence or multiple alignments introduced to computational Biology in 1994 (Krogh et al., 1994). These models capture the level of conservation and the likelihood of each residue in the alignment (Eddy et al., 2015).

This tool accepts a protein FASTA input (see Appendix A.2.1) and generates a variety of outputs, such as plain text, XML, FASTA, aligned FASTA (see Appendix A.2.8),

[1] Expressed Sequence Tags (ESTs) are mRNA fragments obtained by single sequencing which can be useful to identify genes.

Clustal (see Appendix A.2.6), PHYLogeny Inference Package (PHYLIP) (see Appendix A.2.9), Tab-Separated Values (TSV), and JavaScript Object Notation (JSON). Scripts from the HMMER package are used by different tools and systems, like InterProScan (Jones et al., 2014) (see Table 3.2).

3.1.3. Domain and family prediction tools

Predicting the domain and family of a sequence can give some important insights into its function. This task becomes even more important when the query has no significant matches to nucleotide or protein sequences in the target database. This section describes some of the domain and family prediction tools available.

3.1.3.1. InterProScan

InterProScan (Jones et al., 2014) is a protein functional analysis tool that searches into the InterPro database (see Section 2.5.1.4) for *signatures* that can functionally characterize a set of sequences. It can classify the queries into families and predict the existence of domains and important sites. It is written in Perl and combines different signature recognition methods (Zdobnov and Apweiler, 2001) (see Table 3.2). These methods are used separately based on the database they process.

InterProScan can take either nucleotide or protein sequences in FASTA (see Appendix A.2.1) or EMBL (see Appendix A.1.1) formats. It can produce the output in a variety of formats: XML, Generic Feature Format version 3 (GFF3) (Welcome Trust Sanger Institute, 2012; Ensembl Genome Bowser, 2015) (see Appendix A.3.1), HTML, Scalable Vector Graphics (SVG), or TSV. Like BLAST, InterProScan output (see Appendix A.2.3) provides an *e-value* and information about the target sequence. InterProScan can also report the GO terms associated to the signature found, which does not happen with BLAST and it is very helpful for the annotation process.

Database	**Scanning method**
PROSITE patterns	*ScanRegExp*[*]
PROSITE profiles	*pfscan* from *Pftools*[**]
PRINTS	*FingerPRINTScan* (Attwood et al., 2000)
Pfam	*hmmpfam* from *HMMER 2.3.2* package (Eddy, 1998)
ProDom	*BlastProDom.pl*[***] which wraps BLAST and applies a special filter to it
SMART	*hmmpfam* from *HMMER 2.3.2* package (Eddy, 1998)
TIGRFAMS	*hmmpfam* from *HMMER 2.3.2* package[***](Eddy, 1998)
PANTHER	*hmmsearch* from *HMMER 2.3.2* package[***](Eddy, 1998) and BLAST

[*] By Wolfgang Fleischmann (Wolfgang.Fleischmann@ebi.ac.uk)
[**] By Philipp Bucher (Philipp.Bucher@isrec.unil.ch)
[***] By Emmanuel Courcelle (emmanuel.courcelle@toulouse.inra.fr), and Yoann Beausse (beausse@toulouse.inra.fr)

Table 3.2: Examples of InterProScan scanning methods.

3.1.3.2. PfamScan

PfamScan (Finn et al., 2014) receives a protein FASTA file (see Appendix A.2.1) and searches for domains against a library of Pfam HMMs. PfamScan output can be written in JSON format or plain text (see Appendix A.2.5).

3.1.3.3. CD-Search

CD-Search (Marchler-Bauer and Bryant, 2004) is a web-based tool that uses the RPS-BLAST (see Section 3.1.1) algorithm to search into CDD (see Section 2.5.1.3) for structural and functional domains. Although based on RPS-BLAST (Altschul et al., 1997), CD-Search is optimized regarding accuracy and running time, producing results slightly different from the stand-alone version of this tool. It accepts a FASTA (both nucleotide and AA) file and produces very visual results (see Appendix A.2.4).

CD-Search output looks intuitive and straightforward for the expert, but being user-oriented, it is not well-suited for machine processing. RPS-BLAST output, on the other hand, is plain text easily parseable. This advantage, together with the fact that CD-Search is web-based only, makes RPS-BLAST more suitable for pipeline integration.

3.1.4. Multiple alignments tools

Multiple alignment is a widespread technique that can be very helpful to study structural, evolutionary, and functional similarities. Due to this important role in Bioinformatics, there are several different implementations of it. Here, three popular multiple aligners are briefly presented.

3.1.4.1. ClustalW

ClustalW (Larkin et al., 2007) is the most popular multi-aligner. Its latest version was implemented in C++ and it accepts formats such as FASTA and EMBL (see Appendices A.2.1 and A.1.1) as input. ClustalW offers six different types of output, including FASTA alignment format (see Appendix A.2.8) and being the default the CLUSTAL format (see Appendix A.2.6).

3.1.4.2. T-Coffee

T-Coffee (Notredame et al., 2000) follows an approach similar to ClustalW, but it generates more accurate alignments. The cost of this quality is a higher running time. T-Coffee output is very similar to the ClustalW one (see Appendix A.2.7).

3.1.4.3. MUSCLE

MUSCLE (MUltiple Sequence Comparison by Log-Expectation) (Edgar, 2004) is a multiple alignment tool well-known for its fast and high-quality performance. It accepts

a FASTA file (see Appendix A.2.1) as input and generates a multiple alignment report in FASTA alignment format (see Appendix A.2.8) as output.

3.1.5. Phylogenetic tools

Phylogenetics is the biological field in charge of the study of evolutionary relationships among a set of organisms. This study is based on homology (see Section 2.3) and relies on the comparison of sequences from different species, and on the construction of a "genealogical" tree that can show the proximity between these species.

The majority of the phylogenetic tree-reconstruction techniques uses a table that contains the distances between the sequences of the data set. This *distance table* is created based on multiple alignments, like the ones produced by ClustalW (Larkin et al., 2007) (see Section 3.1.4.1), T-Coffee (Notredame et al., 2000) (see Secion 3.1.4.2), and MUSCLE (Edgar, 2004) (see Section 3.1.4.3).

There are many methods used to build phylogenetic trees (e.g., distance, parsimony, or likelihood) and many different tools that implement them. This section gives a brief overview of some of these tools.

3.1.5.1. PHYLIP

PHYLIP (PHYLogeny Inference Package) (Felsenstein, 2009) is a well-known package for inferring evolutionary trees. It provides different methods to construct these trees and also includes bootstrapping and consensus trees. This tool accepts inputs in the PHYLIP format (see Appendix A.2.9) and writes the tree results in the Newick format (see Appendix A.2.10).

3.1.5.2. Belvu

Belvu (Sonnhammer and Hollich, 2005) is better known as a multi-alignment viewer that can color residues by conservation or by residue type. It can also edit this information, although in a limited way. Besides, Belvu is a phylogenetic tool and it is capable of generating distance matrices among sequences and implementing distance-based tree reconstructions. When in phylogenetic mode, it accepts the FASTA alignment format (see Appendix A.2.8) and writes a Newick tree output (see Appendix A.2.10).

3.1.6. Orthology tools

As explained in Section 2.3, the prediction of orthologous sequences can increase the reliability of the functional annotation, because genes that share a common ancestor are more likely to keep their functionalities than the ones originated by duplication (i.e., parologous). However, finding orthologs is not a simple task, and it can get even more complicated due to biological processes such as gene transference or conversion (Storm and Sonnhammer, 2002). There are different approaches to predict orthology. Here, some of these methods are presented.

3.1.6.1. InParanoid

InParanoid (Sonnhammer and Östlund, 2015) is a tool created to identify orthologs. It uses pairwise similarities generated by BLAST (Altschul et al., 1997) to build ortholog clusters. It can be very accurate, but it requires the complete proteome (i.e., the set of proteins expressed by a genome) of two species as input. Thereby, it is not suitable for annotating sequences which specie has not been completely sequenced or sequences outside the proteome context.

3.1.6.2. Orthostrapper

The standard procedures for ortholog prediction are based on phylogenetic trees. Nevertheless, these trees are calculated based on some arbitrary parameters, which may impair the quality of results. Orthostrapper (Storm and Sonnhammer, 2002), instead of creating an optimal tree, analyzes a set of bootstrap trees and uses the frequency of orthology assigned to them as a support value for orthology inference. This tool is written in Java and uses Belvu (Sonnhammer and Hollich, 2005) (see Section 3.1.5.2) to calculate the bootstrap trees. It receives a FASTA alignment (see AppendixA.2.8) as input and writes a list of the sequences and their likelihood of being orthologs (see Appendix A.2.11).

Id.	Attribute	Description	Homolog sequence	Database	Output format	Availability
BLAST	▪ Different purposes See Table 3.1	▪ Local alignment	▪ FASTA	▪ Any sequence dataset (nucleotide or AA) can be formatted as SDB	▪ BLAST report ▪ XML ▪ TSV ▪ CSV	▪ Stand-alone ▪ Web server
HMMER	▪ Protein homology ▪ Pairwise and multiple alignment	▪ Profile HMMs	▪ protein FASTA	▪ Any AA sequence dataset properly formated	▪ Plain text ▪ XML ▪ FASTA ▪ FASTA alignment ▪ Clustal ▪ PHYLIP multiple alignment ▪ TSV ▪ JSON	▪ Stand-alone ▪ Web server
InterProScan	▪ Classify into families ▪ Predict domains and important sites ▪ Add GO terms	▪ Local alignment ▪ Other methods See Table 3.2	▪ FASTA ▪ EMBL	▪ InterPro	▪ XML ▪ GFF3 ▪ HTML ▪ SVG ▪ TSV	▪ Stand-alone ▪ Web server
PfamScan	▪ Domain search	▪ HMM	▪ protein FASTA	▪ Pfam	▪ JSON ▪ Plain text	▪ Stand-alone ▪ Web server
CD-Search	▪ Domain search	▪ Local alignment	▪ protein FASTA	▪ Any AA sequence dataset can be formatted as SDB	▪ BLAST report ▪ XML ▪ TSV ▪ CSV	▪ Web server
ClustalW	▪ Aid in the study of structural, evolutionary and functional similarities	▪ Multiple alignment	▪ FASTA ▪ EMBL	▪ NA*	▪ Clustal ▪ FASTA alignment ▪ Phylogenetic tree	▪ Stand-alone ▪ Web server
T-Coffee	▪ Aid in the study of structural, evolutionary and functional similarities	▪ Multiple alignment	▪ FASTA	▪ NA*	▪ FASTA alignment ▪ Simple multi-alignment	▪ Stand-alone ▪ Web server

MUSCLE	▪ Aid in the study of structural, evolutionary and functional similarities	▪ Multiple alignment	▪ FASTA ▪ Alignment format	▪ NA*	▪ FASTA alignment ▪ HTML ▪ Clustal ▪ Simple multi-alignment	▪ Stand-alone ▪ Web server
PHYLIP	▪ Inferring evolutionary trees	▪ Parsimony ▪ Distance matrix ▪ Likelihood ▪ Bootstrapping ▪ Consensus trees	▪ PHYLIP multiple alignment	▪ NA*	▪ Newick	▪ Stand-alone ▪ Web server
Belvu	▪ Multiple alignment viewer ▪ Multiple alignment editor ▪ Tree reconstruction	▪ Distance matrix	▪ FASTA alignment	▪ NA*	▪ Newick	▪ Stand-alone ▪ Web server
InParanoid	▪ Orthology inference	▪ Local alignment	▪ Complete proteome of two species	▪ NA*	▪ Own format	▪ Stand-alone** ▪ Web server
Orthostrapper	▪ Orthology inference	▪ Analysis of bootstrap trees	▪ FASTA alignment	▪ NA*	▪ Own format	▪ Stand-alone ▪ Web server

Table 3.3: Supporting tools for the annotation problem.

* Not applicable.
** Only available upon request to the author.

3.2. Annotation systems

There are systems that propose complete solutions to the functional annotation problem. These systems, in general, take advantage of the existing stand-alone tools, combining them with some knowledge and heuristics in the domain. They tend to employ the same knowledge, but are implemented following different computational approaches. This section presents systems that follow AI approaches, such as Knowledge-Based Systems (KBSs) and Multi-Agent Systems (MASs) (see Sections 3.2.1, 3.2.2, and 3.2.3), as well as non-AI systems (see Section 3.2.4). It also explains the methodology used to assess the performance of the functional annotation methods presented in the first large-scale community-based CAFA, and describes the three-top performing systems in this evaluation (see Section 3.2.5).

3.2.1. Knowledge-based systems

KBSs are intelligent systems that use knowledge and inference methods to solve complex problems that require human expertise. Besides being able to emulate the human reasoning process in some aspects, these systems have many attractive features. First, they increase the availability of expertise, since it is accessible on any computer that supports the systems. Second, the cost of providing expertise per user is reduced. Third, unlike human experts, KBSs do not retire, quit, die, get sick or tired, or experience any emotional changes. Thereby, the knowledge lasts indefinitely and it is not biased by emotions, stress, or fatigue. Fourth, KBSs can comprise the knowledge of multiple experts, even from different fields. Fifth, these systems have mechanisms to explicitly explain in detail the reasoning process that led to the conclusion. Sixth, KBSs may respond faster than the human expert. All these qualities explain why KBSs have been used for decades and have been applied to multiple domains, including in Biology.

One of the most popular types of KBS is the Rule-Based Expert System (RBES) (Giarratano and Riley, 1998). Besides the benefits common in KBSs, this type has the advantage of being very modular, making easy to encapsulate the knowledge and evolve the system. Moreover, rules are a natural way of structuring the knowledge on procedural solving problem. Using rules also promotes the traceability of the reasoning process. These features makes this type of system more straightforward to understand and test with the help of experts.

According to (Giarratano and Riley, 1998), a RBES is organized as depicted in Figure 3.1 and is composed of:

- **Knowledge Base (KB):** a set of rules.
- **Working memory (WM):** the set of facts that are true at a given moment. They are used by the rules.
- **Inference engine:** carries out the inference process, deciding which rules are currently satisfied by the facts and updating the WM.

- **Agenda:** a list of rules satisfied by the facts in the WM at a given moment. They are sorted by some priority criteria. High priority rules are executed first by the inference engine.
- **User interface:** the mechanism that allows the user-system interaction.
- **Explanation facility:** describes the system reasoning process to the user.
- **Knowledge acquisition facility:** an automatic way for the user to populate the KB. This component is an optional feature in many systems.

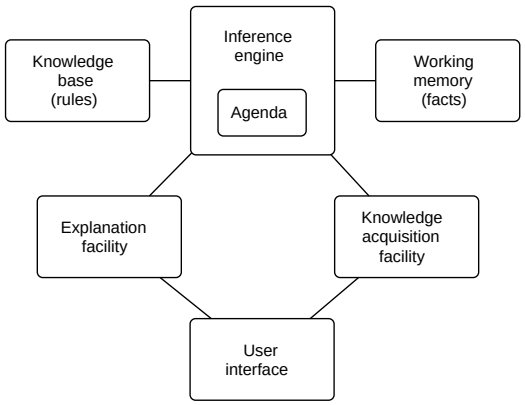


Figure 3.1: Components of a RBES.

KBSs, and in particlar RBESs, seem quiet advantageous for functional annotation prediction, since they can be a solution for the expert bottleneck without losing the annotation quality. However, their usage is not very popular for this specific problem. This section presents some KBSs developed for functional annotation.

3.2.1.1. FIGENIX

FIGENIX (Gouret et al., 2005) is a system that automates the pipelines for structural and functional annotation. It incorporates a RBES that supervises the annotation process, making key decisions, checking intermediate results, and refining the dataset. Its architecture is composed of three layers: database, server-side components, and graphical interface. It was developed in Java and uses POSTGRESQL as database management system.

The system has an "Annotation Engine" responsible for running several annotation tools in parallel and consulting local copies of NR, SwissProt, Ensembl, and Pfam. This engine works in association with the RBES, which integrates static empiric rules based on expert knowledge and dynamic information generated during the execution. The RBES uses backward chaining and is implemented in PROLOG.

The functional annotation pipeline is based on phylogenomic inference and incorporates: a BLAST (Altschul et al., 1997) against NR, SwissProt (Magrane and Consortium, 2011), and Ensembl (Flicek et al., 2014); *hmmpfam* (Eddy, 1998) for domain detection; multiple alignments generated by ClustalW (Larkin et al., 2007); phylogenetic reconstruction based on multiple alignments; orthology inference based on the created tree; and functional annotation based on the GO database (Ashburner et al., 2000) and other online sources.

Although FIGENIX can produce reliable functional annotation, it has some drawbacks. The system is only accessible through a web server [2], which makes it closed for users regarding the tools it executes, their parameters, the databases they use, or the size of the input.

In 2012, FIGENIX developers released GLADIX (Dainat et al., 2012), a MAS combined with a KBS. The system is publicly available through a virtual machine. It is just capable of calculating orthology, and it does not generate functional annotations.

3.2.1.2. EAP

EAP (Potter et al., 2004) is an automated annotation pipeline of genomic sequences. It is written in Perl, uses MySQL to store data, and comprises two parts. The first is a set of Perl modules (*Runnable* and *RunnableDB*) that wraps some popular analysis tools. The second lays on top of the wrappers and acts as a job submission system (*RuleManager*), controlling the submission of a large number of job to a computer farm.

Runnables are in charge of executing analysis tools such as BLAST (Altschul et al., 1997), GENSCAN (Burge and Karlin, 1997), and RepeatMasker (Smit et al., 2013-2015). In the pipeline, both input and output are written/read to/from the database. In order to keep the pipeline modularity, each *Runnable* has a *RunnableDB* associated to it.

The *RuleManager* is a script that automatically runs all the analyses in the pipeline. It works as a job scheduler, looking for available input sequences and submitting them to the farm based on their dependencies. These dependencies are called *Rules*. There are three types of *Rules* grouped according to their dependency level. The first is related to the job that does not depend on any previous analysis. The second is the job that depends on the completion of one or more prior steps. The third is the one that requires that all jobs related to the input have finished. Like the input and output, running jobs and rules information is stored in the database.

After running the gene building pipeline (e.g., BLAST (Altschul et al., 1997), GENSCAN (Burge and Karlin, 1997), and RepeatMasker (Smit et al., 2013-2015)), InterPro (Hunter et al., 2012) domains are assigned to the predicted protein through a protein annotation pipeline. This pipeline comprises two types of analysis: one involving InterPro and its components; and the other using protein annotation approaches like transmembrane prediction (e.g., TMHMM (Krogh et al., 2001)), signal peptide inference (e.g., SignalP (Petersen et al., 2011)), coiled coil (e.g., ncoil (Lupas et al., 1991)), and low-complexity (e.g., Seg (Wan and Wootton, 2000)) identification.

[2] `http://ioda.univ-provence.fr/IodaSite/Figenix.jsp`

EAP has been successfully used to annotate genes from human, mouse, rat, zebra fish, and other species. It has been tailored to process whole genomes, not being suitable for sequences outside this context. Moreover, its local installation is not straightforward, impairing its usage at small labs which lack of computer experts.

Its architecture also presents some limitations. Although it has a RBES, the knowledge comprised by the KB is basically related to priorities in tool execution, not applying explicitly any further biological reasoning process. Encapsulating this knowledge inside the code hinders the system evolution. Another issue that compromises EAP evolution is the fact that it uses its own inference engine instead of using a widespread and better supported one.

3.2.1.3. BioMediator

BioMediator (Cadag et al., 2007) combines a data integration system with an inference engine in order to elucidate functional annotations. It uses a federated database system, not storing any information locally. The queries are sent to the sources (e.g., CDD (Marchler-Bauer et al., 2013), Pfam(Finn et al., 2014), PROSITE (Sigrist et al., 2013), and ProDom (Kahn et al., 2008)) to be processed and analyzed in real-time. A client-side footprint is created to track the query status. Although this has the advantage of using up-to-date data, it slows down the system execution, besides of having all the limitations related to web-based execution.

The KB was populated mainly with the knowledge acquired through the annotation of *Haemophilus influenzae*, *Leishmania major*, *Trypanosoma brucei*, and *Trypanosoma cruzi*. The inference system was implemented using the Java Expert System Shell (Jess) (Sandia National Laboratories, 2009), and it can act upon the data at any stage of the process. After processing the data, the annotations are sorted based on their level of evidence, and the top one, which has the greatest amount of evidences, is transfered to the query.

BioMediator seems to be a very flexible and promising solution to the annotation problem. However, it is currently not available, neither as a web server nor as standalone software.

3.2.2. Multi-agent systems

A MAS is a group of interacting, potentially heterogeneous, agents. These Intelligent Agents (IntAs) are computational entities ranging from simple reactive ones to those complex, deliberative, and autonomous. Among other features, MASs are able to take advantage of computational resources across different platforms, being a natural way of representing and building distributed systems. They can also incorporate different applications into the agent society through wrapper agents. IntAs frequently communicate to each other through a message system supported by agent platforms. They can also offer solutions to manage expert knowledge, usually through deliberative agents. Moreover, MASs enhance other system features, such as the overall performance, efficiency, reliability, extensibility, robustness, maintainability, responsiveness, flexibility, and reusability

(Zain et al., 2011).

IntAs have proved to be advantageous for applications that imply repetitive and time-consuming activities, and also require knowledge management, such as integrating multiple information sources and modeling of complex, dynamic systems. Moreover, since agents are proactive and reactive (at least in most of the literature), they can deal with problems in a more human-like way than other software paradigms.

All these features make MASs very suitable for the studied domain. However, just a few bioinformatics tools have been developed following this approach (Merelli et al., 2007). Most of them can be considered hybrid systems that combine MASs with other AI approaches. These are reviewed later in Section 3.2.3. Next section describes one example of pure MAS approach for functional annotation.

3.2.2.1. GeneWeaver

GeneWeaver (Bryson et al., 2001) is a MAS that combines analysis tools and databases in a flexible and robust way. It was developed aiming to address some of the main problems in Bioinformatics: data integration and management, and data analysis.

This system comprises a community of agents that interact with each other to automate bioinformatics tasks, including functional annotation. These agents are:

- **Broker Agents:** register information about other agents.
- **Primary Database Agents:** manage remote databases (e.g., PDB (Bernstein et al., 1977b), PIR (Wu et al., 2003), SwissProt, or TrEMBL (Magrane and Consortium, 2011)), keeping them up-to-date and in a format comprehensible for the other agents.
- **Non-redundant database Agents:** construct non-redundant databases based on the data managed by the *Primary Database Agents*.
- **Calculation Agents:** wrap and integrate analysis tools for different purposes such as: BLAST (Altschul et al., 1997) and FASTA (Pearson and Lipman, 1988) for homology search; InterProScan for motif patterns search; ClustalW (Larkin et al., 2007) for multiple alignments; PhD (Rost and Sander, 1993) and PSIPRED (Buchan et al., 2013) for secondary structure prediction; MEMSAT (Jones et al., 1994) for membrane topology prediction; and Threader (Jones et al., 1992) and GenThreader (Jones, 1999) for fold recognition.
- **Genome Agents:** manage genomic data of specific organisms.

GeneWeaver is implemented in Java and uses the BioAgent Language (BAL) (i.e., a language specified following the lines of the Knowledge Query and Manipulation Language (KQML) (Mayfield et al., 1996)) for communication. Its default transport communication is the Remote Method Invocation (RMI) (Farley, 1998). Agents can be implemented

in other languages using the Common Object Request Broker Architecture (CORBA) (Vossen, 1997) as long as they support BAL.

The system is promising, but it is more centered in data integration and management than in functional annotation. Moreover, it lacks of a KBS, being its knowledge inside the code. Also, it is not publicly available.

3.2.3. Hybrid systems

As mentioned before, KBSs can alleviate the expert's workload and improve the automatic annotation accuracy, while MASs can handle heterogeneous and distributed resources and an evolving environment. Some systems combine the strengths of KBSs and MASs, being a potential solution for the annotation problem. There are a few systems for annotation that follow this approach. Next subsection present two of the most relevant.

3.2.3.1. BioMAS

BioMAS (Decker et al., 2002) is a MAS that incorporates some KBSs for genomic annotation. It was initially developed for automating the annotation and data storage of herpes viruses. It is implemented in Java and developed through the DECAF (Graham et al., 2003) MAS toolkit, following the RETSINA (Decker and Sycara, 1997; Decker et al., 1997; Sycara et al., 1996) information gathering model and TÆMS (Decker and Lesser, 1993; Wagner et al., 1997).

This system comprises four overlapping agent communities. They are in charge of integrating remote gene sequence annotations from different sources with the gene sequences at the Local Knowledge Base Management Agent (LKBMA). This allows complex queries to the LKBMAs through a web interface, obtaining information in order to infer the gene function and assigning GO terms, and structural annotation.

In order to carry out these tasks, BioMAS has *Information Extraction Agents* (IEAs), *Task Agents* and *Interface Agents*. The first ones wrap public web sites (i.e., ProDom (Kahn et al., 2008), SwissProt (Magrane and Consortium, 2011)/PROSITE (Sigrist et al., 2013), PSORT (Nakai and Horton, 1999), and GenBank (Benson et al., 2013)). The second group can be of two types. *Annotation Agents* which control the annotation process, storing the query sequences and their annotations, submitting queries to the IEAs, and indicating the source of annotations. *Sequence Source Processing Agents* test the input for internal consistency. The third group allows the user to add knowledge (e.g., manual annotate) to the local KB and to query the complete annotated KB.

When functionally annotating a sequence, BioMAS has two *Task Agents* in charge of assigning GO terms. The *Ontology Agent* manages the GO information and some cross-references to other ontologies, and it is able to map non-GO terms to GO terms. The *Ontology Reasoning Agent* wraps an algorithm for inferring GO terms for unknown gene products.

Although the algorithm encapsulated by the *Ontology Reasoning Agent* uses rules to deduce the ontology, it cannot be considered a proper KBS, since the rules seem to

be hard-coded and it lacks an inference engine. Moreover, the KBSs in BioMAS are more related to data management, having a limited KB that is not related to functional annotation.

BioMAS is well-designed and very modular and flexible, but wrapping web tools limits the system knowledge and performance. The system also has another constraints: important inference knowledge is inside the code, and there are no experiments in the literature describing its accuracy. Besides, it seems to be web-based, but apparently it is not publicly available.

3.2.3.2. EDITtoTrEMBL

EDITtoTrEMBL (Möller et al., 1999, 2001) is a MAS that integrates a RBES. It was developed for automating the annotation process, and has a special focus on the semantic consistency of annotations. It was developed in Java, uses RMI for inter-process communication and distribution, and the RBES is based on logic programming with Well-Founded Semantics with eXplicit negation (WFSX) (Alferes and Pereira, 1996).

This system treats the annotation as a workflow. It provides a flexible software framework for analysis, being able to deal with in-house or external programs (Möller et al., 1999). It dynamically decides which analysis each sequence must undergo, based on the declarative description of the previous tools. Thereby, the annotation pipeline can vary according to the query sequence.

The MAS comprises two types of agents, *Dispatchers* and *Analyzers*. The first ones are responsible for registering the *Analyzers* and mediating and facilitating the contact with them, controlling the flow of entries among these agents. The second group wraps the incorporated heterogeneous data sources, providing an homogeneous environment and consistent use of the vocabulary.

Analyzers can search in databases (e.g., ENZYME (Bairoch, 2000), PROSITE (Sigrist et al., 2013), Pfam (Finn et al., 2014), and PRINTS (Attwood et al., 2000)), and execute applications to enrich the annotation (e.g., TMHMM (Krogh et al., 2001) for prediction of transmembrane proteins, or NNPSL (Reinhardt and Hubbard, 1998) for prediction of sub-cellular location). Besides, these agents are able to set parameters for each individual query as well as rerun the tools with different settings in order to find an optimal result. *Analyzers* are also responsible for ensuring the consistency of the outcomes, using a controlled vocabulary and some manually curated set of rules from SwissProt (Magrane and Consortium, 2011) to assess the output quality and rephrasing it when needed.

EDITtoTrEMBL is well-structured and robust. Besides, the fact that there is no pre-established annotation workflow, being the annotation guided by the analysis outcome, is close to the actual expert's approach. However, the KB is more used to control the interaction between agents and to create correct vocabulary than to apply expert knowledge during the inference process. Moreover, WFSX is not a well-known and supported language, which can impair the system evolution. EDITtoTrEMBL apparently is only accessible by the EBI, not being publicly available.

3.2.4. Other approaches

The majority of the annotation systems do not apply any AI approaches in their development. In fact, they often work more as pure pipelines, chaining inputs/outputs and tools, rather than trying to apply expert knowledge to solve the problem. However, they can be very successful with the annotation. This section presents some of these systems.

3.2.4.1. BASys

BASys (Bacterial Annotation System) (Domselaar et al., 2005) is a web server [3] for the detailed automatic annotation of bacterial sequences. It runs more than 30 programs in order to obtain a range of annotation related information such as gene/protein name, GO function, COG function (Wheeler et al., 2004), possible paralogs and orthologs, secondary and tertiary structure, signal peptides, reactions, or pathways.

The system architecture has three layers: front-end web interface, annotation engine, and reporting system. The first is in charge of data submission, and annotation scheduling, monitoring, and reporting. The second integrates the database comparison with the computational sequence analysis. The third shows the outcomes in a user-friendly way.

BASys has two rounds of annotation. Initially, it attempts to annotate the data through a BLAST (Altschul et al., 1997) against UniProtKB (Magrane and Consortium, 2011) and CyberCell (Sundararaj et al., 2004) (an *Escherichia coli* database), and if the results meet some criteria the annotation is transfered. Otherwise, the system tries to fill the remaining annotations through additional similarity searches on other databases such as NR of bacterial sequences, PDB (Bernstein et al., 1977b), and COG (Wheeler et al., 2004). In order to enrich the annotation, other sources like Pfam (Finn et al., 2014) and PROSITE (Sigrist et al., 2013) are used, together with tools for predicting signal peptide and transmembrane domain (e.g., PredictSPTM (Cruz, Unpublished data)) and secondary structure (e.g., PSIPRED (Buchan et al., 2013)).

Although BASys creates a very detailed and visual output, this system is only available as a web server, having all the typical constraints of them (see Section 3.2.1.1). Moreover, even though there are rules for annotating sequences, they are hard-coded instead of being part of a KB. Another issue, is the fact that BASys can just deal with bacterial data, which also limits its usage.

3.2.4.2. Blast2GO

Blast2GO (Götz et al., 2011) is an annotation and visualization tool developed specifically for biologists (i.e., not computer expert users). It is a Java application available in three versions: as a web-based application (with Java Web Start), stand-alone graphical interface, and stand-alone bash mode. All the versions execute an homology search through BLAST (Altschul et al., 1997) at NR and other databases from NCBI. This

[3] `http://wishart.biology.ualberta.ca/basys`

search is carried out through NCBI's servers, but can be done locally in the bash version. Custom databases can also be added to the BLAST (Altschul et al., 1997) search for the local versions of the tool.

The system uses an heuristic based on the *e-value* and a minimal alignment length to retrieve significant hits from BLAST (Altschul et al., 1997). The values of this heuristic can be set by the user. GO terms associated with each significant hit are mapped through cross-references, generating at the end of the process a list of candidate annotations from different hits.

Aiming to enrich the annotation, Blast2GO also performs an InterPro (Jones et al., 2014) search (see Section 3.1.3.1) for protein domain information. This search is carried out on the EBI servers and its input is limited, which is common in web-based tools. However, there is an option to submit unlimited sequences upon registration.

The system also supports manual curation and the comparison between groups of GO terms. An annotation coherency function is implemented, based on GO Consortium (Ashburner et al., 2000) rules, in order to maintain the consistency of the GO annotations.

Blast2GO invests in visual outputs, producing different types of interactive graphs. Besides GO terms, it also support vocabularies like KEGG (Kanehisa and Goto, 2000), InterPro (Hunter et al., 2012), and Enzyme Commission numbers (EC numbers) [4].

Blast2GO adopts a user-friendly approach (at least in its graphical version) and it is said to have a high annotation accuracy (65-70 %) (Chen et al., 2012; Conesa et al., 2005). However, it was developed as a semi-automatic annotation system, therefore requiring manual curation to get highly accurate results. The system also presents constraints regarding its design and implementation. First, its knowledge is hard-coded. Second, it mainly relies on web-servers to execute tools like BLAST (Altschul et al., 1997) and InterProScan (Jones et al., 2014), which may limit the input, the parameters, and the system efficiency. Third, its bash version does not comprise all its functionalities, being basically able to perform BLAST (Altschul et al., 1997) and GO mappings.

3.2.4.3. FastAnnotator

FastAnnotator (Chen et al., 2012) combines some annotations tools (i.e., LAST (Sheetlin et al., 2005), Blast2GO (Götz et al., 2011) (see Section 3.2.4.2), PRIAM (Claudel-Renard et al., 2003), and RPS-BLAST (Altschul et al., 1997)) in a web-based annotation system. The system was implemented in Python and Perl, and consists of four stages: finding the best hits in NR with LAST (Sheetlin et al., 2005); assigning GO terms with Blast2GO (Götz et al., 2011); enzyme classification with PRIAM (Claudel-Renard et al., 2003); and domain identification through RPS-BLAST (Altschul et al., 1997) against Pfam (Finn et al., 2014).

The system is said to be able of efficiently annotating sequences (Chen et al., 2012). Among its drawbacks are that it is only available through a web server [5], encapsulates all the knowledge inside the code, and there are no comparisons of its performance with

[4] A numerical classification for enzymes.

[5] `http://fastannotator.cgu.edu.tw`

other systems.

3.2.4.4. AutoFACT

AutoFACT (Koski et al., 2005) is a functional annotation system developed to overcome some limitations in the domain (i.e., web-based submission, substantial manual intervention, limited output formats, and small range of information resources). This system is able to analyze both protein and nucleotide inputs; indicate the most informative functional description based on multiple BLAST (Altschul et al., 1997) reports; assign metabolic pathways and enzyme information, besides the usual GO terms; and generate output in HTML, GFF (see Appendix A.3.1), and plain text.

When annotating a nucleotide entry, the system tries different workflows and databases until it is able to generate an annotation. Firstly, the system performs a search in a rRNA database. If a match that fits a minimum length and percent of identity criteria is found, the ribosomal annotation is transfered. Otherwise, the system uses the databases specified by the user to carry out further searches. These databases are sorted by the user based on their informative level. In order to determine the most informative descriptions, the system searches for hits containing uninformative expressions such as "hypothetical", "unknown", "chromosome", and others (Koski et al., 2005). These hits are excluded from the potential annotation dataset. The top informative hit which has a top informative synonym in other searched database is transfered as annotation. At this point, extra information (e.g., COG (Wheeler et al., 2004) functions, KEGG (Kanehisa and Goto, 2000) pathways, GO terms, and EC numbers) are also assigned. If no informative hit is found or if there is no match against any of the databases, the system executes a RPS-BLAST (Altschul et al., 1997) against Pfam (Finn et al., 2014) and SMART (Letunic et al., 2012a) in order to annotate based on domains.

Even though AutoFACT has been able to overcome some of the constraints it was designed to, it has still several limitations. The system has many rules and all are kept inside the code, which makes the decision process less transparent to the user and hinders its evolution. Besides, the system uses limited knowledge to annotate the sequences, since instead of combining homology and domains to infer the annotation, it relies just on the homology when it is found.

3.2.5. CAFA: the first large-scale community-based Critical Assessment of protein Function Annotation

Developing systems able of precise automatic annotation is a big challenge faced by Bioinformatics, but evaluating these systems is a very difficult task as well. Besides complex, the annotation assessment can be subjective and be impaired by the lack of usage of proper vocabulary. Even though many works compare the performance of different systems and "calculate" how good they are, so far, no methodology has been considered by the research community as the gold-standard.

CAFA (Radivojac et al., 2013) is the only large-scale experiment that has proposed

a benchmark to evaluate functional annotation outcomes. Its first round has evaluated 54 different systems using 866 sequences from 11 phylogenetically distinct species [6].

Its methodology follows two different approaches to achieve this goal. The first is a *protein centric* one, which aims to answer the question "what is the function of a particular protein". The second one is *term centric* and focus on the proteins associated with a particular functional term. Since the goal of this work is centered in inferring the protein function, just the first approach was taken into consideration.

Section 3.2.5.1 explains the *protein centric* metric, while Section 3.2.5.2 describes the three methods which had the best performance in the CAFA experiment.

3.2.5.1. CAFA protein centric methodology

The *protein centric* method uses a metric based on *precision*, *recall*, and the *F-measure*, which has been broadly employed in pattern recognition and information retrieval. This metric compares the GO terms obtained by the annotation process, along with the scores related to them, to manually curated data from SwissProt (Magrane and Consortium, 2011), and calculates a value that indicates the performance of the annotation method. It has the advantage of being fairly intuitive and easy to interpret the results.

CAFA measurement works as follows. Let CPGO the GO terms automatically predicted and EPGO the GO terms obtained by manual annotation. The *precision* (see Equation (1)) is the fraction of GO terms that were correctly predicted (CPGO ∩ EPGO) rather than incorrectly.

$$precision = \frac{\sum(CPGO \cap EPGO)}{\sum CPGO} \tag{1}$$

The *recall* (see Equation (2)) is the fraction of GO terms that were correctly inferred (CPGO ∩ EPGO) rather than missed.

$$recall = \frac{\sum(CPGO \cap EPGO)}{\sum EPGO} \tag{2}$$

These metrics are then combined in a single performance measure, the *F-measure*. It is calculated as the harmonic meant between *precision* and *recall* (see Equation (3)).

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3}$$

Precision and *recall* can be considered with different decision thresholds (in [0,1]). If the algorithm is able to provide these thresholds, based on them a *precision-recall curve*

[6] CAFA specie (and number of sequences): *Pseudomonas aeruginosa* (2), *Streptococcus pneumoniae* (25), *Bacillus subtilis* (16), *Arabidopsis thaliana* (86), *Saccharomyces cerevisiae* (5), *Xenopus laevis* (16), *Homo sapiens* (285), *Mus musculus* (231), *Rattus norvegicus* (45), *Dictyostelium discoideum* (2), and *Escherichia coli* K-12 (153).

can be drawn. Otherwise, if the algorithm outputs only a fixed score (for instance 1), its performance will be represented by just a single point. The same idea applies to the *F-measure*, being the F_{max} the maximum value over all algorithms thresholds. F_{max} value varies between 0 and 1, being 1 a perfect predictor.

CAFA's assessment was based on two GO (Ashburner et al., 2000) categories, BP and MF. Thereby, a F_{max} was calculated for each one. In the case of MF, annotations described just as "protein binding" were excluded from the analysis, since this term is not considered informative by itself. Figure 3.2 depicts results for the top-performing methods for both categories together with two baseline methods (BLAST and Naive[7]).

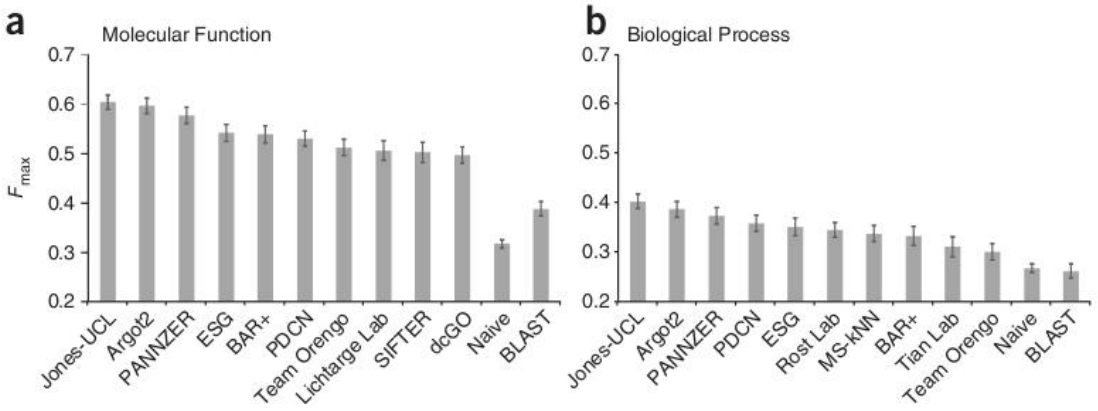


Figure 3.2: (a,b) F_{max} for the ten top-performing and baseline methods for MF (a) and BP (b). Confidence intervals of (95 %) were calculated using bootstrap with 10,000 interactions on the set of the annotated sequences.

Source: Radivojac et al. (2013)

CAFA also performed a domain analysis on eukaryotic sequences, separating the ones associated to single domains from the ones with multidomains. This analysis showed that the majority of the sequences are single-domain (see Figure 3.3).

3.2.5.2. CAFA three top-performing algorithms

54 systems were assessed by the CAFA experiment. 33 of them outperformed BLAST (Altschul et al., 1997) in the MF category, while 26 had better results than this baseline in the BP category. The three top-performing methods were Jones-UCL (Cozzetto et al., 2013), Argot[2] (Falda et al., 2012), and PANNZER (Koskinen et al., 2015). They presented high performance in both categories (MF and BP). In this section they are briefly described.

[7]Baseline method that uses the prior probability of each term in the database of manually annotated proteins as the prediction score for that term. For instance, the term "protein binding" appears with relative frequency of 0.25, so each target protein was associated with score 0.25 for that term. Therefore, this method assigned the same predictions to all targets.

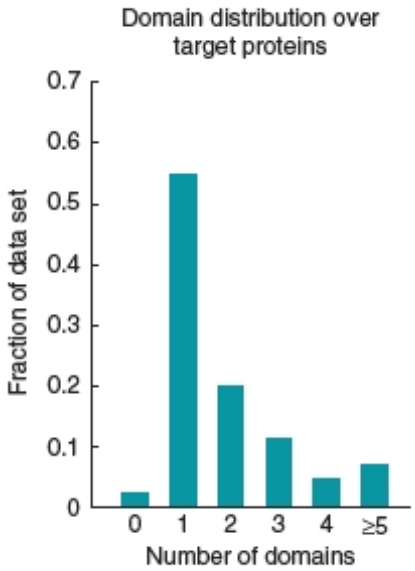


Figure 3.3: Distribution of target proteins with respect to the number of Pfam domains they contain.

Source: Radivojac et al. (2013)

Jones-UCL

Jones-UCL (Cozzetto et al., 2013) is a system that integrates a range of tools and data from different sources to generate reliable annotations. It performs a PSI-BLAST (Altschul et al., 1997) against UniRef90 [8] (Suzek et al., 2015) to obtain possible candidates. For the query sequences with hits, the system predicts GO terms from SwissProt (Magrane and Consortium, 2011) through text-mining and from tri-gram mining using a Naive Bayes approach (McCallum and Nigam, 1998). The system runs FFPRED (Minneci et al., 2013) to predict GO for eukaryotic queries, and applies a Support Vector Machine (SVM) regression, called FunctionSpace (Lobley, 2010), on high-throughput data information (e.g., microarrays) for the same purpose. Data from the eggNOG collection of orthologous (Powell et al., 2014) is also applied to infer GO terms based on evolutionary distance. GO terms inference is also made from profile-profile comparison using a PSI-BLAST (Altschul et al., 1997) in order to detect very distant evolutionary relationships. After executing all methods, their results are combined through a network propagation algorithm based on the GO (Ashburner et al., 2000) graph structure, and a final list of GO terms is produced.

Jones-UCL annotation with multiple data sources was the top-performing in the

[8] UniRef is part of UniProt (Consortium, 2015)(see Section 2.5.1.2)

CAFA experiment, having a $F_{max_{MF}} \approx 0{,}6$ and $F_{max_{BP}} \approx 0{,}4$. Despite its performance in CAFA, the system has some drawbacks. First, it does not use any KBS, being all the knowledge hard-coded. That issue does not only hinder the software evolution, but it also makes very difficult for experts to understand the process: there are many programs running in parallel and decisions being made, and no explanation facility. Second, it applies knowledge that does not imply any real contribution for the annotation (i.e., the high-throughput data integration) (Cozzetto et al., 2013; Radivojac et al., 2013). Besides, it uses PSI-BLAST (Altschul et al., 1997), which according to (Radivojac et al., 2013) has been proved not to produce improvements over BLAST (Altschul et al., 1997), just consuming more run-time. Third, it is not publicly available.

Argot^2

Argot^2 (Falda et al., 2012) (Annotation Retrieval of Gene Ontology Terms) is a web-based tool [9] designed for functional annotation of nucleotides or protein sequences. It is able to deal from small datasets to large genomes. This tool basically combines the results of a BLAST (Altschul et al., 1997) against UniProtKB (Magrane and Consortium, 2011), with the outcome of HMMER (Eddy, 1998) against Pfam (Finn et al., 2014), and with GO terms extracted from UniProtKB-GOA (Camon et al., 2004) [10]. In order to provide the most accurate GO annotation, the system uses an algorithm that clusters and weights these terms based on semantic similarities, and the *e-values* obtained from the homology and domain searches.

The system has been used to annotate complete genomes (e.g., grape (Velasco et al., 2007) and apple (Velasco et al., 2010)), and has proven to generate reliable annotations. Nevertheless, it is only available through a web-server and employs limited and hard-coded biological knowledge.

PANNZER

PANNZER (Protein ANNotation with Z-scoRE) (Koskinen et al., 2015) is a high-throughput functional annotator developed in Python that uses R (R Core Team, 2014) for statistics and MySQL for data storage. It combines homology search and a weighted k-nearest neighbors approach with statistical testing to generate reliable annotations.

The system performs a BLAST (Altschul et al., 1997) against UniProtKB (Magrane and Consortium, 2011). The results are filtered based on their percentage of identity, alignment coverage, sequence length, and level of informativeness (see Section 3.2.4.4 for uninformative descriptions). In parallel to the filtering, the system uses a statistical approach (i.e., non-linear weighting) to generate a non-linear taxonomic distance score. This approach works like an alternative to the orthology prediction, taking into account

[9] http://www.medcomp.medicina.unipd.it/Argot2

[10] GOA is an EBI initiative to provide high-quality GO annotations to proteins in the UniProt Knowledge Base (UniProtKB)

the evolutionary distance among organisms without calculating evolutionary trees. The outcomes created by both tasks are combined through statistics methods, generating the annotation description and the GO terms related to it.

The knowledge applied in PANNZER is mainly based on statistical regressions, being very distant from the reasoning processes carried out by an expert annotator during manual annotation. This approach reduces the traceability of the outcome and limits the system evolution through experts' contributions. Moreover, the knowledge applied is limited, since the system only uses one source database and does not apply any domain information during the inference.

Name	AI approach	Language	Tools	Input	DB	Extra knowledge	Output	Availability
FIGENIX	▪ RBES	▪ Java ▪ PROLOG ▪ POSTGRE	▪ BLAST ▪ ClustalW ▪ hhmpfam ▪ others	▪ FASTA	▪ NR ▪ SwissProt ▪ Ensembl ▪ Pfam	▪ Orthology	▪ Functional annotation with GO	▪ Web server
EAP	▪ RBES	▪ Perl ▪ MySQL ▪ *ad-hoc* inference engine	▪ BLAST ▪ RepeatMasker ▪ GeneScan ▪ InterProScan ▪ TMHMM ▪ ncoil ▪ Seg ▪ Signal peptide	▪ Whole genome	▪ Swall[*] ▪ Unigene ▪ EMBL ▪ InterPro	▪ Transmembrane ▪ Coiled coils ▪ Signal peptide ▪ Low-complexity	▪ Functional annotation with GO	▪ Stand-alone
BioMediator	▪ RBES	▪ Java ▪ Jess	▪ PRS-BLAST ▪ BLAST ▪ Protégé [**]	▪ Any sequence	▪ CDD ▪ NR ▪ Pfam ▪ PROSITE ▪ ProDom	—	▪ Functional annotation	▪ Unknown
GeneWeaver	▪ MAS	▪ Java	▪ BLAST ▪ FASTA ▪ InterProScan ▪ ClustalW ▪ PhD ▪ PSIPRED ▪ MEMSAT ▪ Threader ▪ GenThreader	▪ Any sequence	▪ PDB ▪ PIR ▪ SwissProt ▪ TrEMBL	▪ Secondary structure ▪ Membrane topology ▪ Fold recognition	▪ Functional annotation	▪ Unknown
BioMas	▪ MAS ▪ RBES	▪ Java	▪ BLAST ▪ InterProScan ▪ PSort	▪ Any sequence	▪ ProDom ▪ SwissProt ▪ PROSITE ▪ PSort ▪ GenBank	▪ Sub-cellular location ▪ Transmembrane domains ▪ Cell signaling	▪ Functional annotation with GO	▪ Unknown

EDITtoTrEMBL	▪ MAS ▪ RBES	▪ Java ▪ WFSX	▪ TMHMM ▪ NNPSL	▪ FASTA ▪ Any sequence	▪ ENZYME ▪ PROSITE ▪ Pfam ▪ PRINTS	▪ Sub-cellular location ▪ Transmembrane protein prediction	▪ Functional annotation with GO	▪ Unknown
BASys	▪ NA^{+}	▪ Perl ▪ MySQL ▪ *ad-hoc* inference engine	▪ BLAST ▪ RepeatMasker ▪ GeneScan ▪ InterProScan ▪ TMHMM ▪ ncoil ▪ Seg ▪ Signal peptide	▪ Bacterial sequences	▪ UniProt ▪ CyberCell ▪ Bacterial NR ▪ PDB ▪ COG ▪ Pfam ▪ PROSITE	▪ Paralogs and orthologs ▪ Signal peptide ▪ Transmembrane regions ▪ Secondary structure ▪ 3D structure ▪ Enzyme identification ▪ KEGG information ▪ Sub-cellular localization ▪ Molecular weight ▪ Isoelectric point ▪ Operon Structure	▪ Detailed functional annotation with GO and extra information	▪ Web server
Blast2GO	▪ NA^{+}	▪ Java	▪ BLAST ▪ InterProScan	▪ FASTA ▪ Any sequence	▪ NR ▪ Custom databases [++] ▪ InterPro	▪ Enzyme identification ▪ KEGG information	▪ Functional annotation with GO	▪ Web server ▪ Stand-alone
FastAnnotator	▪ NA^{+}	▪ Python ▪ Perl	▪ LAST ▪ Blast2GO ▪ PRIAM ▪ RPS-BLAST	▪ FASTA transcripts	▪ NR ▪ Pfam	▪ Enzyme identification	▪ Functional annotation with GO	▪ Web server
AutoFACT	▪ NA^{+}	▪ Perl	▪ BLAST ▪ RPS-BLAST	▪ FASTA ▪ Any sequence	▪ rRNA ▪ Custom database	▪ KEGG pathways ▪ Enzyme identification	▪ Functional annotation with GO	▪ Web server ▪ Stand-alone
Jones-UCL	▪ NA^{+}	▪ Unkown	▪ PSI-BLAST ▪ FFPRED ▪ FunctionSpace ▪ Naive Bayes ▪ Network propagation	▪ FASTA ▪ Any sequence	▪ UniRef90 ▪ SwissProt ▪ eggNOG	▪ Gene expression ▪ Orthology ▪ Protein-protein interation	▪ GO terms	▪ Unknown

Argot[2]	▪ NA[+]	▪ Unkown	▪ BLAST ▪ HMMER ▪ GO clustering and weighting ▪ Semantic similarity	▪ FASTA ▪ Any sequence	▪ UniProtKB ▪ Pfam ▪ UniProtKB-GOA	—	▪ GO terms	▪ Web server
PANNZER	▪ NA[+]	▪ Python ▪ R ▪ MySQL	▪ BLAST ▪ Weighted k-nearest neighbors	▪ FASTA ▪ Any sequence	▪ UniProtKB	—	▪ Functional annotation with GO	▪ Web server ▪ Stand-alone

Table 3.4: Systems developed for functional annotation.

[*] Non-Redundant Protein Sequence Database incorporating SwissProt, TrEMBL and TrEMBL's weekly updates.
[**] Ontology editor from the Stanford Center for Biomedical Informatics Research. http://protege.stanford.edu.
[+] Not applicable.
[++] Only stand-alone versions.

3.3. Analysis of the state-of-art

Many supporting tools and systems to predict functional annotation have been developed in the last decades. The first ones are often stand-alone programs implemented to deal with a specific part of the annotation problem, not being able to communicate directly with others or produce final and reliable annotations. The systems, on the other hand, try to combine different tools in order to overcome communication issues and to produce accurate annotations. Section 3.1 lists some of the most popular tools focusing on their purpose (e.g., homology, domain search, and orthology prediction), while Section 3.2 describes a small subset of functional annotation systems that integrate some of these tools and other ones as well. These systems present significant differences across two orthogonal dimensions: the biological aspect, which comprises the knowledge they take into consideration, the algorithms they encapsulate, and their goals for annotation (see Section 3.3.1); and the development perspective, which involves computational aspects such as the paradigm and methodology adopted to build them, and how they are made available (see Section 3.3.2).

3.3.1. Biological aspects: knowledge, tools, and goals

Currently, the functional annotation problem has been just partially solved due to its complexity and the broad knowledge it requires. This knowledge involves so many different areas of expertise (e.g., Biology, Biochemistry, and Bioinformatics) that none of the system developed so far is able to incorporate all of it. Besides their knowledge differences, the type of input and output accepted and produced by them vary, together with the tools and algorithms they use.

In regard to the applicable knowledge, the systems, in general, at least consider the homology relationships, and the domains (and families) that compose the protein, together with GO terms associated to them (Bryson et al., 2001; Cadag et al., 2007; Chen et al., 2012; Decker et al., 2002; Domselaar et al., 2005; Falda et al., 2012; Götz et al., 2011; Gouret et al., 2005; Koski et al., 2005; Möller et al., 1999; Potter et al., 2004). Nevertheless, there are systems, like EDITtoTrEMBL (Möller et al., 1999) (see Section 3.2.3.2), that adopt a more domain-centered approach, not employing homology knowledge, or like PANNZER (Koskinen et al., 2015) (see Section 3.2.5.2), which bases predictions just on homology. Aiming to increase the annotation reliability, some systems plunge into deeper knowledge, adding expertises on: orthology (Cozzetto et al., 2013; Domselaar et al., 2005; Gouret et al., 2005), transmembrane topology prediction (Bryson et al., 2001; Decker et al., 2002; Domselaar et al., 2005; Möller et al., 1999; Potter et al., 2004), secondary structure (Bryson et al., 2001; Domselaar et al., 2005; Potter et al., 2004), signal peptide (Domselaar et al., 2005; Potter et al., 2004), 3D structure (Bryson et al., 2001; Domselaar et al., 2005), sub-cellular location (Decker et al., 2002; Domselaar et al., 2005; Möller et al., 1999), cell signaling (Decker et al., 2002), enzyme reactions and pathways (Chen et al., 2012; Götz et al., 2011; Koski et al., 2005), gene expression and protein-protein interaction (Cozzetto et al., 2013), and so on.

Even though there are many data sources available, systems usually search on a small and fixed number of databases. FastAnnotator (Chen et al., 2012) (see Section 3.2.4.3), for instance, just uses NR and Pfam (Finn et al., 2014), missing potential insights from other sources. Argot^2 (Falda et al., 2012) (see Section 3.2.5.2) also falls into the same drawback, just searching on UniProtKB (Magrane and Consortium, 2011) and Pfam. PANNZER (Koskinen et al., 2015) does not even perform domain recognition, relying only on homology information from UniProtKB (Magrane and Consortium, 2011).

Applying a broad knowledge seems to be a good strategy to make better predictions. However, it is important to take into consideration the relevance of the expertise used. Sometimes more knowledge is just translated into more running-time and not a better inference, like in Jones-UCL (Cozzetto et al., 2013) (see Section 3.2.5.2) with its high-throughput data integration (Radivojac et al., 2013).

Being able to choose the databases the system uses (e.g., using NR bacterial instead of NR when annotating bacterias) can be very advantageous. It allows more problem-oriented searches and annotations, and might decrease the processing time. However, only a few systems are flexible enough to deal with the usage of chosen external databases (Götz et al., 2011; Koski et al., 2005).

When it comes to the tools the systems support, they do not seem to be very flexible either. The tools the systems integrate are directly connected to the knowledge they comprise. As the knowledge of the majority of systems tends to revolve around homology and domain searches, the systems in general encapsulate tools for that purposes. BLAST (Altschul et al., 1997) is the most common homology tool, being used for this end by 11 of the 13 systems (Bryson et al., 2001; Cadag et al., 2007; Cozzetto et al., 2013; Decker et al., 2002; Domselaar et al., 2005; Falda et al., 2012; Götz et al., 2011; Gouret et al., 2005; Koski et al., 2005; Koskinen et al., 2015; Möller et al., 1999; Potter et al., 2004) presented in Section 3.1. Nevertheless, systems like FastAnnotator (Chen et al., 2012) use less popular homology tools like LAST (Sheetlin et al., 2005). In domain identification, there is no unanimity regarding tools. Some systems (Bryson et al., 2001; Decker et al., 2002; Domselaar et al., 2005; Götz et al., 2011; Potter et al., 2004) use InterProScan (Jones et al., 2014), and others (Cadag et al., 2007; Chen et al., 2012; Koski et al., 2005) RPS-BLAST (Altschul et al., 1997). In any case, systems usually stick to one tool, instead of being able to perform with different ones and gather more information, since these tools work with distinct databases.

The algorithms employed in the inference stage are also crucial for the prediction accuracy. System like Jones-UCL, Argot^2, and PANNZER apply statistical methods and hierarchical clustering to generate accurate outputs. On the other hand, other systems employ structured expert knowledge (Cadag et al., 2007; Gouret et al., 2005) or just some basic heuristics (Chen et al., 2012; Domselaar et al., 2005; Götz et al., 2011; Koski et al., 2005). When developing a system, one should consider how the knowledge is applied to propose the solution. Very complex and obscure calculations tend to be less intuitive to the user, hindering the system usage and evolution. On the other hand, rules and heuristics represent the knowledge in a more human-like way and render the decisions made by the system easier to understand. KBSs have a great advantage in this aspect,

since they have explanatory facilities to report the reasoning process.

Another important aspect to consider is the target of the system, which can limit its usage. Some systems, like EAP (Potter et al., 2004) (see Section 3.2.1.2), were developed to annotate complete genomes, not being able to deal with genomic sequences outside this context. Others focus on specific biological domains, such as bacterial (Domselaar et al., 2005) and transcript (Chen et al., 2012) annotation.

The outcome produced by the system is also a key feature to take into account, specially regarding two aspects: the knowledge it comprises and its format. Since there is no standard for annotation results, systems present their outcome in different ways. In general, the output comprises a description of the annotation, the domains and patterns identified, and the GO terms associated to both annotation and domains. However, some systems (Cozzetto et al., 2013; Falda et al., 2012) just focus on producing a list of GO terms, instead of enriching the annotation with extra information. The outcome format is also very important, and it should be easy to read and process. Web-based systems usually produce very visual outputs (Domselaar et al., 2005; Götz et al., 2011), which can be fairly illustrative and intuitive for the user. Nevertheless, this kind of output just works for small sequence datasets, and it is more oriented to the user, being difficult to process computationally. Moreover, the majority of systems just supply a single output format, and sometimes it does not even follow any Bioinformatics standards. Just a few systems, like FastAnnotator (Chen et al., 2012), try to overcome this hurdle offering different outcome formats (e.g., HTML, GFF (see Appendix A.3.1), and plain text).

3.3.2. Development approach

There are multiple choices to make regarding system development. Among them appear the style of programming„ computational paradigm, the model used to lead the development, and how the system is made available.

The style of programming refers to the structure and elements of the program and the interpretation of their execution. There are two main models: imperative and declarative. The imperative model focuses on how a task is carried out, describing each step of the execution. The declarative model is result-oriented. It tells the machine what should happen without specifying how, which is responsibility of some engine available.

The imperative model seems to be more commonly applied when developing annotation systems. Java, Perl, and Python are probably the most used imperative languages to implement annotators. In fact, the three are widely employed in Bioinformatics programming, having special libraries and modules (e.g., BioJava (Holland et al., 2008), BioPerl (Stajich et al., 2002), and BioPython (Cock et al., 2009)) to process and analyze this kind of data. FIGENIX (Gouret et al., 2005) (see Section 3.2.1.1), BioMediator (Cadag et al., 2007) (see Section 3.2.1.3), GeneWeaver (Bryson et al., 2001) (see Section 3.2.2.1), BioMAS (Decker et al., 2002) (see Section 3.2.3.1), EDITtoTrEMBL (Möller et al., 1999), and Blast2GO (Götz et al., 2011) are examples of annotation systems implemented in Java. EAP (Potter et al., 2004) (see Section 3.2.1.2), BASys (Domselaar et al., 2005) (see Section 3.2.4.1), and AutoFACT (Koski et al., 2005) (see Section 3.2.4.4) were developed

in Perl, while FastAnnotator (Chen et al., 2012) combines Perl and Python. Conversely, systems that use declarative paradigms are less popular. Just a few systems, like FIGENIX (Gouret et al., 2005) (with PROLOG) and (Cadag et al., 2007) (with Jess) have been developed.

The choice of the programming style involves some important considerations. Declarative approaches focus on the logics without specifying the control, thereby they are usually regarded as closer to the expert than the imperative ones. Moreover, the simplified way to represent the knowledge makes declarative languages easier to examine and understand. On the other hand, imperative approaches let programmers fine-tuning manually the control of the system, so they can be more computationally efficient. Instead of choosing just one paradigm, some of the annotation systems presented previously (Cadag et al., 2007; Gouret et al., 2005) try to combine their strengths.

There are also different declarative approaches. Choosing between them depends on the type of knowledge and inference used in the studied domain (Brachman and Levesque, 2004). For instance, logic programming languages, such as PROLOG, are known to deal well with deductive processes (Gouret et al., 2005). On the other hand, rule-based approaches, like Jess and Drools (The JBoss Drools Team, 2012), are more problem-solving oriented, being appropriate to express heuristics.

Besides the domain features, other factors should also be considered when selecting a language. For example, the existence of well-established frameworks and strong supporting communities that guarantee a long-term support for these languages. This support is specially beneficial for the development, maintenance, and extension of the system. PROLOG, Jess, and Drools are examples of well-supported declarative languages. Even though widely-used languages provide evident development benefits, some systems prefer to rely on in-house solutions (Potter et al., 2004) or less popular languages, like WFSX (Möller et al., 1999). Decisions like these might have a negative impact in the system development and evolution.

The way the inference is carried out should also be considered. Clear and intuitive reasoning processes are easier to understand, test, and explain. Besides, they can contribute to the system maintenance and evolution. Although some inference approaches, like machine learning, can play a positive role in the annotation problem, their processes can be difficult to interpret (Radivojac et al., 2013). The same can happen with systems that apply less transparent or intuitive prediction processes (Cozzetto et al., 2013; Falda et al., 2012; Koskinen et al., 2015).

Rules are known to be a natural way of representing heuristic knowledge. They promote a better understanding of the system reasoning process and contribute to its evolution. Some of the systems presented in Section 3.2 adopt a rule-based approach. Nevertheless, their rules are mainly related to data management and to control interactions among tools and their execution (Decker et al., 2002; Möller et al., 1999; Potter et al., 2004), not comprising relevant biological knowledge.

Taking into account the characteristics of the domain and its limitations can bring benefits to the system development as well. Approaches like MASs have proven to be able to deal with some particularities (e.g., heterogeneous data, distributed systems, and

constant evolution of resources) of the biological domain (Merelli et al., 2007). Nonetheless, just a few systems (Bryson et al., 2001; Decker et al., 2002; Möller et al., 1999) take advantage of this approach.

Another important point is the support of the development process itself. The use of Software and Knowledge Engineering methodologies aids the development, maintenance, and evolution of complex systems. Besides, it contributes to the process repeatability and the analysis of its tasks and outcomes. This can promote the involvement of experts in the domain and their collaboration with engineers. Literature contains many examples of well-proved methodologies applicable to different aspects of the development of annotation systems. For instance, CommonKADS (Schreiber et al., 2000) (see Appendix B.1) to elicit and model knowledge, MAS-CommonKADS (Iglesias et al., 1997) to build MASs with a an important component of knowledge management, and INGENIAS (Pavón et al., 2005) (see Appendix B.3) for MASs that work in distributed environments with heterogeneous resources. Although using these methods is recognized as a good practice, the systems studied do not seem to take into consideration (at least not explicitly documented) this kind of methodologies in their development.

The way systems are made available is also an important point to keep in mind. Web-based systems seem to be very attractive, specially for biologists. Nevertheless, they present several limitations, such as non-customizable type, version, and parametrization of the involved software and the data sources considered, and the tasks running-time (Andrade et al., 1999), not to mention possible security issues. Besides, web-based systems tend to focus on the visual aspects of the output (Domselaar et al., 2005; Götz et al., 2011), making them more illustrative and user-friendly, but less practical to be processed automatically.

3.3.3. Conclusions

An analysis of the existing solutions for the annotation problem helps to understand the requirements and available resources, as well as the limitations that should be addressed. This section has provided this information related to the knowledge to consider, the features to include, and the development approach.

An analysis of the biological aspects shows the basic knowledge to consider (e.g., homology, domain identification, and GO terms), the knowledge that can add more accuracy to the process (e.g., orthology), and even that which is irrelevant. It also points out the most useful tools (e.g., BLAST (Altschul et al., 1997), InterProScan (Jones et al., 2014), and RPS-BLAST (Altschul et al., 1997)) and data sources (e.g., NR, UniProtKB (Magrane and Consortium, 2011), InterPro (Hunter et al., 2012), CDD (Marchler-Bauer et al., 2013), and GO (Ashburner et al., 2000)).

This study can also highlight strengths and weaknesses one should take into account when designing an alternative solution. First, flexibility regarding tools and data sources is needed, since it can produce more clues when annotating, and enables a more problem-oriented approach. Second, the process carried out to infer the annotation has an important role. Methods that apply knowledge in a way closer to the human cognition

(e.g., rules and heuristics) can be followed easier, which promotes system understanding, testing, maintenance, usage, and evolution. Third, versatility in the system goals for annotation is also beneficial, overcoming limitations in the data type of inputs. Fourth, the outcome of the system should comprise at least basic information (e.g., annotation description, domains and patterns, and GO terms) and be understandable for both human and systems. Fifth, producing outcomes in different standard formats (e.g., GFF (see Appendix A.3.1), GenBank (see Appendix A.1.2), and EMBL (see Appendix A.1.1) formats) can add value to the system.

Analyzing the development aspects of the annotation systems built so far can also produce good requirements for proposing a new one. The paradigm that seems to suit better this problem is a hybrid declarative and imperative approach, being the first used for the inference and the other for controlling the system.

The right choice of languages to implement the system is also very important. Well-established and supported languages should be preferred, since they aid the development, maintenance, and evolution of the system. The language is also closely related to the inference process. Transparent and straightforward inference processes help experts to understand and test the system, contributing to its development and extension. Rules represent the heuristic and procedural knowledge in a human-like way, and according to (Xavier et al., 2013), they fit well the annotation problem, which is a classification task.

Another aspect to consider is the domain features. The current one has features of data heterogeneity, distribution, and quick evolution that makes it suitable for an agent-oriented approach. Systems that do not consider these features may have problems to succeed and evolve in the long term.

The system development process is also crucial for that evolution. State-of-the-art methodologies are available to obtain knowledge and model it, and to design and adapt the system. Unfortunately, the use of such methodologies does not seem to be widely spread in the studied domain.

Finally, the way the system is made available plays a significant role as well. Web-based systems are usually user friendly, but they have many limitations regarding automated processing and integration with other systems.

In order to overcome the limitations discussed here, this work developed MASSA, a functional annotation system that combines a MAS with a RBES. MASSA not only intends to be able to generate precise functional annotation, but also to propose solutions for the aforementioned constraints. MASSA was developed following state-of-the-art Software and Knowledge Engineering methodologies, and incorporates some of the relevant features previously described (e.g., exclusion of uninformative targets, clear reasoning process, flexibility, well-supported languages, variety of outputs, and is stand-alone system). MASSA is described and assessed in detail in Chapter 4.

Chapter 4

MASSA: Multi-Agent System to Support functional Annotation

If knowledge can create problems,
it is not through ignorance that we can solve them.

Isaac Asimov

This chapter presents MASSA, a MAS that integrates a RBES to infer accurate functional annotations. Section 4.1 gives an overview of the system, and Section 4.2 introduces the knowledge and requirements took into consideration when developing it. Section 4.3 explains the system architecture, describing the MAS and RBES organization. Section 4.4 discusses how the system design facilitates its evolution and the knowledge management. Section 4.5 presents the system evaluation based on the CAFA experiment, and Section 4.6 summarizes the chapter.

4.1. Introduction

MASSA is a system for functional annotation designed to address several limitations of the domain appearing in the already developed systems (see Chapter 3). In particular, it considers how to deal with the different tradeoffs of the manual and automatic functional annotation approaches, and with the heterogeneity and constant change of knowledge, resources, and tools. It also has a special focus on evolution, being developed using well-supported methodologies, computational paradigms, and languages.

The amount of genomic data generated nowadays prevents the manual annotation, but the lower accuracy of the automatic approach can compromise its outcome. A potential solution is the combination of the expertise of the human annotator with the fast processing of the machine. This can be done using KBSs capable of emulating the expert reasoning at key points of the process.

The knowledge of a KBS can be described using different approaches. Rules are one of the most common (Giarratano and Riley, 1998), and are also well-suited for the annotation inference process.

Rules are easier to write and examine than code, and they are intended for procedural, factual, and heuristic knowledge (Hayes-Roth, 1985), as that required in gene annotation. This makes them more understandable for biologists, which encourages their more active involvement in the management of that knowledge. Moreover, RBESs are able to provide a trace of all decisions they made to solve a problem, allowing to know which rules they applied and how. The use of rules also encourages modularity at different levels. It decouples domain knowledge (i.e., rules) from information on specific annotation requests (i.e., facts), and blocks of rules focused on different aspects. Rules and types of facts are described declaratively, which facilitates their introduction and modification, as well as the experts' participation in that process.

Knowledge Elicitation (KE) techniques facilitate the join work of engineers and experts when developing KBSs. The knowledge used to build MASSA was acquired following this kind of methodologies. MASSA combines the knowledge obtained by primary research with the one acquired through structured interviews. This is complemented with the study of the available supporting tools and systems for functional annotation. It also considered *community annotation* efforts (see Section 4.2.3).

Besides knowledge, this analysis also brought to light environmental conditions of the process that should be taken into account. The information used here is heterogeneous, available from many distinct locations, and constantly changing. Furthermore, the activities tools perform are repetitive, time-consuming, and resource-intensive, and can be carried out independently for each sequence. All these features make MASs a suitable approach to this problem (Decker et al., 2002; Luck and Merelli, 2005). MASs do not only facilitate to deal with the particularities of the studied domain. This approach also allows to build a modular architecture closer to the manual annotation process. This makes the system workflow more intuitive to the expert, promoting also its maintenance, reusability, and evolution.

MASSA is implemented in Java, adopting widespread technologies such as Jade (Bellifemine et al., 2001) for its agents and Drools (The JBoss Drools Team, 2012) for the RBES. This also helps with development and maintenance. The system also executes some in-house and third-part Perl scripts, a language extensively applied in Bioinformatics. MySQL takes care of the database management.

4.2. Requirements

Building any system requires a deep comprehension of the problem to be tackled and its domain. This knowledge is used to establish the requirements of a possible solution.

The research behind MASSA was fairly extensive. As said in the introduction, the knowledge to implement the system was initially extracted through primary research and structured interviews, being modeled with the CommonKADS (Schreiber et al.,

2000) methodology (see Section 5.1). This knowledge was complemented with an analysis of the available supporting tools and annotation systems (see Chapter 3). This study gave insights regarding the domain and its biological aspects, as well as the system development.

An analysis of the *community annotation* efforts (see Section 5.4) was also carried out through the Social Science framework of the Activity Theory (AT) (Leontiev, 1978) (see Appendix B.2), It aimed to have a better understanding of the domain, and to enrich the set of requirements with information on this quite novel type of effort.

The next subsections present the knowledge and requirements obtained in these studies.

4.2.1. Knowledge elicitation and modeling

Before developing any KBS is crucial to acquire the knowledge to be implemented by it. This knowledge can be obtained from different kinds of sources related to the domain. Primary research can give a good overview of the studied problem and its domain. However, one of the most important sources of expertise are the experts (Xavier et al., 2013). Experts detain knowledge they learned empirically, that often has not been previously cataloged. Nevertheless, this knowledge is organized and stored in a complex cognitive way, requiring the use of some KE techniques to acquire it properly.

In this work, a research was conducted to understand how different experts solve the annotation problem. This investigation was carried out through the KE technique *structured interview*, which is fairly popular since it is a low cost method that does not require any special training. This technique suits perfectly the knowledge to be acquired (Xavier et al., 2013) and it is able to produce structured information, which is easier to translate to rules as required for MASSA.

The experts interviewed in this work were very familiar with the domain, having two types of profile: *academic* and *practitioner* (Xavier et al., 2013). The interviews discovered important information. First, each expert uses a different pipeline to functionally annotate sequences. Therefore, the annotation system has to be as flexible as possible in order to be able to meet the expert needs. Second, different workflows can lead to the same results. Thereby, if there is a solution that is pointed out by different pipelines, it might be more likely to be precise. Third, BLAST (Altschul et al., 1997) is used by all experts, regardless their pipeline. Thus, the system has to support this tool. Fourth, there are some attributes of programs (e.g., the *e-value* and *bit score*), that have to be considered when making decisions. Fifth, programs consult different databases and those are constantly being updated. The system should be flexible enough to deal with different data sources and to keep its local copies updated.

After acquiring the knowledge, it has to be modeled, so it can be translated into the system. CommonKADS (Schreiber et al., 2000) (see Appendix B.1) is a flexible methodology to model the knowledge regardless its context. It also helps to identify the more suitable strategy to address the problem. CommonKADS (Schreiber et al., 2000) and its application are briefly described in Appendix 4.2.1.

CommonKADS (Schreiber et al., 2000) has a *Knowledge Model* that helps to structure knowledge-intensive information-processing tasks. This model comprises three knowledge categories: *Task Knowledge*, *Inference Knowledge*, and *Domain Knowledge*.

The *Task Knowledge* is essential to the system design. It describes the goals to be achieved through the expertise and the strategies to be traced in order to accomplish them. CommonKADS (Schreiber et al., 2000) has some templates that facilitate the task modeling (Schreiber et al., 2000). According to (Xavier et al., 2013), based on these templates and the features of the functional annotation task, it is possible to frame it as an analytical task (i.e., it makes part of a system that is not completely characterized) of classification.

Classification tasks aim to find the association between the features of an object and a class. In the case of the functional annotation problem, the goal is to assign the protein " class" that best describes a given sequence. The *Task Model* also pointed out that the *default method* to solve the problem is a data-driven approach (i.e., the object's initial features are used to generate a set of candidate solutions). That happens because, in the annotation problem, a set of candidates (i.e., target sequences) is obtained from the initial data (i.e., query sequence) through a comparison method.

4.2.2. State-of-the-art analysis

In order to complement the domain analysis and the KE (see Section 4.2.1), a study of the tools and systems to support the functional annotation was carried out (see Chapter 3). This analysis discovered some important system requirements related to two aspects: biological knowledge and system development.

Regarding the biological knowledge it showed that most of systems take into consideration the homolog relationships, domain and family identification, and GO terms association. Besides this basic knowledge, some further analysis, like orthology, can add more accuracy to the prediction.

It also indicated the most common databases (e.g., NR, UniProtKB (Magrane and Consortium, 2011), CDD (Marchler-Bauer et al., 2013), and GO (Ashburner et al., 2000)) and the most used supporting tools (e.g., BLAST (Altschul et al., 1997), InterProScan (Jones et al., 2014), and RPS-BLAST (Altschul et al., 1997)), confirming the popularity of BLAST (Altschul et al., 1997) mentioned in Section 4.2.1. This analysis also showed that systems (like experts in Section 4.2.1), tend to follow different workflows and use distinct databases. This suggests that the system should be flexible and customizable regarding both tools and data sources.

Another conclusion taken from this investigation was that those systems that apply a reasoning process closer to the human cognition, are easier to understand, test, and use by users. Based on that, structuring the key inference knowledge as rules seems to be advantageous for users' adoption and the system lifecycle.

The importance of the system versatility when it comes to the type of input was also highlighted. A system should be able to accept any sequence to be annotated, regardless its context and specie.

One more learned lesson was about the annotation output. Although there is no formal convention about its format and the information it should hold, the annotation result should present some basic information such as its description, the domains and patterns identified, and the GO terms associated with them. Besides, it should adopt a format that can be understood by human and easily processed by machines. Text-based formats like GFF (see Appendix A.3.1), GenBank (see Appendix A.1.2), and EMBL (see Appendix A.1.1) are a good choice. Moreover, providing more than one format can also be beneficial.

As for the development aspects, the first conclusion was that using a declarative approach for the inference and an imperative one for controlling the system seems to be very suitable for the problem. Moreover, the languages used to implement the system should be well-supported, as this facilitates the system development, maintenance, and evolution.

The investigation also showed the importance of selecting a proper inference process according to the considered knowledge, which should be as clear and intuitive as possible for experts and engineers. This reinforces the idea of using rules, and allied with the necessity of accurate results, it points out to using a RBES.

Taking into account the domain particularities was another point to consider. A domain like the one studied should be tackled using an approach that is able to deal with data heterogeneity, distribution, and evolution. The multi-agent approach is able not only to handle such singularities, but also to deal with repetitive and time-consuming activities and knowledge management.

Based on the three last conclusions (i.e., the usage of a hybrid paradigm with RBES and MAS), it seems a good idea to choose Java to implement the system. This widespread imperative language can run in different platforms and has support for agents, in our case with Jade (Bellifemine et al., 2001), and for the inference process, here with Drools (The JBoss Drools Team, 2012), both well-established and supported frameworks.

The research also showed that the way the system is made available should be taken into consideration. Web-based systems have many limitations (e.g., tools they execute, their parameters, the databases they use, or the size of the input). Thereby, stand-alone systems should be preferred.

Although this study could not find any example of application of Software Engineering methodologies to develop these systems, it is well-known that the usage of such practices is essential for a proper system development and evolution. Following this idea, two state-of-the-art methodologies were chosen to guide development: CommonKADS (Schreiber et al., 2000) for the RBES and INGENIAS (Pavón et al., 2005) for the MAS.

4.2.3. Community annotation analysis

Community annotation (Mazumder et al., 2009) is an approach intended to distribute the effort of annotating genomics data among the users of the related databases. Its motivation lies on the lack of enough expert curators to make and validate annotations, which is producing a growing number of highly automated annotations of lower quality

(Jones et al., 2007; Schnoes et al., 2009).

A social analysis of the *community annotation* guided by the AT (Leontiev, 1978) was carried out as described in (Xavier and Fuentes-Fernández, 2015). This analysis was based on conflicts (i.e., *contradictions*) existent in some practices and resources of the traditional annotation processes. It aimed not only to find potential solutions for such *contradictions*, but also to deepen the comprehension of the domain and to collect more requirements for the system. The analysis was centered in two main aspects: the annotation community and it resources.

The study of the community indicated that the participants in the community annotation effort lack of incentives to collaborate in it. One possible solution, that has been successfully applied in other domains, is creating a reward mechanism that could give them credit for their efforts. This kind of mechanism can be easily integrated to an annotation system, but this should also support multiple users. This fact reinforces the choice for the solution of a multi-agent approach, since this kind of system can deal very well with this requirement.

The other aspect was related to how the knowledge is built and shared. Collaborative initiatives like the one studied are processes of co-construction of knowledge. Therefore they require support for the externalization of the knowledge together with the processes that produce it and information on them. The tools currently available lack of such support, focusing on the individual annotation. Besides, they give partial and scattered information on the join process. This issue can be solved with a combination of requirements. First, a trace of the reasoning process should be provided. This reinforces the decision of using RBESs, which among other advantages comprise an explanation facility (see Section 3.2.1). Second, a user monitoring functionality can be added to the system in order to observe the users activities and obtain insights based on their behavior.

4.3. Architecture

MASSA architecture was designed and implemented taking into account the set of requirements previously discussed (see Section 4.2). They aim to overcome domain constraints, improve the system accuracy and flexibility, and facilitate its maintenance and evolution.

MASSA is a MAS integrating a RBES that generates automatically accurate functional annotations. It comprehends two main subsystems (see Figure 4.1): *MASSAPipe* and *MASSAInference*. The former is responsible for executing the annotation pipeline and gathering the outcomes into a database, while the latter aims to infer the best annotation based on the data previously acquired and the set of rules in the KB of the RBES. Thanks to their modularity, these subsystems can work together or independently. The overall MASSA system also includes some additional agents that provide shared services to manage the user interface and coordinate the works.

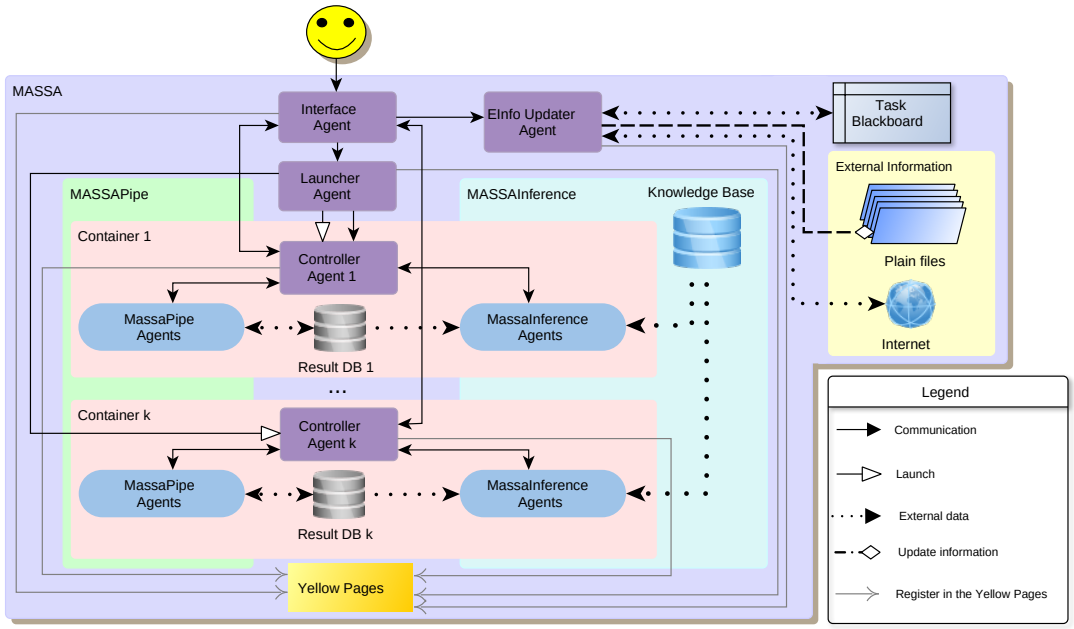


Figure 4.1: MASSA schema for *k projects*.

MASSA can handle multiple requests from different users simultaneously. An annotation request is linked to a *project*, which has assigned its own *environment*. The *environment* includes a *container* for the group of agents working in it, and a database for intermediate results, the Result Database (RDB). Each request is divided in smaller chunks, called *tasks*, in order to parallelize as much as possible its processing. A *task* can involve the execution of one or more tools, scripts, and complementary jobs as *subtasks*. For example, a RPS-BLAST (Altschul et al., 1997) *task* comprises three *subtasks*: *exec_fastatranslate*, *exec_rpsblast*, and *blast2GFF*.

MASSA is mainly implemented using the Jade (Bellifemine et al., 2001) framework. This provides a distributed, fault tolerant, container architecture and platform for MASs. In the case of *environments*, their *containers* are implemented using Jade *containers*. These can be actually distributed over the network, allowing MASSA to take advantage of all the available resources. Jade also facilitates expressing workflows (here tasks in a pipeline) as Finite State Machines (FSMs), as it directly supports these.

Next sections describe in more details the previous elements and relevant implementation aspects. They include: shared components (see Section 4.3.1), MASSAPipe (see Section 4.3.2), and MASSAInference (see Section 4.3.3). Figures 4.2 and 4.3 depicts the task performed by these components and their workflow in INGENIAS terms.

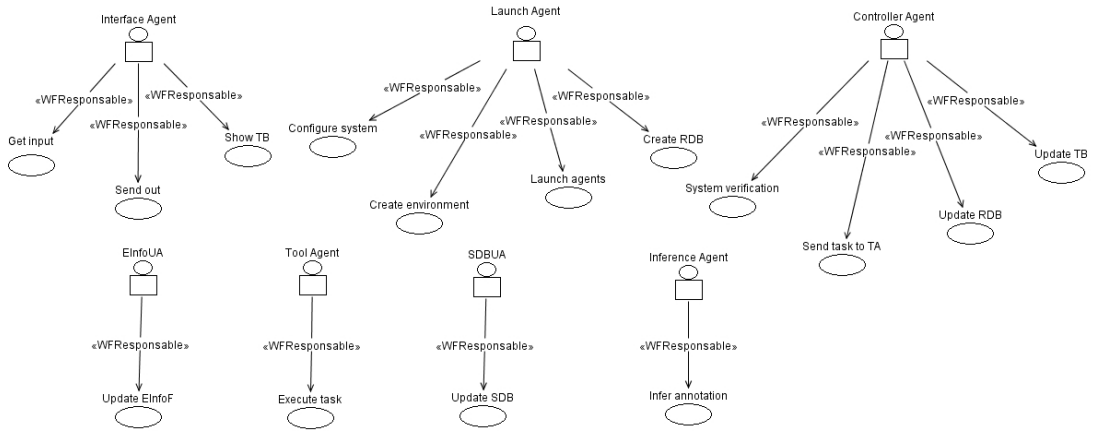


Figure 4.2: Agent diagram for MASSA with INGENIAS.

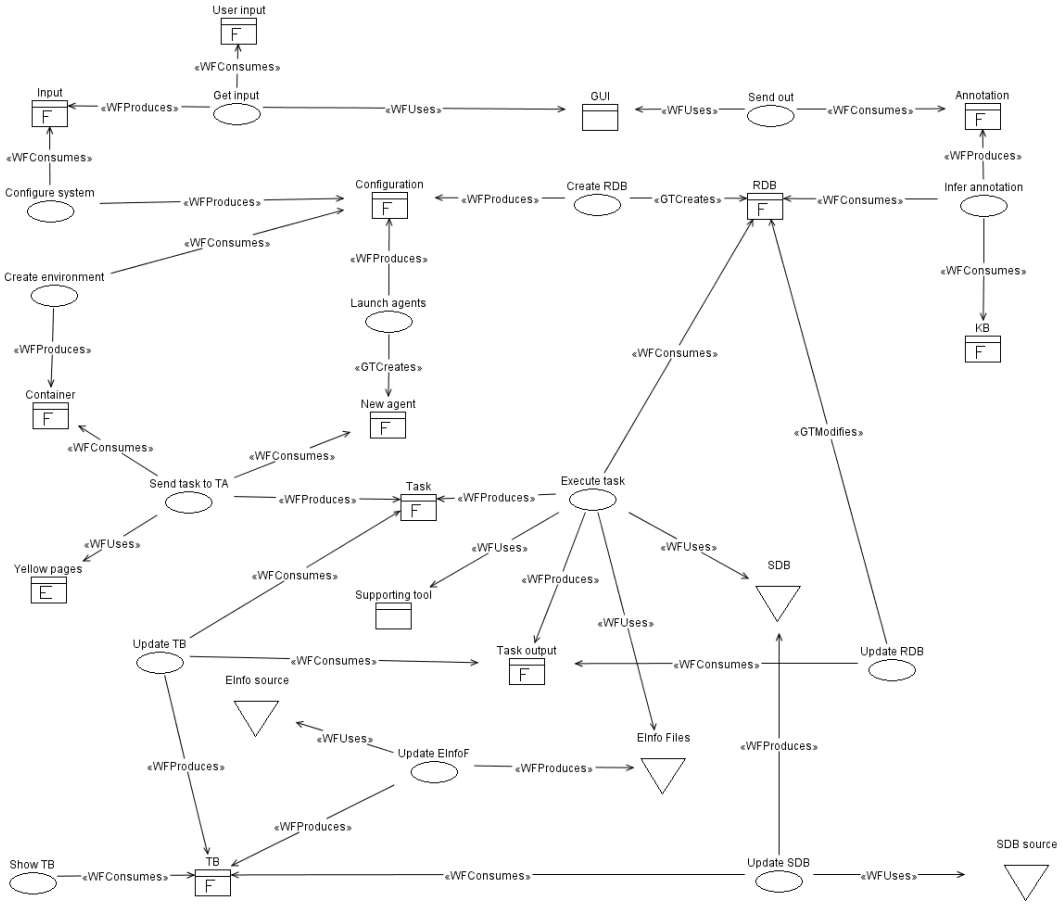


Figure 4.3: Agents' workflow for MASSA with INGENIAS.

4.3.1. Shared components

The shared components of MASSA manage the user interface, and the launching, coordination, and monitoring of *projects*. They make use of several resources that are shared with other subsystems. Next subsections describe these components (see Section 4.3.1.1) and their interactions (see Section 4.3.1.2).

4.3.1.1. Components

MASSA has three agents that are unique and initialized with the system:

- The *Interface Agent* (IA) is the bridge between the user and the system. It manages components to allow entering requests and their configuration, monitoring their solving, and examining the outcomes.
- The *Launcher Agent* (LA) creates the *environment* for each *project*, and launches all the agents required to execute the annotation pipeline and inference. This agent is also responsible for configuring the system and creating the RDBs of *projects*.
- The *External Information Updater Agent* (EInfoUA) updates the External Information Files (EInfoFs) through remote connection to their sources. Theses files mainly have information from CDD (see Section 2.5.1.3) regarding for instance *bit score* thresholds (TBSF) [1], conserved features (cddannot.dat) [2], super families members (FSLF) [3], and database versions (cdd.versions) [4].

Among the agents the LA creates, *Controller Agents* (CAs) also act across subsystems. A CA is responsible for managing the global aspects of a *project*. It implements the *mediator* pattern (Gamma et al., 1993) to coordinate the agents in MASSAPipe and MASSAInference working for the *project*, and their communications with the IA. This organization avoids agents referring to each other explicitly, which improves system flexibility. For instance, it allows adapting MASSAPipe for different pipelines, just by removing and deploying agents. The CA also performs basic verifications before starting request solving, such as checking if there is an ongoing update of needed SDBs or external files.

In order to coordinate agents dynamically, i.e., depending on those available, there is need for an agent discovering service. For instance, CAs use it to assign jobs to the proper agents. In MASSA, that service is Yellow Pages (YPs), which is a common infrastructure in agent platforms like Jade (Bellifemine et al., 2001). Agents register themselves in

[1] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/bitscore_specific_3.14.txt`. [Online: accessed 4-September-2015]

[2] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/cddannot_generic.dat.gz`. [Online: accessed 4-September-2015]

[3] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/family_superfamily_links`. [Online: accessed 4-September-2015]

[4] `ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/cdd.versions`. [Online: accessed 4-September-2015]

the YPs by providing a Service Description (SD) that accounts for the tasks they can perform. Then, other agents can ask the YPs for registered agents that provide certain services.

The system also needs to know the status of *tasks*, for instance for job monitoring and agent coordination. This information is stored into the Task Blackboard (TB) for all *projects*. It is a table where each entry contains the name of an agent, the *task* or *subtask* it is (or was) engaged in, and its status. When a job is assigned to an agent, the requester agent inserts this information into the TB with status "Started". After its completion, the requester agent updates the job status to "Finished" or "Failed" according to its result.

The last shared element to consider is the request itself. A request is described with a FASTA file (see Appendix A.2.1) and an optional Configuration File (CF) (see Appendix A.3.3.2). The CF is a plain file that provides optional settings to MASSA, e.g., programs to be executed, databases to use, number of threads, thresholds for calculations, or number of agents to carry out a certain job.

As the settings in CFs are optional, MASSA has a General CF (GCF) (see Appendix A.3.3.1) with default values to complete requests. This default configuration executes BLASTX (Altschul et al., 1997) against NR and UniProtKB (Magrane and Consortium, 2011), and RPS-BLAST (Altschul et al., 1997) against CDD (Marchler-Bauer et al., 2013), InterProScan (Jones et al., 2014), and Orthostrapper (Storm and Sonnhammer, 2002).

4.3.1.2. Agent's interactions and information exchange

The activities of MASSA start when a user introduces the FASTA file (see Appendix A.2.1) and the CF (see Appendix A.3.3.2) of a request through the graphical interface. The IA checks the correct format of the request, forwarding it to the LA if everything is right. If there is any issue with the request, the IA notifies it to the user and discards the job.

When the LA receives the request, it sets up the *environment* for this job. It sets the variable values for the job using the information in the request complemented with default values from the GCF (see Appendix A.3.3.1) when needed. After that, it starts the CA for this *environment*, and passes the information about the job to it. At this point, the LA also creates a RDB where the *task* outcomes will be stored, and launches the required agents from MASSAPipe (see Section 4.3.2).

When the CA receives the start message from the LA, it checks that all the required input files are available. Then, it processes the *tasks* to perform: it decomposes them into *subtasks* that stores in a queue, and assigns them to the proper agents from MASSAPipe. When these agents finish their *tasks*, the CA uploads the obtained information to the *project* RDB in order to allow MASSAInference agents to start their part.

The annotation process for a request can take a while (even several hours) (see Section 4.5.2), depending on the number of query sequences, the size of the SDBs to consult, and the configuration set. The IA keeps informed the user on the job status at any time. Besides, when each subsystem finishes processing a request, a notification is sent to the

user, either by e-mail or through the graphical interface.

4.3.2. MASSAPipe

MASSAPipe is the subsystem that works as an annotation pipeline, executing tools and scripts that search in SDBs to get likely candidates for the annotation. It is also responsible for storing the obtained information and updating some of the default SDBs from their sources. Next subsections describe its components (see Section 4.3.2.1) and their interactions (see Section 4.3.2.2).

4.3.2.1. Components

MASSAPipe is composed of two types of agents:

- The *Tool Agent* (TA) is the wrapper for the tools and scripts of pipelines. It is responsible for running them, gathering information about the candidates, and filtering and structuring it to be stored into the RDB. It also accesses remote databases (e.g., Entrez Protein (NCBI Resource Coordinators, 2013), UniProt (The UniProt Consortium, 2014), InterPro (Hunter et al., 2012), and CDD (Marchler-Bauer et al., 2013)) and the EInfoFs.
- The *Search Database Updater Agent* (SDBUA) updates the SDBs from their remote sources.

MASSAPipe produces as outcome for each *project* the data in its RDB and a Task Log File (TLF) (see Appendix A.3.3.3). The RDB contains information about the possible candidates for annotation. The information on a candidate includes its identification, e-value, bit score, percent of identity, coordinates where it aligns with the query sequence, and so on, as well as the methods and databases employed to obtain these.

The TLF is a tabular file that comprises the TB information related to the *project*. It includes for each *task* the command executed, its *subtasks*, the date and time they started and finished, and the agent in charge of it. TAs wrap Perl scripts that manipulate Bioinformatics-specific information, as commonly done in the field. Part of these scripts is specifically developed for MASSAPipe, and others are publicly available from third-parts, e.g., some BioPerl (Stajich et al., 2002) modules and GBrowse (Generic Model Organism Database (GMOD) Project, 2013) scripts. Their information is described conforming to common Bioinformatics formats, e.g., FASTA (see Appendix A.2.1) and GFF (see Appendix A.3.1).

4.3.2.2. Agent's interactions and information exchange

Figure 4.4 shows an example of execution in MASSAPipe. The activities start after those of the shared agents (see Section 4.3.1.2). There, the LA sets up the *project* of a request. This activity includes launching its CA and as many TAs as *tasks* appear in the configuration. Then, it passes to the CA the information on the request.

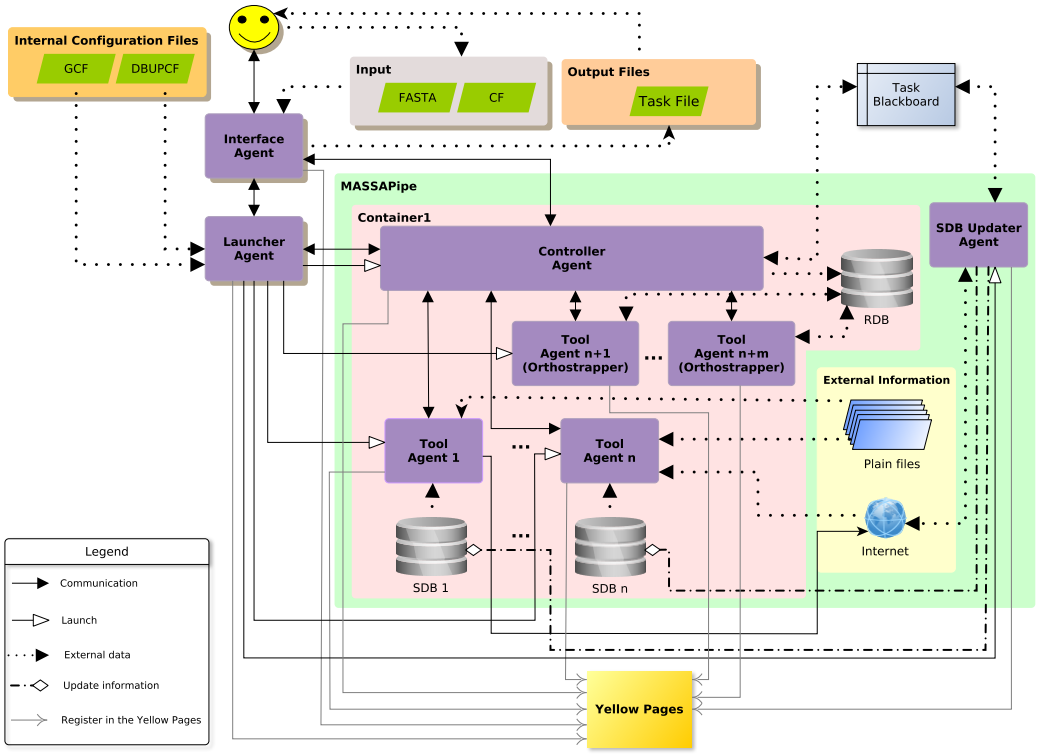


Figure 4.4: Schema of interactions in MASSAPipe for a *project* running n *tasks* and m SDBs. Although InterProScan databases have a different format than the BLAST SDBs, for the sake of conciseness all the databases are illustrated here as SDBs.

The CA validates the request, and decomposes each *task* into *subtasks* that stores in a queue. The CA allocates these *subtasks* to suitable TAs using the YPs. When a TA finishes its *subtask*, it sends to its CA a message confirming the completion together with the outcome, and waits for the next request. The last *subtask* performed by any TA consists in gathering its candidates' information in the GFF format (see Appendix A.3.1). After that, and if there are no errors, the CA informs the TA that its work is done. The TA removes itself from the YPs and finishes.

When all the *tasks* of a request are completed, the CA uploads the resulting GFF files (see Appendix A.3.1) into the RDB of the *project*. Then, it contacts the TAs in charge of Orthostrapper (Storm and Sonnhammer, 2002) execution. These TAs retrieve data from the RDB, use them to infer orthology, and update the RDB with the new information generated. Orthostrapper just work with homology data, thereby the number of TAs responsible for this task depends on the number of TAs (and sources) used to obtain this information. After completion of these *tasks*, the CA sends a message to the IA, which informs the user that this step is finished. If the system is configured to execute

only MASSAPipe, the CA terminates its *container*, removing the whole *environment* (including itself) from the platform.

As aforesaid, SDBs are updated periodically. The SDBUA checks regularly for new versions of them at their sources. If there is one, it updates the related local SDB when no other agent is using it. The SDBUA also uses the TB to block SDBs while it is performing their actual update. On the other side, the CA checks the TB to avoid launching jobs that need blocked SDBs. Currently, the SDBUA is able to update NR, NT, CDD (Marchler-Bauer et al., 2013), and UniProtKB (Magrane and Consortium, 2011). The EInfoUA works in a similar way to the SDBUA.

4.3.3. MassaInference

MASSAInference combines a MAS approach with a RBES to assign annotations to requests. Next subsections explain this subsystem in terms of its components (see Section 4.3.3.1), and in particular its KB (see Section 4.3.3.2), and their interactions (see Section 4.3.3.3).

4.3.3.1. Components

The key agent in MASSAInference is the *Inference Agent* (InfA), which encapsulates the RBES INFAES (see Section 4.3.3.2). The main responsibility of this agent is to infer the best annotation based on the information from the RDB and the rules in the KB. In order to accomplish it, this agent interacts directly with the CA and retrieves the data of candidate annotations from the RDB. The InfA can also access local data (files and a database) to obtain GO terms in order to enhance the annotation predicted.

This subsystem produces seven output files with the results of the annotation and their explanation. They are: a GFF (see Appendix A.3.1), GenBank (see Appendix A.1.2), and FASTA (see Appendix A.2.1) files containing all the query sequences with the proper target information of each annotation; a FASTA file with the sequences that were not annotated; a plain file containing the sorted Potential Candidates List (PCLF) (see Appendix A.3.3.4); a Trace File (TF) that summarizes the reasoning process for the *project*; and a TLF (see Appendix A.3.3.3).

MASSAInference integrates some third-part components. It uses BioJava (Holland et al., 2008) libraries to produce well-formatted GFF files, and its RBES is implemented with Drools (The JBoss Drools Team, 2012). As other rule engines, Drools includes an explanation facility that here describes step by step the reasoning process taken to arrive to each annotation. The TF includes this information for a *project*.

4.3.3.2. INFAES

The KB contains rules that work on the candidate's features present in the RDB. These features (see Section 5.3, Table 3) include, among others, the annotation description, *e-value*, *bit score*, and domains and families. The rules evaluate and combine the

features using thresholds and heuristics to get scores. These scores (see Section 5.3, Table 4) summarize experts knowledge on the relevance and mutual influences of attributes and their values, and point out whether a target sequence is a promising candidate or not. Examples of scores are those for *Orthology*, *Specificity*, *Domain*, and *Conserved Sites* (see Section 5.3, Table 4). The candidate that accomplishes the higher scores, and meets some criteria, is assigned as the annotation of the query sequence.

Currently there are more than sixty rules in the KB. Their complexity ranges from quite simple ones, like "*if the domains or families found in the query sequence are the same as the ones found in the target, set the Domain score to 1; otherwise, set the Domain score to the number of domains or families found in both query and target sequences (Domain or Family Intersection) divided by the total number of domains or families existent in the target sequence*", to complex ones involving multiple variables and conditions. These rules are not totally independent, as they are designed to reproduce the steps in usual expert workflows when annotating sequences. Figure 4.5 shows this basic reasoning schema.

The annotation of a sequence can follow three different paths depending on the information found in the RDB. The best case is when there is information on homolog sequences (a). This path considers all attributes (when they exist), so it is regarded as the most reliable. When that information is not available, the annotation relies only on domain similarities and the conserved sites if any (b). Nevertheless, some sequences do not match to domains either. These candidates are added to a "Not annotated" list (26).

Besides the information to select the path, rules consider other for each candidate when available. This includes: the quality of the source (2), orthology likelihood (5), domains (8, 9, and 12), and conserved sites (15).

After calculating the previous scores, the candidates are compared and sorted in a list of potential candidates (see Appendix A.3.3.4). The ordering happens at the last stage of the inference (see Figure 2 (18) and (25) from Section 5.3) and indicates the most likely and informative annotations. At this point, the existence of GO terms, the values associated with the *bit score*, *e-value*, and *percentage of identity* play an important role in the decision process. If all candidates present the same scores, the KB prioritizes the one associated to GO terms; if both candidates have (or have not) GO terms, the one with the higher *bit score* is preferred; if this information is not enough for making a decision, the *e-value* is taken into account, and the candidate with the lowest is chosen. When all the previous values are the same, the percentage of identity is considered.

4.3.3.3. Agent's interactions and information exchange

Figure 4.6 shows the interactions between agents during a MASSAInference execution. It starts with the LA launching the needed agents.

The CA of the *project* is present if MASSAPipe was executed before MASSAInference, but must be launched when MASSAInference is used as a standalone application. In both cases, information on candidate annotations must be available in the *project* RDB. The number of InfA to launch is the minimum between that specified in the CF and the number of query sequences in the RDB. After launching all the agents, the LA informs

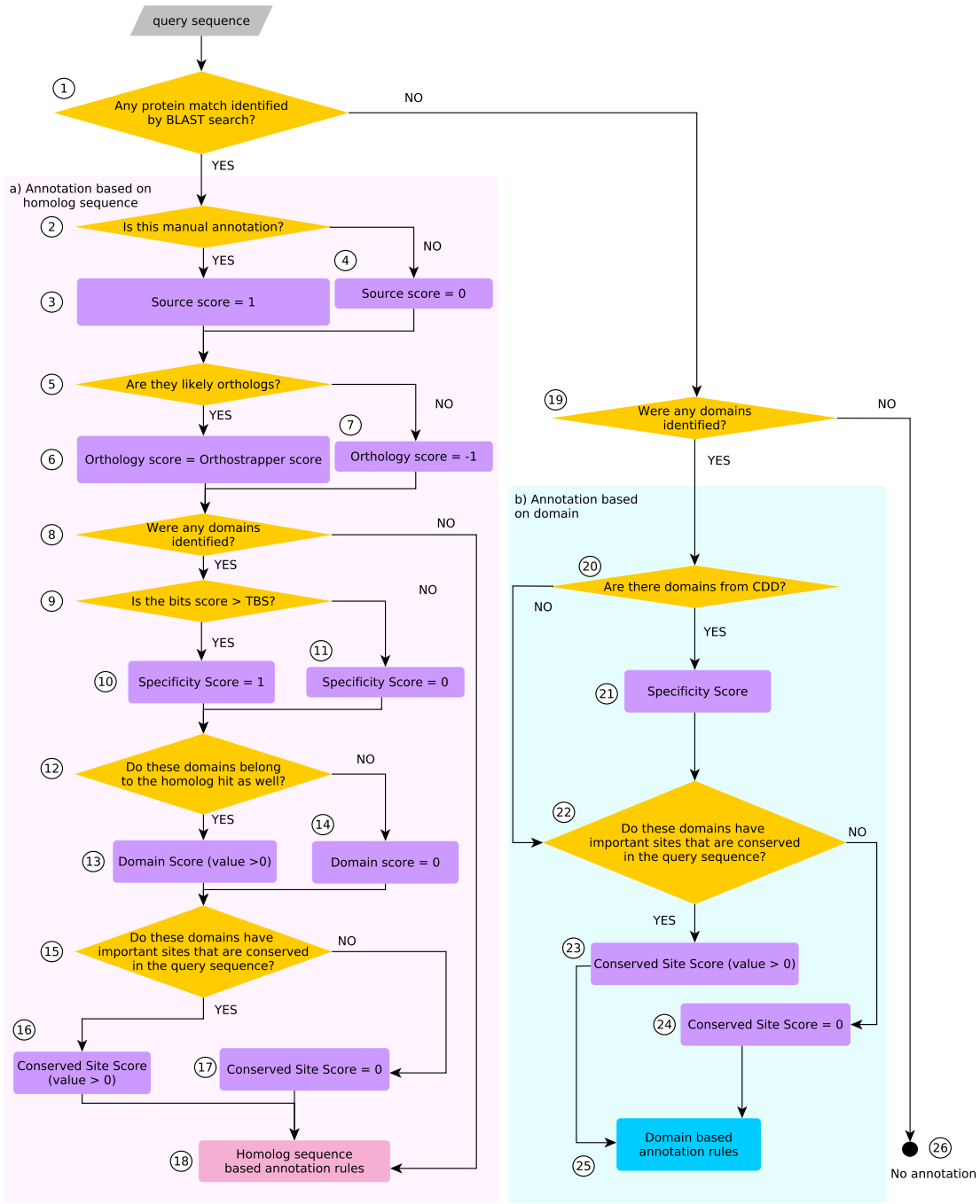


Figure 4.5: Workflow for candidate evaluation.

Source: Xavier et al. (2015)

the CA that the InfAs are ready.

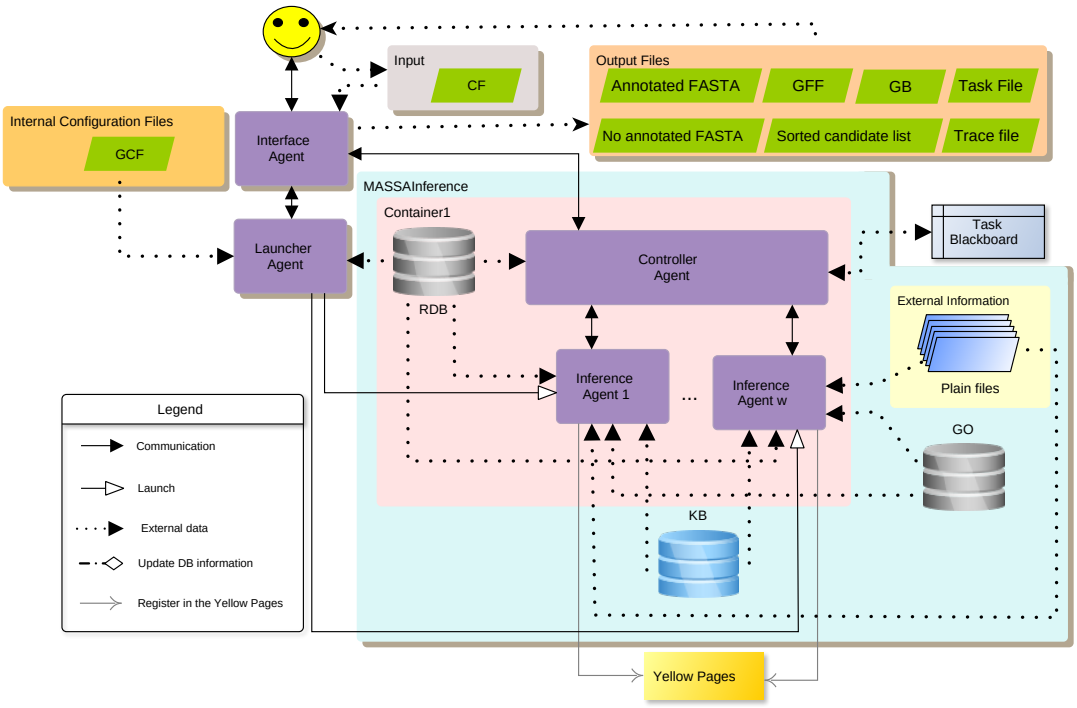


Figure 4.6: Schema of interactions in MASSAInference for a *project* with w InfAs.

The CA sends a message to each InfA notifying which sequences it is in charge of, and adds the started *tasks* to the TB. Each InfA queries the RDB to get the features of its candidates. These data are processed according to the rules in the KB, and the best candidate is assigned as the final annotation. The InfA notifies the CA when its *task* is completed. The CA, in turn, changes the status of the job to "Finished", and tells the InfA to quit. This agent deregisters itself from the YPs and leaves the *container*.

When all InfAs have finished their *tasks*, the CA gathers the information in one GFF file (see Section A.3.1), updates the TB status, informs the IA that the annotation is done, and removes its *container* (together with itself) from the platform. Then, the IA notifies the user and makes the outputs available.

4.4. System evolution and management

Bioinformatics knowledge and tools are in constant evolution. MASSA is designed focusing on adaptation and management of this evolution. MASSA uses CommonKADS (Schreiber et al., 2000) and RBESs (Giarratano and Riley, 1998) to deal with expert KE,

modeling, use, and maintenance, regarding both biological and tools aspects. It follows the agent paradigm with INGENIAS (Pavón et al., 2005) to address issues of heterogeneity of resources, tool integration, distribution, and adaptation to change. From the development point of view, only the integration of new tools requires some programming.

Tools are characterized as new types of *task* (see Section 4.3). A *task* is viewed as a FSM, being each of its states a *subtask*. Usually, the main *subtask* is the execution of the tool. Then, other *subtask* parses its output and stores the relevant information in a GFF file (see Section A.3.1), which is uploaded to the RDB. Consider the example of the InterProScan (Jones et al., 2014) tool. Its *task* can be decomposed into two *subtasks*: *exec_ interproscan* and *interproscan2GFF*. It is important to note that some *tasks* may have more steps, like *RPS-BLAST task* (see Section 4.3).

The FSM of a *task* is defined by a class that extends a Jade *FSMBehaviour*. The definition of each *subtask* implies indicating the command line that should be executed for it. This can be done using an already implemented class as template. Listing 4.1 shows the definition of the InterProScan (Jones et al., 2014) *task*. In this example, the Perl script *interproscan2GFF.pl* parses the information obtained by the new tool and adds it to the GFF file (see Section A.3.1).

Listing 4.1: FSM for the InterProScan *task*, extracted from the *InterProFSMBehaviour* class.

```
//FSM states
this.registerFirstState(new StartBehaviour(aExecName,this.getBehaviourName()),
    "Start");
this.registerState(new
    ControllerSendMessageBehaviour(aExecName,"EXECINTERPROSCAN"),"Subtask1");
this.registerState(new ControllerCheckMailboxOneShotBehaviour(aExecName),
    "CheckMail1");
this.registerState(new
    ControllerSendMessageBehaviour(aExecName,"INTERPROSCAN2GFF"),"Subtask2");
this.registerLastState(new FinishedBehaviour(aExecName,this.getBehaviourName()),
    "Stop");
//FSM transitions
this.registerDefaultTransition("Start", "Subtask1");
this.registerTransition("Start", "Stop",1);
this.registerDefaultTransition("Subtask1", "CheckMail");
this.registerTransition("CheckMail", "CheckMail", -1); //CA waits for the TA subtask
    finishing message
this.registerTransition("CheckMail", "Stop", 1);
this.registerDefaultTransition("CheckMail", "Subtask2");
this.registerDefaultTransition("Subtask2", "CheckMail");
this.registerTransition("CheckMail", "CheckMail", -1); //CA waits for the TA subtask
    finishing message
this.registerTransition("CheckMail", "Stop", 1);
this.registerDefaultTransition("CheckMail", "Stop");
```

The CA is in charge of executing the class of a *task*, as well as sending the definition of the *subtasks* to the TAs that are going to execute them. When after the execution of the tool the InfA retrieves the information from the RDB, the fields (i.e., features) of the RDB are allocated in a specific object of the class *CandidateList* or *DomainCandidateList*. If

a tool added to the system introduces a new feature, a new object should be created in the proper class and the KB extended; otherwise no changes are required.

4.5. Evaluation

Assessing the system performance is essential to verify the precision of the results and check for possible improvements. This section presents two types of evaluation: one based on the system results (see Section 4.5.1) and other that takes into consideration its computational performance (see Section 4.5.2). These are illustrative of the kind of evaluation carried out with MASSA, Additional examples and the evaluation of features for community annotation can be found in the papers in Chapter 5.

4.5.1. Annotation assessment

With the purpose of assessing MASSA results, the 866 phylogenetically diverse sequences from CAFA (Radivojac et al., 2013) were processed by the system. These sequences together with their manually curated annotation are stored at UniProt (The UniProt Consortium, 2014), and were downloaded from this source. Since these sequences already represent a protein, they are stored as AAs. Nevertheless, in a real annotation situation the sequences are written as nucleotides and their *frame* and *strand* (see Section 2.2) are unknown. Thus, to keep the real scenario, the sequences were written to nucleic acids through *backtranseq*, a script of the EMBOSS (Rice et al., 2000) package.

After transforming the sequences to nucleotides they were grouped by species. For each specie a customized version of the used SDBs were created by removing all the sequences related to the specie. This strategy creates a more realistic environment and prevents the query sequence to be annotated with its own annotation. Then, each group of sequences was submitted separately to MASSA.

This experiment was carried out using a six-core 64-bits machine with 32GB of memory, running Ubuntu 12.04. The sequences were processed with MASSAPipe, which executed: BLASTX against NR and UniProtKB (Magrane and Consortium, 2011); RPS-BLAST (Altschul et al., 1997) against CDD (Marchler-Bauer et al., 2013); InterProScan (Jones et al., 2014) running analysis for TIGRFAM (Haft et al., 2003), ProDom (Kahn et al., 2008), PANTHER (Mi et al., 2013), SMART (Letunic et al., 2012b), and Pfam (Finn et al., 2014); and Orthostrapper (Storm and Sonnhammer, 2002).

The results were evaluated using the CAFA's protein-centric methodology (see Section 3.2.5.1). For that purpose, the GO terms produced for each annotation (CPGO) were grouped into two categories: MF and BP. The GO terms related to the manual annotation (EPGO) were downloaded from UniProt (Consortium, 2015) and also separated in both categories. Based on the CPGO and EPGO, and following the metric, *precision*, *recall*, and the F_{max} were calculated. For this assessment, it was considered a fixed score for the annotation. Therefore, the *precision-recall curve* was not drawn and the F_{max} was equivalent to the *F-measure*.

The assessment yielded an overall F_{max} of 0.86 and 0.82 for MF and BP respectively

out of 1 (see Figure 4.7). This result is very encouraging, suggesting MASSA produces high quality annotations.

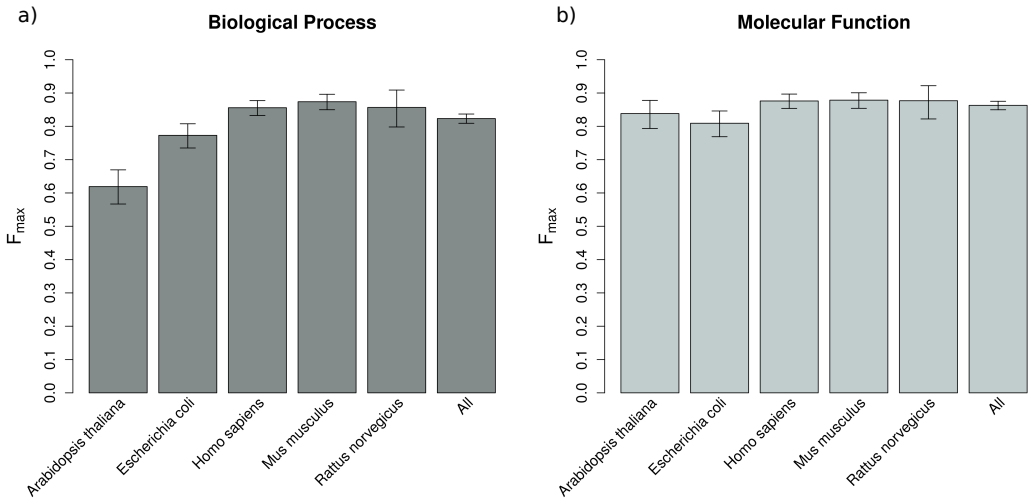


Figure 4.7: Performance assessment for a) BP and b) MF. Only species with more than 30 sequences were plotted separately. The last bar of each plot contains all the 866 sequences examined. Confidence Intervals (95 %) were calculated through bootstrap with $n = 10,000$ iterations on the set of query sequences. Annotations containing only "protein binding" as MF were excluded from the analysis due to its uninformative character (Radivojac et al., 2013).

Source: (Xavier et al., 2015)

Still following the CAFA's benchmark, a domain analysis was also carried out. This analysis showed that 95 % of the domains assigned to the annotations are from Pfam (Finn et al., 2014), although other sources have been used. Moreover, as depicted in Figure 4 from Section 5.3, the majority of the sequences were associated with only one domain, result that was highly expected (Radivojac et al., 2013). Besides, the domain distribution is very similar to the one presented in CAFA (Radivojac et al., 2013) (see Figure 3.3).

The results obtained by MASSA were compared to the ones presented in CAFA (Radivojac et al., 2013). Comparing Figures 3.2 and Figure 3 from Section 5.3, MASSA outperformed the 10 top-performing algorithms from CAFA (Radivojac et al., 2013). This outperformance is due to a combination of the system high quality inference and the evolution of the databases. CAFA experiment was carried out for 15 months since the end of 2010. After almost 5 years SwissProt (Magrane and Consortium, 2011) has

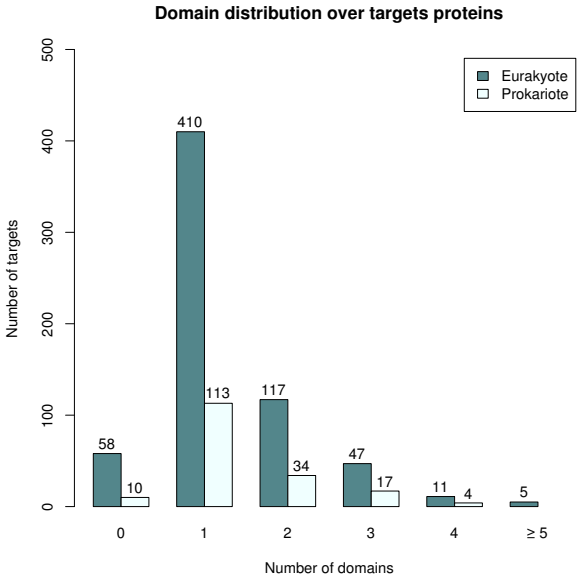


Figure 4.8: Distribution of single-domain and multi-domains from Pfam over eukaryotes and prokaryotes sequences. Eukaryotes distribution is very similar to that presented in CAFA.

Source: (Xavier et al., 2015)

increased in 5.42 %[5] [6]. As the databases are in constant evolution, it is very difficult to establish how these new data can impact this experiment. MASSA and the systems presented in CAFA (Radivojac et al., 2013) access local and remote data sources, so it is not feasible to reproduce the exact environment of this experiment.

4.5.2. Computational performance

The annotation of the 285 human sequences from CAFA was completed after little more than 17 hours. As shown in Table 4.1, MASSAPipe is the part of the system that consumes more time. That happens due to the size of the SDBs. NR, for instance is more than 48 times bigger than all other databases, which explains the difference of execution between BLAST_1 and the other tasks. After collecting the candidates information, the system spent about 1 hour (6.4 % of the processing time) to annotate the sequences and produce the final outcome.

[5] SwissProt release from September 2010 had 519348 sequences. ftp://ftp.uniprot.org/pub/databases/uniprot/previous releases/release2010_09/knowledgebase/Swiss-Prot statistics.html. [Online: accessed 10-June-2015]

[6] SwissProt current release (2015_08) has 549.088 sequences. ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/relnotes.txt. [Online: accessed 4-September-2015]

Since MASSA supports any well-formated SDB, one possible way to improve the system performance is using custom SDBs. These data sources can be created by removing a subset of sequences that are not closely phylogenetically related to the one being annotated. For example, when annotating bacteria sequences, creating a SDB that only contains sequences from bacterias like BaSys (Domselaar et al., 2005) does (see Section 3.2.4.1).

	Task	Time	Total Time
MASSAPipe	BLAST_1	15h 48min	16h 8min
	BLAST_2	3h 4min	
	InterProScan	3h 19min	
	RPS-BLAST	1h 1min	
	Orthostrapper_1	5min	
	Orthostrapper_2	20min	
MASSAInference			1h 6min
			17h 14 min

Table 4.1: Performance of MASSA for the annotation of 285 human sequences following the pipeline proposed in Section 4.5.1.

4.6. Conclusions

This chapter presented MASSA, a MAS that integrates a RBES to support accurate annotation. MASSA was designed based on a deep investigation of the domain and its tools, and with a special focus on evolution to facilitate adaptation to new researchers' needs and knowledge. In order to obtain requirements to develop MASSA, experts were interviewed and a primary research was carried out. The interviews indicated that although the annotation workflows vary, all experts use BLAST, and that different pipelines can go towards the same prediction. Besides, it also indicated the output attributes experts take into consideration when making decisions, as well as some limitations in the domain.

To complement this investigation, an extensive study of the tools and systems developed to tackle the annotation problem was carried out as well. This analysis was very valuable for many reasons. First, it gave a deeper understanding of the domain and its limitations. Second, it pointed out what some systems are doing to overcome these constraints. Third, it also indicated what developers take into consideration when designing these systems. Fourth, it showed the knowledge the systems are considering. Fifth, it also highlighted the most popular tools and data sources used. Sixth, it pointed out some weaknesses to be avoided in these systems. Seventh, it indicated good practices that should be incorporated to the proposed system.

In order to gather less common requirements for the system, an analysis of the community annotation emerging practices under the light of the AT (Leontiev, 1978) was also made. This analysis pointed out some reasons of the lack of success of these efforts, and also some requirements that could be added to the system to improve its accuracy and allow a community support.

Based on all the results of these studies and following two state-of-the-art methodologies, CommonKADS (Schreiber et al., 2000) and INGENIAS (Pavón et al., 2005), MASSA was developed. The system presents a very modular and flexible architecture that comprises two subsystems MASSAPipe and MASSAInference. This decouples the knowledge acquisition from the inference. This architecture and the way the system was developed facilitate the system maintenance and evolution. Besides, it makes the annotation process fairly transparent to the user, easing its understanding.

After implemented, the system annotation was assessed using the metrics proposed by CAFA (Radivojac et al., 2013). This evaluation suggests that the system can produce accurate annotations. When compared to the top-performing algorithms from CAFA (Radivojac et al., 2013), MASSA also showed a high level of accuracy, outperforming all the systems. A computational performance analysis was also carried out. It showed the system is able to annotate about 300 sequences in little more than 17 hours. This result is mainly due to the size of NR, but is in line with other tools for this problem.

Chapter 5

Papers presented

There are really three parts to the creative process.
First there is inspiration, then there is the execution,
and finally there is the release.

Eddie Van Halen

This chapter presents the publications that are part of this work.

5.1. Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology

5.1.1. Citation

Xavier, D., Morán, F., Fuentes-Fernández, R. and Pajares, G. Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology. *Expert Systems with Applications*, vol. 40(10), pages 3943-3952, 2013.

5.1.2. Abstract

Finding the genes that exist within a DNA sequence and assigning them biological features and functions is one of the biggest challenges of Genomics. This task, called annotation, has to be as accurate and reliable as possible, because this information will be applied in other researches. Ideally, each sequence should be annotated and validated by a human expert, who has the knowledge to infer the most appropriate annotation. Nevertheless, the huge amount of genomic data produced by the new sequencing technologies prevents this practice. Developing expert systems that are able to annotate sequences automatically and emulate the expert involvement in certain key points of the process

would enhance the annotation quality. In this work, the CommonKADS methodology is innovatively applied for this purpose. It is used to structure and model the knowledge required to build an expert system able to deal with the functional part of sequence annotation, i.e. establishing the biological purpose of the sequence. This approach provides the first general framework for the aforementioned problem, which can be easily extended to related issues.

5.1.3. References

(Akerkar and Sajja, 2010), (Alonso and Guijarro, 2004), (Altschul et al., 1997), (Awad and Ghaziri, 2004), (Benson et al., 2011), (Berger and Leighton, 1998), (Bernstein et al., 1977a), (Burton et al., 1990), (Crescenzi et al., 1998b), (Curwen et al., 2004), (Durbin et al., 2011), (Edwards et al., 2009), (Finn et al., 2010), (Flicek et al., 2011),(Friedberg, 2006), (Giarratano and Riley, 1998),(Gouret et al., 2005),(Holland et al., 2008), (Hunter et al., 2012), (Iglesias et al., 1997), (Keedwell and Narayanan, 2005), (Lehninger et al., 2008), (Marchler-Bauer et al., 2011), (Pajares and Santos, 2005), (Pevsner, 2009), (Potter et al., 2004), (Rice et al., 2000), (Sayers et al., 2012), (Schreiber et al., 2000), (Stajich et al., 2002), (Generic Model Organism Database (GMOD) Project, 2013), (The Gene Ontology Consortium, 2000), (The JBoss Drools Team, 2012), (UniProt Consortium, 2012).

Expert Systems with Applications 40 (2013) 3943–3952

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology

Daniela Xavier [a], Federico Morán [a], Rubén Fuentes-Fernández [b], Gonzalo Pajares [b,*]

[a] Department of Biochemistry and Molecular Biology I, Universidad Complutense de Madrid, Avd. Complutense s/n, 28040 Madrid, Spain
[b] Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, C/Profesor José García Santesmases s/n, 28040 Madrid, Spain

ARTICLE INFO

Keywords:
CommonKADS
Expert system
DNA sequence
Functional annotation
Bioinformatics

ABSTRACT

Finding the genes that exist within a DNA sequence and assigning them biological features and functions is one of the biggest challenges of Genomics. This task, called *annotation*, has to be as accurate and reliable as possible, because this information will be applied in other researches. Ideally, each sequence should be annotated and validated by a human expert, who has the knowledge to infer the most appropriate annotation. Nevertheless, the huge amount of genomic data produced by the new sequencing technologies prevents this practice. Developing expert systems that are able to annotate sequences automatically and emulate the expert involvement in certain key points of the process would enhance the annotation quality. In this work, the CommonKADS methodology is innovatively applied for this purpose. It is used to structure and model the knowledge required to build an expert system able to deal with the functional part of sequence annotation, i.e. establishing the biological purpose of the sequence. This approach provides the first general framework for the aforementioned problem, which can be easily extended to related issues.

1. Introduction

With the advent of new sequencing technologies, organisms and even complete ecosystems can be sequenced at low cost and in short periods of time. Generating genomic data is no longer a problem, but to process, analyse and apply these data in a significant and useful way still are (Friedberg, 2006). The huge amount of genomic data slows down or even prevents the execution of whatever process that needs the human intervention during processing or analysis stages.

Gene annotation is one of the major challenges of the Genomics. This task consists in finding the genes that exist within a DNA sequence and assign them biological features, such as the name of the protein they code for or the biological process they are involved in. The *structural annotation* discovers where are the known genes, genetic markers and other landmarks. The *functional annotation* aims to predict the biological function of genes and proteins. This paper takes into account only the functional annotation.

The annotation assigned to a sequence should be as accurate and reliable as possible, since it could be used in further biological, medical, or pharmaceutical researches. Moreover, uploading annotated sequences to public databases and using this information to annotate other sequences is a common practice in the Bioinformatics community. Therefore, a miss-annotation could be propagated to future annotations (Friedberg, 2006).

The process to annotate a sequence involves the execution of pipelines composed by many Bioinformatics programs. Highly skilled professionals analyse their outputs and, based on their Biological and Biochemistry knowledge, infer the most appropriate information for each sequence. In spite of the trustful and accurate character of the manual annotation, this process is extremely time consuming, labour-intensive, and expensive, and therefore, it is not suitable for great volumes of data. In order to avoid the drawbacks of manual annotation, automated annotation methods need to be employed for the vast majority of genomes (Edwards, Stajich, & Hansen, 2009).

This approach sacrifices some quality in the annotation to get results faster and cheaper by removing the experts. However, given the potential impact of miss-annotations, its results has to be manually revised (or *curated*), recreating again the manual bottleneck.

One possible solution to this problem is to design an Expert System (ES) that is capable of emulating the human expertise during the annotation process. However, develop such system is a laborious and complex task that requires a deep comprehension of the applications domain. Moreover, it requires being able to elicit the relevant knowledge and providing it in a suitable format for automated processing.

Obtaining the knowledge required to solve the studied problem is a crucial phase in the development of any Knowledge-Based

* Corresponding author. Tel.: +34 1 3947546; fax: +34 1 3947547.
E-mail address: pajares@fdi.ucm.es (G. Pajares).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.12.088

System (KBS), and in particular ES. The success of the system depends to a large extent on the accuracy of the information acquired. There are knowledge elicitation techniques that can facilitate this process, such as structured interviews, protocol analysis, and laddered grid (Burton, Shadbolt, Rugg, & Hedgecock, 1990).

The development of Knowledge-Based Systems (KBS) requires the ability to understand and structure the knowledge in order to incorporate it into the ES. Knowledge Engineering provides methodologies that facilitate this process and consequently helps the systems design and implementation. CommonKADS (Schreiber et al., 2000) is a leading methodology to support structured knowledge. This methodology is a flexible and powerful tool that can be employed in any context-based problem.

This work consists in using CommonKADS approach to analyze and structure the knowledge required to develop an ES for functionally annotating DNA sequence without taking into account the genomic context. As far we can ascertain, this is the first formal description of the application of this methodology in the bioinformatic field. Therefore, here is presented a novel general framework to the functional annotation problem that can be adapted for different pipelines and extended to related matters. The knowledge employed here was obtained using the knowledge elicitation technique *structured interview* and was extract from experts in Biology and Bioinformatics.

The paper is organised as follows. Section 2 provides the background needed to understand the annotation problem, and describes the current state of the art in tools for annotation. The presentation of the proposed approach starts in Section 3 with an overview of the CommonKADS methodology and the knowledge elicitation techniques applied. Section 4 describes the knowledge modelling process for the aforementioned problem, while Section 5 characterises the main requirements for the related system. Finally, Section 6 discusses the conclusions and future work.

2. Biological background

The hereditary information of all living being, with exception of virus, is stored in a macromolecule called Deoxyribonucleic Acid (DNA). This molecule consists of two complementary long strands, linked through hydrogen bonds, twisted around each other, forming a double helix shape. The strands are mainly composed of smaller molecules called nucleotides. Each nucleotide, in turn, consists of a deoxyribose sugar, one phosphate and one of the four nitrogen-rich bases: Adenine (A), Guanine (G), Thymime (T) and Cytosine (C). A DNA sequence is formed by the combination of these four bases (Fig. 1a) and its length is measured in base pairs (bp), that is, according to the number of bases in the string. The complete sequence along each strand is called genome.

Some of these sequences contain the genetic information required to produce other cellular components, such as proteins. These sequences are the genes and are responsible for life maintenance and transmission of genetic traits to the descendants. In most organisms, genes consist of exons and introns, that is, coding and non-coding sequences respectively. During the transcription (see Fig. 1a), exons are transcribed into Messenger Ribonucleic Acid (mRNA), which is further translated (see Fig. 1a) into a sequence of amino acids.

Proteins are present in every cell and are responsible for all metabolic processes produced in an organism and for the life maintenance. In a very simplistic way, a protein can be viewed as a long chain of thousands units called amino acids held together by peptide bonds. Due to a biological phenomenon called alternative splicing, where different combinations of exons can be used to create a mRNA, one gene is capable of coding different proteins (Fig. 1b).

The role of a protein in the metabolic processes depends on its structure. The protein sequence is its primary structure, which can be written as a combination of the twenty amino acids letters (see Fig. 1c.1). This structure sometimes folds, creating turns, helixes or sheets, which form a two-dimensional organisation called secondary structure (see Fig. 1c.2). The set of secondary structures folded in the space forms the tertiary structure (see Fig. 1c.3). This three-dimensional structure is determined by its amino acids sequence and is directly related to the protein function. The prediction of the tertiary structure based on its primary organisation has been shown to be a NP-complete problem (Berger & Leighton, 1998; Crescenzi, Goldman, Papadimitriou, Piccolboni, & Yannakakis, 1998).

As depicted in Fig. 1d, the protein architecture can be composed of one or more *families* and *domains*. There are some parts of a protein that can evolve, function and exist independently of the whole protein chain. These regions are denominated *protein domains*, and are usually high-density zones due to the accumulation of folds. A *protein family* is a group of evolutionary-related proteins that have similar primary and/or tertiary structures and resembling functions. In general, proteins that share domains are grouped in the same family. The members of a protein family are known as homologous proteins. If two homologs are present in the same specie, they are called paralogs, whereas, if they are in different species, they are ortologs.

The process of tracing evolution involves first identifying suitable families of homologous proteins and then using them to reconstruct evolutionary paths (Lehninger, Nelson, & Cox, 2008). Comparing DNA is based on the fundamental assumption that if two DNA sequences are similar, they probably share the same function, even if they occur in different parts of the genome or across two or more genomes (Keedwell & Narayanan, 2005). On the other hand, protein comparison starts from the premise that the linear sequence determines the tertiary structure which, in turn, determines the function.

There are many algorithms that align and compare two or more sequences (nucleotide/amino acids) or do searches through genetic databases in order to find the sequence's evolutionary relatives or to infer the sequence function based on known ones. BLAST (Altschul et al., 1997) is one of the most widely algorithms used to do homology searches.

In order to extract genetic information from a DNA sample, this molecule has to be partial or completely sequenced. Next Generation Sequencing (NGS) technologies allow sequencing huge volumes of DNA in short periods and at low cost. However, this high throughput methods generates a massive amount of data that should be processed and analysed.

Interpreting the genomic data and assigning it biological meaning is one of the major challenges of the post-genomic era (Gouret et al., 2005). Annotation is the process by which the landscape of genomic DNA is surveyed, and key features of the sequence are described (Pevsner, 2009). The *structural annotation* aims to find genes, their location in the sequence, the intron/exon structure and predict the protein sequences they encode (Gouret et al., 2005), while the *functional annotation* is centred in discovering the biological function of the sequence.

Gene Ontology (GO) (The Gene Ontology Consortium, 2000) was created in order to standardising the representation of the functional annotation across species and databases. This controlled vocabulary of terms has a graph structure and describes the gene biological process, molecular function and cellular component.

The annotation process can be carried out manual or automatically. The first clearly guarantees highest-quality data and most accurate gene structures, but this process is slow and can produce conflicting interpretations of the analysis (Potter et al., 2004). On the other hand, the automated annotation is faster, does not

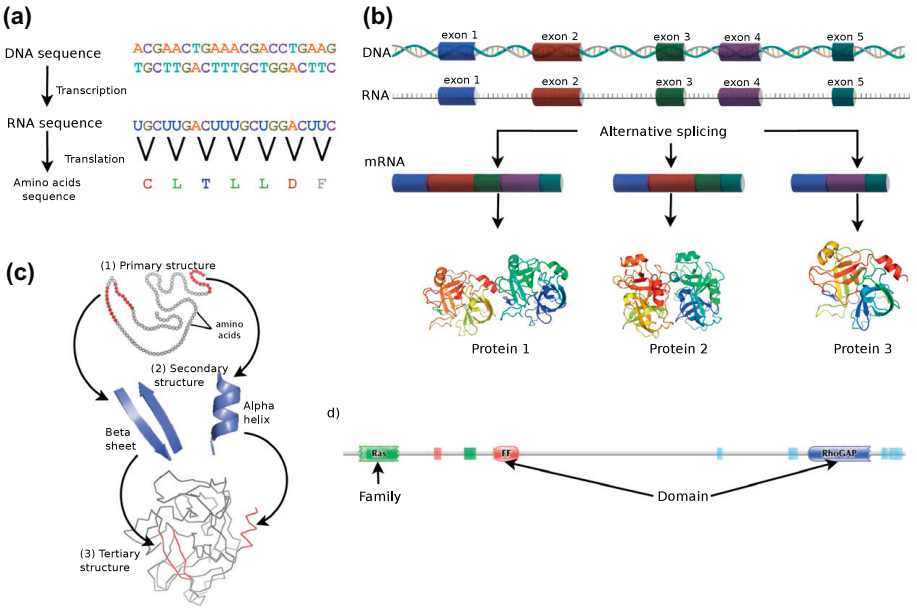


Fig. 1. (a) Simplified central dogma of molecular biology. (b) Alternative splicing schema: one gene can code more than one protein. (c) Protein structure. (d) The protein B2RTN5_MOUSE predicted architecture consists of one family (RAS) and two domains (FF and RhoGAP).

require an expert team, and maintains the consistency of the analysis, but it is less accurate and the results can be less specific. Although, ideally, the annotation should be done by experts who have biological and biochemistry knowledge to infer the features that suit better a certain DNA sequence, the advent of high throughput sequencing methods prevents this practice. Therefore, there is need to develop more accurate automated annotation methods. Moreover, since the functional annotation is mainly carried out through homology techniques, the genetic databases have an important role in this task, as they will be the source of the information for the new annotations. This fact reinforces the necessity of really accurate annotations in order to avoid error propagation.

A great variety of genetic information is publicly available and can aid during the annotation and other processes as well. There are databases for genes and genomes (e.g. GenBank (Benson, Karsch-Mizrachi, Lipman, Ostell, & Sayers, 2011) and Ensembl (Flicek et al., 2011)), proteins (e.g. NCBI Entrez Protein Database (Sayers et al., 2012) and UniProt (UniProt Consortium, 2012)), domains and families (e.g. NCBI's Conserved Domains (CDD) (Marchler-Bauer et al., 2011), InterPro (Hunter et al., 2012), and Pfam (Finn et al., 2010)), gene ontologies (e.g. GO (The Gene Ontology Consortium, 2000)), tertiary structure (e.g. PDB (Bernstein et al., 1977)), and so on. Even though some of these databases are linked to each other through cross-references, in general, each one has its own format and sometimes it is difficult to gather related information.

Currently, some ES annotators, such as Ensembl (Curwen et al., 2004) and FIGENIX (Gouret et al., 2005), have been developed. Nevertheless, these systems have some limitations: the first one was designed to deal with complete genomes, while the second is only available to online execution. Regarding the first issue, there are many scientific researches, such as gene discovery, that do not need to sequence and annotate the whole genome of an organism, as they are focused on specific sequences. Thus, creating an ES capable of annotating DNA sequences without considering the genome context will aid them. This would contribute to enhance the quality of the genomic data uploaded into the public databases and consequently, contribute to improve future annotations. The second issue is related to the fact that online systems oblige users to submit their data to external servers. This is an important drawback, as the annotation frequently works on sensitive information.

3. CommonKADS and knowledge elicitation

3.1. CommonKADS

CommonKADS (Schreiber et al., 2000) is a flexible methodology that offers a set of tools to model KBS regarding not only the knowledge needed but also its context and proposal. Through this methodology, it is possible to identify the strategy that best fits the problem to solve. Apart from that, it establishes the methodological bases to tackle the problem in a general way, allowing these bases to be applied to any similar problem, independently of its complexity. Other benefits of using CommonKADS include improved communication, standardization, technology support, and availability of reusable components (Akerkar & Sajja, 2010).

The CommonKADS process is organised around three analysis activities that specify six models. Fig. 2 summarises it.

In the *Context* analysis, an *Organisation Model* is used to understand the organisational context and environment where the problem lays. This supports the evaluation of the feasibility and benefits of employing a KBS to solve the problem. During this phase, the organisational task layout and the agents that perform theses tasks are also analysed, through a *Task* and an *Agent Model*, respectively.

The second analysis, the *Concept*, is focused on the conceptual description of the knowledge applied in the task. The *Knowledge*

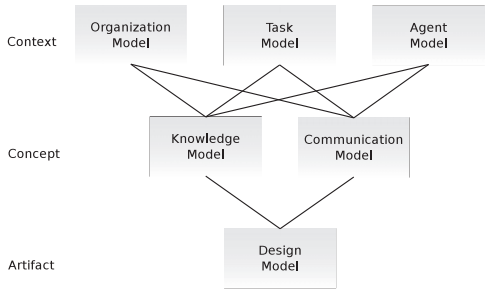


Fig. 2. CommonKADS models (Schreiber et al., 2000).

Model takes into consideration the knowledge and reasoning requirements to develop the system, while the *Communication Model* deals with the communicative transactions between the agents involved.

The main focus of the last analysis, i.e. *Artifact*, is the technical aspects of the computer implementation. This phase culminates in the *Design Model*, where the technical system specifications needed to implement the functions proposed in the knowledge and communication models are specified. The construction of all models is not mandatory. It depends on the goals of the project as well as the experiences obtained during its development.

The current work is centred in describing the knowledge model. Its major goal is analysing and structuring the knowledge required to develop an ES for solving the annotation problem. For this purpose, it performed complete context and concept analyses, and partially the artifact one.

3.2. Knowledge elicitation

Before modelling the knowledge, it is essential to extract it from the sources that contain this information. Knowledge Elicitation (KE) provides a set of techniques to facilitate the acquisition of the material required to structure a more formal description of the problem. It generates organised data such as diagrams, lists, informal rules, and formulae. This task is the main bottleneck of the KBS developing, since this activity demands much time and its results should be accurate and robust. If the knowledge is not correctly obtained or contains errors, the system will be doomed to fail.

Information can be extracted from different kinds of sources related to the domain. Experts are one of the most important sources of knowledge and sometimes they have information that was not previously catalogued. However, this information is structured and stored in a complex cognitive way, what hinders its acquisition (Alonso & Guijarro, 2004). Therefore, to be successful in this task, it should be focused and structured, but also as open as possible. Moreover, it is important to take into account which technique fits better in the problem-solving domain, the type of expert from whom the knowledge will be obtained, and the type of knowledge to be extracted.

Many techniques can be employed, separately or complementarily, to extract knowledge. The interview is the most popular of these techniques and has the benefit that it is a low cost method that does not require any special training from the knowledge engineer. Other advantage is that this technique allows the obtainment of different types of knowledge at distinct levels of the KBS developing process, independently of the application domain. The interview can vary from completely unstructured to formally planned. The unstructured interview has no detailed planning and, consequently, few constrains. In contrast, the structured interview is the formal version of the first. In this case, the knowledge engineer plans and manages the session aiming to extract specific knowledge from the expert.

The type of expert who facilitates the information is crucial for the success of the KE process as well. Aspects such as communication and verbalization skills, and attitude towards the domain have a great influence in the way the knowledge engineer should conduct the information extraction. Experts can be classified as academics, practitioners or samurais (Schreiber et al., 2000). The formers are accustomed to verbalize their knowledge and structure it in a logical way, but they do not apply it in a regular basis. The second kind deals with the problem-solving daily, and since their theoretical knowledge about the subject and the domain is not as deep as that of academic experts, they are used to apply heuristics in their decision process. The latter carry out their job in an automatic way due to their lacking of theoretical training.

The KE method is directly linked to the type of knowledge to be obtained. For that reason, it is essential to apply the technique that best suits the type of information regarding the domain. According to Awad and Ghaziri (2004) the knowledge can be procedural, declarative, semantics or episodic, according to the depth (from shallow to deep) of understanding of the problem area.

Procedural knowledge is the knowledge applied to carry out a specific procedure. It is the most shallow and is related to skills that demand the repetition of the knowledge over and over again, such as psychomotor tasks or learning a language. It is so highly automated that becomes a natural part of the person. Since this knowledge is so rooted in the expert, it is difficult to be verbalized.

On the other hand, the declarative knowledge is the one that could be verbally expressed in a simple way. This type is often in the short-term memory, and for that reason it is related to uncomplicated information that is ready to be recalled. This knowledge is useful for early stages of the KE and it can be properly acquired through structured interviews.

Semantic knowledge is a deeper kind of knowledge, which resides in the long-term memory, and reflects the cognitive structure, representation and organisation of the expert. It includes major concepts, vocabulary, facts, and relationships.

Episodic knowledge is the one based on the experimental episodes the expert faced during her/his life. However its application is automated for the expert, this knowledge is so chunked in the long-term memory that the expert experiment difficulty in remembering and explaining it.

This work aims to comprehend how experts solves the annotation problem and, from this understanding, extract the rules and heuristics used by them during this process. The experts interviewed have an academic/practitioner profile, as they have both theoretical and practical knowledge, and are very familiar with the domain. The type of knowledge to be acquired is mainly procedural, since it is related to how a task is carried out, but it has declarative, semantic, and episodic characteristics as well. This makes its extraction even more complex.

The KE technique employed was the structured interview because it perfectly suits the type of knowledge we intend to obtain. Moreover, this approach produces structured data that are easier to analyse, facilitating the process of extracting usable knowledge.

The interviews held with different experts gave a global vision of the annotation process and helped the engineer to obtain specific rules employed by these specialists. At this stage, the main conclusion is that each expert employs her/his own annotation pipeline, what makes very important to design a flexible annotation model that can be modified according to the expert's knowledge and needs.

Moreover, after comparing the interviews, it is possible to summarise the following findings: (1) different paths can lead to the same solution; (2) BLAST (Altschul et al., 1997) is used by all experts, despite of the pipeline; (3) some output attributes of programs, such as e-value, are key to guide decisions in the annotation process; and, (4) each program consults different databases, for instance, Ensembl's (Flicek et al., 2011) *protein databases* and NCBI's *non-redundant protein database* (NR) are processed by BLASTx, while CDD (Marchler-Bauer et al., 2011) database is handle by RPS-BLAST (Altschul et al., 1997).

4. Knowledge model

The main focus of this work is to organise and structure the knowledge needed to design the ES proposed. Since one of the major challenges of Knowledge Engineering is to discover structures that are capable to model the knowledge in a schematic way, methodologies that are able to accomplish this goal excel in this field.

CommonKADS, as depicted in Fig. 2, supports a Knowledge Model (KM) that facilitates the structure of a knowledge-intensive information-processing task. This model is structured similar to traditional analysis models in Software Engineering. It is composed of three knowledge categories, hierarchically arranged, which are discussed in next sections for the annotation problem (see Fig. 3).

4.1. Domain Knowledge

The Domain Knowledge includes the leaves of the tree (see Fig. 3c), and covers the domain-specific knowledge and the types of information tackled in the application. This category has two basic elements: Domain Schema and Knowledge Base (KB).

The domain schema describes the domain-specific knowledge and information schematically, by means of the association of attributes and values to certain domain instances that share similar features. These instances are called *concepts*, and each one of their attributes requires a *type* (e.g. concepts, relations, and rule types) to specify the values accepted by these attributes. Fig. 4a depicts the concept of sequence and some of its attributes and values. Concepts can be linked through relationships, as the one in Fig. 4b, which indicates that a DNA sequence codes zero or more proteins. Complex dependencies between concepts are represented by logical relationships in rules. Fig. 4c shows an example of a rule related to the analysed domain: *if a sequence is a pseudo-gene then it cannot code a protein*. The domain schema can also contains super/subtypes (see Fig. 4d), and can be described at different levels of abstraction.

In order to understand the scope of the application is important to be able to schematize it and to create a KB containing the instances of knowledge types existents in the Domain Schema. The KB built in this work consist of the information obtained during the KE process together with biological data from public genetics databases, such as GenBank (Benson et al., 2011), CDD (Marchler-Bauer et al., 2011), and Ensembl (Flicek et al., 2011).

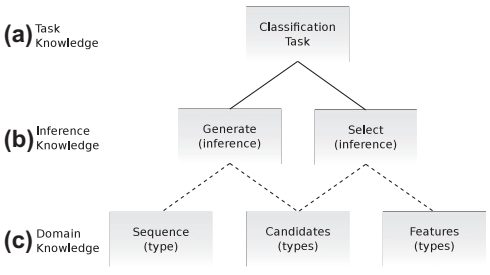


Fig. 3. Knowledge categories schema for a DNA sequence annotation application.

4.2. Inference knowledge

The Inference Knowledge is the middle knowledge category of the tree (see Fig. 3b), and describes the basic steps required to use the Domain Knowledge. This knowledge mainly consists of *inferences*, *knowledge roles*, and *transfer functions*.

Inferences apply the knowledge existent in the KB to deduce new information from an input. This element can be understood as a block that represents the machine reasoning, that is, actions to verify, order, generate, predict, and evaluate. The inference is described as a function of its input and output, without explaining the process employed to generate the output. These input and output, in turn, are structured in terms of the roles they play in the reasoning process. A knowledge role is called dynamic when it varies according to the invocation of the inference, and static if it stays to a certain extent stable during the process. Static roles generally make part of the Domain Knowledge employed to make the inference. In Fig. 4.e, the static information of the role DB (member of the KB) is applied to generate protein candidates to the input sequence.

The transfer function is responsible for the communication with the external world. It transfers information between the reasoning process and the environment that is outside of the system, such as another system or a user. There are four types of transfer function and they vary according with who has the initiative, the system or the exterior, and where the information resides, inside or outside the system. The functions *obtain* and *receive* are the most used in knowledge models (Pajares & Santos, 2005).

Fig. 5 depicts a general vision of the proposed application that can be adapted to different annotation pipelines. The system *receives* the dynamic role *sequence* from the *user* and passes it to a *program*, which using a specific *database* (an static role) *generates candidates* and *selects* the best ones based on *rules* (another static role). The rules change according to the program and the approach used. They are based on attribute features of the output, such as BLAST's e-value or bitscore. The *results* of the rule filtering are represented by a list of data and its respective attributes and values.

The set of inferences *generate*/*select* can be applied for n programs, creating a flexible design that can be adapted to other pipelines. Each one of the n programs runs in parallel using its own database and rules. After the execution of these programs, the content of their output (*Results* in the Fig. 5) is inserted into a database.

It is also possible to run programs that depends on the output of others, as occurs in Fig. 5.a. In this case, the output obtained is employed as input of another program in order to generate complementary data. Even though these data are not essential for the decision process, they can reinforce the annotation inferred, raising its reliability.

Once all programs are finished, the transfer function *present* retrieves *all* [*the*] *results*, and these are compared based on *rules* of the KB. The final result, the dynamic role *protein*, is *presented* by the system to the *user* together with a log file that details all the reasoning process carried out.

Fig. 6 exemplifies the application of the general schema previously proposed to a system that contains three programs ($n = 3$): BLASTx (Altschul et al., 1997) against Ensembl (Flicek et al., 2011) *protein*, BLASTx against NR and RPS-BLAST (Altschul et al., 1997) against CDD (Marchler-Bauer et al., 2011). In this example, the pipeline were designed to annotate data from *Sparus aurata* (sea bream fish), thus a protein database containing all model

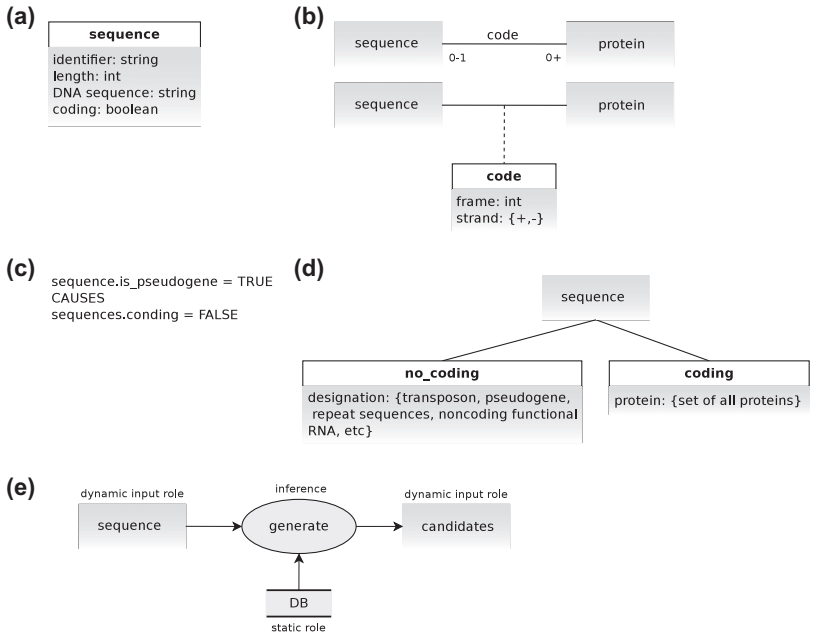


Fig. 4. Different abstractions of the Domain schema: (a) concept, (b) relationship, (c) rule, (d) subtype and, (e) knowledge roles for the inference generate.

fishes from Ensembl were created in order to complement, in a more specif way, the information of the NR database.

Due to the model modularity and the reusability of the approach, it is quite simple to alter the pipeline. Programs and databases can be changed according to the problem to solve and the strategy traced. Moreover, since the rule descriptions also follow a modular structure, it is easy to add or remove rules without causing changes in the main code.

4.3. Task knowledge

The Task Knowledge is located in the root of the knowledge category tree, as shown in Fig. 3a. It is crucial for the system design, as it describes the objectives to be achieved through the knowledge and the strategies to be implemented in order to reach them. This knowledge describes a recursive decomposition of a top-level task in sub-tasks, which in turns are decomposed into simpler subtasks, as depicted in Fig. 7. At the lowest level of this decomposition, the tasks are linked to one or a series of inferences and transfer functions in the Inference Knowledge category.

The main elements of this category are the *task* and the *task method*. The former is related to the question "what can be done?", and defines a complex reasoning goal in terms of input and output roles. Each task has a task method that describes how this task is carried out through its decomposition into sub-functions. The task method answers the question "how can it be done?", and is composed of a set of sub-functions and a control structure that specifies the order in which these functions are carried out.

When modelling the task, a straightforward approach is to base it upon a generic task pattern from CommonKADS Template Knowledge Models library (Schreiber et al., 2000). These templates specify the steps to be taken as part of a specific task, preventing the engineer from "reinventing the wheel". They have proved to be useful in developing a range of common systems for many projects (Schreiber et al., 2000). The templates can be used in different domains without significant changes and be reused regardless of the application, what is one of the greatest advantages of this methodology.

Task templates are divided according to the system the task operates on. They can be analytic, if representing an existing system that is not completely characterised. In this case, the objective of the task is producing more information about the system. In contrast, the synthetic approach is used when the system does not exist and the purpose is constructing its first description.

Since the goal of this work is to describe a system that is not completely known, the analytical approach is the one that fits best. This category specifies tasks such as classification, diagnosis, assessment, monitoring, and prediction.

The task of the functional annotation of DNA sequences can be viewed in an objective way as finding the protein "class" that best characterises a certain sequence. This premise fits perfectly the well-known analytical task of classification. This task usually involves objects from the nature, such as animals and plants, and its objective is to discover the association between the features of an object and a class from a predefined set of classes.

Each generic task template has four basic elements that together facilitate the task specification: the *general characterisation*, the *default method*, the *variation method*, and the *typical schema of domain*.

The first, the general characterisation, describes the features of the typical task. This part defines the *objective* of the task. It also specifies the *terminology* applied to describe the object to be classified in function of its *class*, *attributes*, and *features*. The object *input*, the class *output*, and a typical *example* are given as well. Table 1 shows an example of this characterisation.

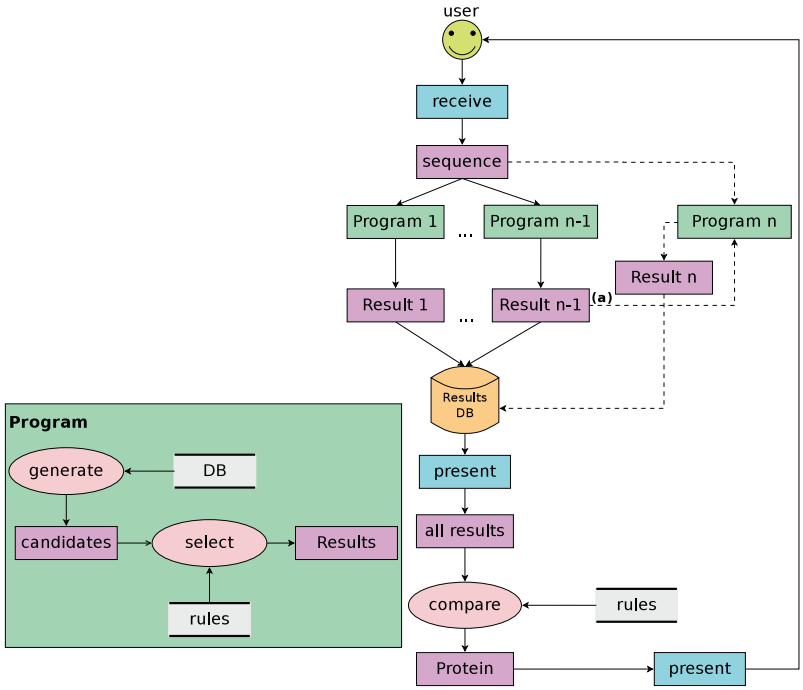


Fig. 5. The general schema for the application: inferences, knowledge roles and transfer functions.

The default method represents the first decision to be made. It can be data-driven, when the object's initial features are used to generate a set of candidate solutions, or solution-driven, if the starting point is the set of all possible solutions and this set is reduced based on the information acquired. In the case that the default method cannot be used, method variations can be applied to reach the solution.

In this system, the candidate solution set is obtained from the initial data through alignments against different databases, therefore the default method is data-driven. Moreover, all input objects can be processed by the default method, so there is no need to use method variations. The schema for the sequence classification task and its inference structure is very similar to the one depicted in Fig. 5, but without the static roles.

5. Analysis and implementation

The system for functional annotation will be a rule-based ES according to the schema proposed in Fig. 6. The previous sections describe the acquisition of the knowledge about the domain of the application, and model the generic task to be carried out by that system. Its rules were obtained through interviews held with experts, and complemented with the previous knowledge in the domain of the engineer. The development of the system also requires specifying additional requirements such as its execution environment and user interface, and defining some basic components of an ES (Giarratano & Riley, 1998), such as its KB, working memory, and inference engine.

The ES will be first developed for Ubuntu 10.04 and then extended to other operating systems. The user interface will be text-based and graphical as well, in order to be as much flexible and friendly as possible.

The system's KB comprises external databases from different public sources and a collection of rules compiled based on the knowledge of the experts. These rules will be described according to the production rule system Drools (The JBoss Drools Team, 2012). Table 2 describes the KB elements and their respective formats.

The active memory, where the input and intermediate data are kept, will be stored in a physical database (see Fig. 6.a). If needed, at the end of the annotation process the user can dump these data to further analysis.

Since Drools will be used as the rule management system, the main inference engine will have a forward chaining approach. This engine, represented by the inference *compare* (see Fig. 6b), is fed by the KB and its own agenda, and it infers the final annotation, i.e. *Protein*. Apart from that, each program will have its own inference engine in order to achieve the best results.

One advantage of rule systems, such as Drools, is that they already provide explanation facilities. In this application, the reasoning process will be summarised in a log file, which will be available for the user under request.

In a first version, the knowledge acquisition facility will not be implemented, as this implies a relevant effort on interface design and validation. Nevertheless, it could be easily added to the system in the future thanks to its modularity.

The overall system has been designed as a collection of pipelines that can be carried out in parallel. Its single input is a DNA sequence file in FASTA format, as shown in the example of Table 1. After the execution of all the pipelines, the results are gathered

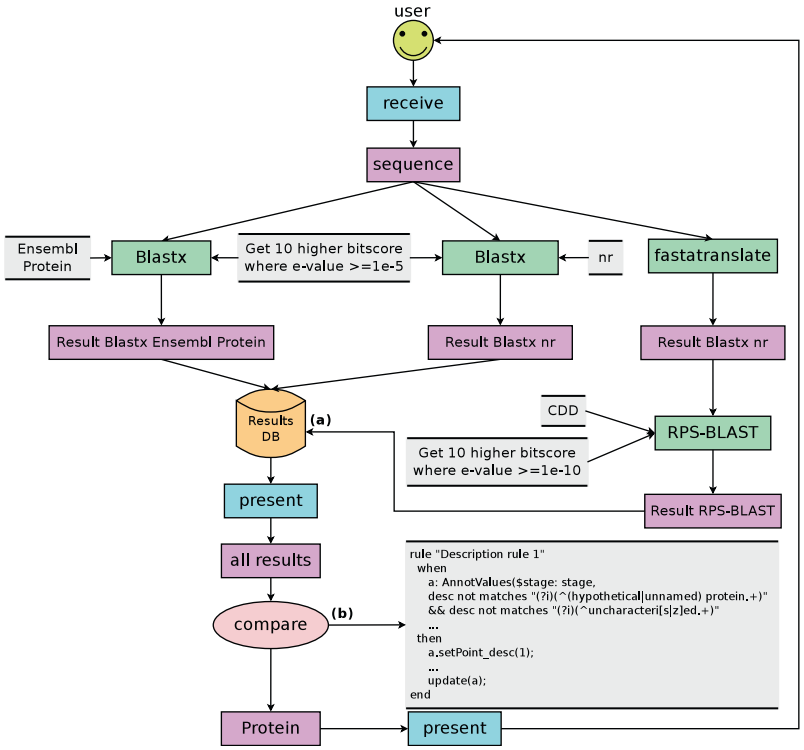


Fig. 6. General schema applied to a concrete pipeline: BLASTx against Ensembl Protein and NR databases and RPS-BLAST against CDD.

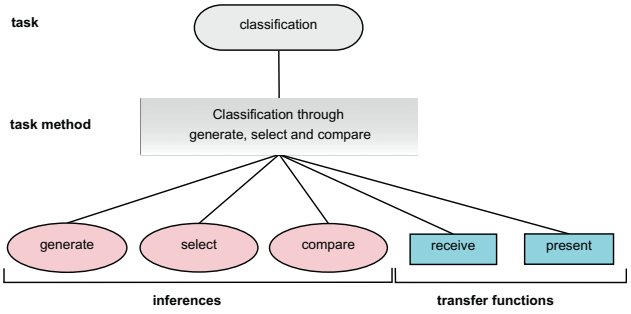


Fig. 7. Diagram for decomposition of the classification task.

and analysed, generating a GFF (Durbin, Haussler, Stein, Lewis, & Krog, 2011) file, an annotated FASTA file, and a reasoning trace file.

The implementation will be mainly in Java due to its communication with Drools, its portability and its facilities to build user interfaces. At first, some reused Perl scripts will complement this implementation, but they may be replaced by Java code in the future.

Perl scripts will be used in the implementation of the inference select, during the *Program* stage (see Fig. 5). BioPerl (Stajich et al., 2002) module Bio::SearchIO, which handles Bioinformatics formats such as the BLAST output, aids the processing and parsing of this inference outcome, facilitating the application of filtering rules. The selected information will be then written in a intermediary file, through Bio::Tools::GFF module, and finally loaded into the Results DB by *bp_load_gff.pl*, a Perl script from the GBrowse (Stein et al., 2002) packge that uses the module Bio::DB::GFF.

Java framework for handling biological data, BioJava (Holland et al., 2008), and its extension, BioJavax, will be employed as well.

Table 1
General characterisation for the system classification task.

Objective	Infer the protein corresponding to a given sequence. This classification is based on values and features obtained through local alignments of the object sequence with sequences from different databases
Example	>gi|34783212 Homo sapiens mitochondrial protein S35 AGGAACTCTAACTCCTTGCCACTCAAGAAATGTCCTTCAGAATATGCCTTCCGCATGTCGTCT CAGTGCCCGGCTATTGGTGAAGTCACCAGGCCTACTAATTCCAAGTCTATGAAAGTGGTGAAACTGTTT AGTGAACTGCCCTTGGCCAAGAAGAAGGAGACTTATGATTGGTATCCAAATCACCACACTTACGCTGAAC TCATGCAGACGCTCCGATTTCTTGGACTCTACAGAGATGAGCATCAGGATTTTATGGATGAAAGGATGAAAACCAAGGAAAGGAAGGGAAAAGAGCAGCAAAAAGGAAATAG TGTTGGGTCCCTCAAGAGGAGACTTTCTTCCTCAGTGGCGGAGAAGAAGTGCATTTATTGTCTTTCC ACATATTGGAGGAATGTCATCTTTCTAAATGAAGTTTATTTGGAGGAACACAGTGCATCTCCTTGGTGAAA TCTAATCCGGTTACATTGTGGCTGTTTCTTCTTGAACAATTCTATCTTGGCCTTGGC CGTGTAATGTGAGGTTTACCTGATTCTCTCTAATGAAATAAATACCTAAGTTATTGTTTGAAAAAAAAAAA AAA
Terminology	Object: sequence object Class: an element of the class protein
Features	Attributes: identifier, nucleotide sequence, sequence length • The identifier is a string of alphanumeric characters that can also contain:":", "." and "-" characters. In the example: gi|34783212 • The nucleotide sequence is a string composed only of "A", "C", "G", "T" and "N" characters. • The sequence length is a natural number that shows the number of base pairs in the sequence. In the example: 563 bp
Input	DNA sequence
Output	A specific protein

Table 2
Knowledge elements for the system proposed in Fig. 6.

KB element	Description	Format
NR	Non-redundant protein database	Binary indexed files, created by formatdb algorithm (Altschul et al., 1997) for protein databases
CDD	Conserved domains database	
Ensembl	Model fishes database	
Rules	Collection of rules	Drools

The library *org.biojavax.bio.seq* aims to describe the rich implementations of sequences, locations and features, whereas, *org.biojava.bio.program.gff* is responsible for GFF manipulation. Since these frameworks are very powerful and can provide many functionalities, others libraries will be added in future updates.

6. Conclusion and future works

The technological advances in the last ten years have increased the volume of biological data generated all over the world. Nowadays, the challenge is how to process, analyse, store and use these data in an efficient way. A key example of this situation is the gene annotation process, one of the most difficult tasks of Genomics, which has to identify and characterise the genes in a DNA sequence.

Ideally, the gene annotation process should be carried out by human experts, who evaluate each sequence and, based on their biological and biochemistry knowledge, infers its proper features. However, the huge amount of biological data makes of human intervention a bottleneck in this task. The automatic annotation alleviates this bottleneck, but it is less accurate and reliable than the manual approach. Moreover, it still requires human experts to validate (i.e. *curate*) its results given the potential impact of any misprediction in future research and its applications.

One possible solution is the creation of ES that are capable to emulate the expert reasoning in key steps of the annotation process. The development of ES is a complex task and requires extensive knowledge of the application domain and the expert reasoning process. Nevertheless, there are methodologies for modelling ES, such as CommonKADS, which facilitate the development and can be employed in different scopes.

This work proposes a knowledge model for functional annotation system regardless the DNA sequence context and the structured interview as KE technique. The development of the knowledge model applied the CommonKADS methodology. Through this methodology, a general framework was created, using the classification task as a cornerstone. Due to the high generalisation level achieved, the model designed here can be applied for different kinds of pipelines and any DNA functional annotation task.

Thanks to the efforts carried out in this work, it was possible to verify that the use of CommonKADS can be extended to different bioinformatics problems, such as the classification of sequences from metagenomic data or the prediction of protein three-dimensional structure.

This work also intends to highlight the importance of the application of Knowledge Engineering methodologies in systems development for Bioinformatics. These techniques provide the required tools to organise and structure the knowledge in a way it can be understood by all the people involved in the systems development. The KE techniques have a great value as well. They are a powerful mean to obtain the information that is essential for the comprehension of the applications domain.

With the purpose of evaluating the accuracy of the system, 624 annotated sequences of *Sparus aurata* (sea bream) were retrieved

from Entrez Protein Database (Sayers et al., 2012), translated to nucleotide (through *backtranseq* from Emboss package (Rice, Longden, & Bleasby, 2000)) and processed by the pipeline proposed in Fig. 6. Since NR contains all non-redundant protein data from Entrez Protein and the sequences used to test the performance of the system are from the same source, all sequences from this specie were removed from NR database used in this pipeline. This measure prevented the possibility to have the same sequence in both sets (query and hit) and consequently, guarantees a more realistic result. Ensembl (Flicek et al., 2011) fishes database used in this test has not any sequence from sea bream, and therefore it was not modified. Then, each annotation inferred by the system was compared with the annotation in Entrez. The implemented prototype annotated satisfactorily more than 70% of the sequences.

The work discussed in this paper is part of an ongoing work with several open lines for improvement. Even though the prototype got a high percentage of correct annotations, the number of involved rules and programs still need to be increased in order to create more accurate inferences. As part of the future work, we also intend to extend the implementation of this system to a multi-agent system and model it using MAS-CommonKADS (Iglesias, Garijo, González, & Velasco, 1997). Finally, it is also interesting to get a higher involvement of annotation experts in the development of the system, in particular in the refinement of the knowledge the system uses. Here, improvements in the tools to explain the actual annotation process followed by the system for a sequence, and the interactive participation of experts during the process, can be of great help.

Acknowledgements

This work has been supported by Grant CSD2007-00002 Consolider-Ingenio 2010, MICINN (Spain).

Thanks are due to the anonymous referees for their very valuable comments and suggestions.

References

Akerkar, R., & Sajja, P. (2010). *Knowledge-based systems*. Jones & Bartlett Learning.
Alonso, A., & Guijarro, B. (2004). *Ingeniería del conocimiento. Aspectos Metodológicos* (1 ed.,). Pearson Prentice Hall.
Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., et al. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research, 25*, 3389–3402.
Awad, E. M., & Ghaziri, H. M. (2004). *Knowledge management*. Pearson Education India.
Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., & Sayers, E. W. (2011). GenBank. *Nucleic Acids Research, 39*, 32–37.
Berger, B., & Leighton, T. (1998). Protein folding in the hydrophobic–hydrophilic (HP) is NP-complete. In *Proceedings of the second annual international conference on computational molecular biology. RECOMB '98* (pp. 30–39). New York, NY, USA: ACM.
Bernstein, F. C., Koetzle, T. F., Williams, G. J., Meyer, E. F., Brice, M. D., Rodgers, J. R., et al. (1977). *European Journal of Biochemistry, 80*, 319–324.
Burton, A., Shadbolt, N., Rugg, G., & Hedgecock, A. (1990). The efficacy of knowledge elicitation techniques: A comparison across domains and levels of expertise. *Knowledge Acquisition, 2*, 167–178.
Crescenzi, P., Goldman, D., Papadimitriou, C. H., Piccolboni, A., & Yannakakis, M. (1998). On the complexity of protein folding. In *Proceedings of the second annual international conference on computational molecular biology. RECOMB '98* (pp. 61–62). New York, NY, USA: ACM.
Curwen, V., Eyras, E., Andrews, T. D., Clarke, L., Mongin, E., Searle, S. M., et al. (2004). The ensembl automatic gene annotation system. *Genome Research, 14*, 942–950.
Durbin, R., Haussler, D., Stein, L., Lewis, S., & Krog, A. (2011). *GFF (general feature format) specifications document*. <http://www.sanger.ac.uk/resources/software/gff/spec.html>.
Edwards, D., Stajich, J. E., & Hansen, D. (2009). *Bioinformatics tools and applications*. Springer.
Finn, R., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J., et al. (2010). The Pfam protein families database. *Nucleic Acids Research, 38*, D211–D222.
Flicek, P., Amode, M. R., Barrell, D., Beal, K., Brent, S., Chen, Y., et al. (2011). Ensembl 2011. *Nucleic Acids Research, 39*, D800–D806.
Friedberg, I. (2006). Automated protein function prediction – The genomic challenge. *Briefings in Bioinformatics, 7*, 225–242.
Giarratano, J. C., & Riley, G. (1998). *Expert systems: Principles and programming. Computer science series*. PWS Publishing Company.
Gouret, P., Vitiello, V., Balandraud, N., Gilles, A., Pontarotti, P., & Danchin, E. G. (2005). Figenix: Intelligent automation of genomic annotation: Expertise integration in a new software platform. *BMC Bioinformatics, 6*, 198.
Holland, R. C. G., Down, T. A., Pocock, M., Prlic, A., Huen, D., James, K., et al. (2008). Biojava: An open-source framework for bioinformatics. *Bioinformatics, 24*, 2096–2097.
Hunter, S., Jones, P., Mitchell, A., Apweiler, R., Attwood, T. K., Bateman, A., et al. (2012). InterPro in 2011: New developments in the family and domain prediction database. *Nucleic Acids Research, 40*, D306–D312.
Iglesias, C. A., Garijo, M., González, J. C., Velasco, J. R. (1997). *MAS-CommonKADS: A comprehensive agent-oriented methodology* (an extended version of this paper has been published in the Journal Mathematical Modelling and Scientific Computing, 8).
Keedwell, E., & Narayanan, A. (2005). *Intelligent bioinformatics: The application of artificial intelligence techniques to bioinformatics problems*. Wiley.
Lehninger, A., Nelson, D. L., & Cox, M. M. (2008). *Lehninger principles of biochemistry* (5th ed.). W.H. Freeman.
Marchler-Bauer, A., Lu, S., Anderson, J., Chitsaz, F., Derbyshire, M., Deweese-Scott, C., et al. (2011). Cdd: A conserved domain database for the functional annotation of proteins. *Nucleic Acids Research, 39*, 225–229.
Pajares, G., & Santos, M. (2005). Inteligencia Artificial e Ingeniería del Conocimiento, Ra-Ma, Librería y Editorial Microinformática.
Pevsner, J. (2009). *Bioinformatics and functional genomics*. Wiley Blackwell.
Potter, S. C., Clarke, L., Curwen, V., Keenan, S., Mongin, E., Searle, S. M., et al. (2004). The ensembl analysis pipeline. *Genome Research, 14*, 934–941.
Rice, I., Longden, P., & Bleasby, A. (2000). Emboss: The european molecular biology open software suite. *Trends in Genetics, 16*, 276–277.
Sayers, E. W., Barrett, T., Benson, D. A., Bolton, E., Bryant, S. H., Canese, K., et al. (2012). Database resources of the national center for biotechnology information. *Nucleic Acids Research, 40*, 13–25.
Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Walter, V. d. V., et al. (2000). *Knowledge engineering and management: The CommonKADS methodology*. Cambridge, MA: MIT Press.
Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., et al. (2002). The bioperl toolkit: Perl modules for the life sciences. *Genome Research, 12*, 1611–1618.
Stein, L. D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., et al. (2002). The generic genome browser: A building block for a model organism system database. *Genome Research, 12*, 1599–1610.
The Gene Ontology Consortium, 2000. Gene ontology: Tool for the unification of biology. *Nature Genetics 25*, 25–29.
The JBoss Drools Team. Drools expert user guide, 2012.
UniProt Consortium (2012). Reorganizing the protein space at the universal protein resource (UniProt). *Nucleic Acids Research 40*, D71–D75.

5.2. MASSA: Multi-Agent System to Support Functional Annotation

5.2.1. Citation

Xavier, D., Crespo, B., Fuentes-Fernández, R. and Gómez-Sanz, J. J. Massa: Multi-agent system to support functional annotation. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection* (edited by Y. Demazeau, F. Zambonelli, J. M. Corchado and J. Bajo), vol. 8473 of *Lecture Notes in Computer Science*, pages 291-302. Springer-Verlag, Berlin-Heidelberg, Germany, 2014.

5.2.2. Abstract

Functional annotation aims to predict the biological function of DNA sequences. This complex and time-consuming task has to process huge amounts of data and get high quality results. In order to guarantee the quality of the outcome, the annotation should be carried out by human experts, but the great volume of biological data produced lately demands a high degree of automation. The features of this problem (i.e., knowledge-based, distributed resources, and an evolving environment) make it suitable for an agent approach. This paper presents MASSA, a Multi-Agent System to support functional annotation. MASSA combines the potentialities of the agent approach with a Rule-Based Expert System to reproduce the annotation steps, including the human reasoning, at the inference stage. The expert system integrates knowledge on Biology and tools. A case study on the annotation of sequences of four phylogenetically distinct species illustrates the results and use of MASSA.

5.2.3. References

(Alferes and Pereira, 1996), (Altschul et al., 1997), (Bellifemine et al., 2001), (Benson et al., 2013), (Bodenreider and Stevens, 2006), (Bryson et al., 2001), (Crescenzi et al., 1998b), (Decker et al., 2002), (Giarratano and Riley, 1998), (Gouret et al., 2005), (Lehninger et al., 2008), (Luck and Merelli, 2005), (Magrane and UniProt Consortium, 2011), (Marchler-Bauer et al., 2013), (Möller et al., 1999), (NCBI - Natioanal Center for Biotechnology Information, 2013), (NCBI Resource Coordinators, 2013), (Pavón et al., 2005), (Potter et al., 2004), (Quevillon et al., 2005), (Remm et al., 2001), (Schreiber et al., 2000), (Storm and Sonnhammer, 2002), (The Gene Ontology Consortium, 2000), (The JBoss Drools Team, 2012), (Welcome Trust Sanger Institute, 2012), (Wolstencroft et al., 2013), (Xavier et al., 2013).

MASSA: Multi-Agent System to Support Functional Annotation

Daniela Xavier[1,2], Berta Crespo[3], Rubén Fuentes-Fernández[4], and Jorge J. Gómez-Sanz[4]

[1] Dept. of Biochemistry and Molecular Biology I, Universidad Complutense de Madrid, Avd. Complutense s/n, 28040 Madrid, Spain
[2] GARP (Genomic and RNA Profiling Core), Department of Molecular and Human Genetics, Baylor College of Medicine, One Baylor Plaza, 77030 Houston, USA
[3] Dept. of Fish Physiology and Biotechnology, Instituto de Acuicultura de Torre la Sal, Consejo Superior de Investigaciones Científicas (CSIC), Torre la Sal, Ribera de Cabanes s/n, 12595 Castellón, Spain
[4] Dept. of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, C/ Profesor José García Santesmases s/n, 28040 Madrid, Spain

Abstract. Functional annotation aims to predict the biological function of DNA sequences. This complex and time-consuming task has to process huge amounts of data and get high quality results. In order to guarantee the quality of the outcome, the annotation should be carried out by human experts, but the great volume of biological data produced lately demands a high degree of automation. The features of this problem (i.e., knowledge-based, distributed resources, and an evolving environment) make it suitable for an agent approach. This paper presents MASSA, a Multi-Agent System to support functional annotation. MASSA combines the potentialities of the agent approach with a Rule-Based Expert System to reproduce the annotation steps, including the human reasoning, at the inference stage. The expert system integrates knowledge on Biology and tools. A case study on the annotation of sequences of four phylogenetically distinct species illustrates the results and use of MASSA.

Keywords: Functional annotation, Multi-Agent System, Rule-based Expert System, Bioinformatics.

1 Introduction

Predicting the biological function of Deoxyribonucleic Acid (DNA) sequences is one of the many challenges Bioinformatics faces. This task, called *functional annotation*, has to be as accurate and reliable as possible due to its impact in further researches [11]. In order to guarantee the quality of the annotation, experts should manually annotate each sequence. However, the great volume of genomic data generated lately makes this practice only suitable for few sequences or model organisms. The automatic annotation, on the other hand, rapidly processes big data sets at low cost, but produces less accurate results.

Y. Demazeau et al. (Eds.): PAAMS 2014, LNAI 8473, pp. 291–302, 2014.

The annotation process involves many tasks, such as, comparing sequences, accessing information resources, and inferring the function. Though there is a variety of tools that supports experts in these tasks, they present some limitations. First, some tools (or their outcome) are not quite intuitive for the final users. Second, they are in general standalone programs, so users have to combine their results manually. Third, they encapsulate knowledge in the code of their components, what hinders the expert involvement on creating and validating it. Finally, these tools are not designed in general to evolve, though this should be a core feature given their domain. For instance, the majority of them use biological information stored in multiple and heterogeneous databases, which are distributed and constantly being updated. Their proper maintenance largely depends on being able to integrate easily new or modified information sources.

In order to address these issues, some works have considered the combined use of Expert Systems (ESs) and Multi-Agent Systems (MASs). On one side, ESs [9] are a well-known approach to avoid the expert bottleneck when automating processes. In particular, Rule-Based ESs (RBESs) are suitable to deal with factual and heuristic knowledge, like that used at the inference stage of functional annotation [28]. On the other side, agents have proved to be useful for applications that imply repetitive and time-consuming activities, and also require knowledge management, such as integrating multiple information sources and tools, and modeling complex dynamic systems [12].

Nevertheless, this last group of systems also presents its own open issues. Most of them [6,8] still encapsulate relevant parts of knowledge in code. Besides, the applied knowledge is mainly related to the flow of data between basic tools [6,8], with only some rules dealing with expert heuristics [15]. This puts aside the core of the usual process of experts, which is on biological constraints and relationships. Finally, they seem not to apply well-founded methodological approaches in their development, at least according to information in literature.

This paper presents MASSA (*MAS to Support functional Annotation*), which overcomes some of the previous limitations. This is achieved through a design focused on an expert-oriented management of knowledge and facilitating evolution and maintenance. MASSA combines an agent-oriented approach with RBESs to infer accurate annotations and being also able to take advantage of distributed computational resources, collect data from different sources, and maintain its data sources up to date. The work applies two state-of-the-art methodologies: INGENIAS [18] for the MAS; and CommonKADS [22] for modeling the knowledge employed for the RBES, as described in [28].

MASSA includes two main subsystems. *MASSAPipe* manages a flexible pipeline of traditional Bioinformatics tools and databases in order to collect the basic information for the process. *MASSAInference* integrates the RBES that makes the inference applying knowledge on Biology and Bioinformatics tools.

The rest of the paper discusses these aspects in detail. Section 2 introduces briefly the annotation problem. MASSA is presented in Section 3, and Section 4 describes its functioning and performance for a set of sequences. Section 5 reviews the related work. Finally, Section 6 discusses some conclusions and future work.

2 Biological Background

In most organisms, the hereditary information is stored in macromolecules called Deoxyribonucleic Acid (DNA). *Genes* are segments of DNA that are responsible for the transmission of genetic traits from an organism to its descendants. Genes can code polypeptide molecules called *proteins*.

A protein can be understood as a chain of amino acids (i.e., *primary structure*) that folds into itself, creating two-dimension structures (i.e., *secondary structure*) such as turns, helixes, or sheets. These elements are packed in the space into compact globular units, forming the *tertiary structure*.

The protein's role is directly related to its tertiary structure. Some regions of the protein's primary structure may vary substantially without affecting its biological function. However, some regions are crucial for the protein's function and preserved over evolutionary time, like *domains* and *conserved sites*.

A protein *family* is a set of proteins that share an evolutionary relationship and have a significant similarity in primary structure and/or with similar tertiary structure and function. The members of a protein family are called *homologs* and are usually identical across a 25% or more of their sequences. Two *homologs* are said to be *paralogs* if they are found in the same species, or *orthologs* if they belong to different species.

The *functional annotation* aims to predict the protein's function of a given sequence. This prediction can be done from the tertiary structure, but this involves time-consuming, labor-intensive, and expensive processes, which are only affordable for few sequences. The function can also be obtained from the primary structure, but it is a NP-complete problem [7], and hence frequently unfeasible computationally. An alternative method is taking advantage of evolutionary relationships. Orthologs often preserve their biological role, and thus identifying them allows transferring functional information between genes from different organisms with a high degree of reliability [21]. Since finding orthology is not a trivial task, its prediction can be complemented with other features, such as conserved domains and residues, to enhance the quality of the annotation.

Many tools support the different steps of the annotation process. BLAST [2] (and its many derivatives such as BLASTP, BLASTX, TBLASTX, BLASTN, and RPS-BLAST) is one of the most popular. It is used to look for regions of local similarity between a query sequence and a target dataset. Popular tools are also those to recognize domains (e.g., InterProScan [20]), and to predict orthology (e.g., Orthostrapper [23]).

There is a great amount of genetic information publicly available that can be used in the annotation process. There are databases for genes (e.g., GenBank [4]), protein data (e.g., Entrez Protein [17] and UniProt [13]), domains and families (e.g., Conserved Domains Database (CDD) [14]), and for ontologies (e.g., GO [24]). These databases are updated regularly, and some of them make available for download pre-formatted Search Databases (SDBs) ready to use with BLAST, like Non-Redundant proteins (NR) and CDD, both from the National Center for Biotechnology Information (NCBI) [16].

3 MASSA Architecture

MASSA combines a MAS with a RBES to generate accurate functional annotations and overcome domain constraints. These include: evolving and heterogeneous knowledge to consider; multiple tools to combine without standardized common interfaces; distributed data sources frequently updated and with different structures; heavy computation tasks and queries that frequently make use of shared computational resources. MASSA is organized in subsystems that can handle simultaneously multiple user requests (see Fig. 1).

An annotation request to MASSA comprises a FASTA file and, optionally, a Configuration File (CF). FASTA [16] is a standard text-based format to specify nucleotide or peptide sequences. The CF contains a list of *tasks* and their parameters to be used in the pipeline. Experts usually include this information to adjust the process.

Each request is linked to a *project*, which has assigned its own execution *environment*. The *environment* includes a *container* for the group of agents working in it, and an outcome database for intermediate results, named the Result Database (RDB). The RDB stores the information on annotation candidates, which will be used later in the inference process. The system divides each request in smaller chunks, called *tasks*, in order to parallelize as much as possible its processing. A *task* can involve the execution of one or more tools, scripts, or complementary jobs, depending on its goal. In the case that a *task* executes more than one step, it can be decomposed into *subtasks*.

The structure of MASSA is mainly organized around two subsystems, *MASSAPipe* and *MASSAInference*, and several agents that provide shared services. MASSAPipe aims to execute an annotation pipeline of traditional Bioinformatics tools. MASSAInference infers the best annotation based on the data previously acquired and the set of rules in its Knowledge Base (KB). Although these subsystems work together in MASSA, their modular design makes possible to use them separately. The other agents are related to standard services (e.g., lifecycle management and yellow pages), user interface, and work coordination.

MASSA start-up initializes only three agent instances: an *Interface Agent* (IA) to manage the user interface; a *Launcher Agent* (LA) to launch projects; and an *External Information Updater Agent* (EInfoUA) to update files from external sources. The IA receives user requests and passes them to the LA. Then, the LA creates a *Controller Agent* (CA) for each request. The CA is responsible for managing global aspects of the request *project* and coordinating the other agents in MASSA working for it, including the communications with the IA. The CA decomposes the work in the *project* and delegates it to the relevant agents. First, agents in MASSAPipe gather the required information, and when they finish, the CA uploads the resulting information to the RDB. After that, agents in MASSAInference calculate their annotation, that the IA returns as the answer to the request.

MASSA is mainly implemented in Java, using Jade [3] for the MAS and Drools [25] for the RBES. The system also contains scripts in Perl to manipulate Bioinformatics-specific information, and uses MySQL for database management.

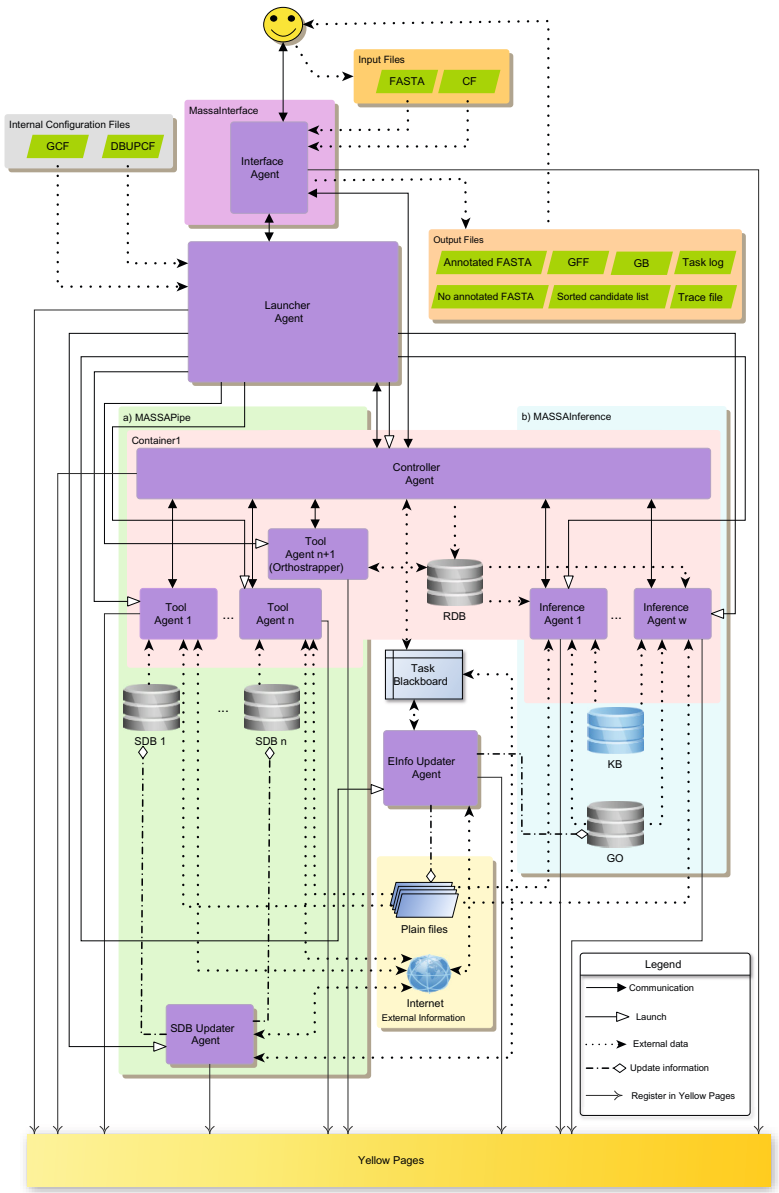


Fig. 1. MASSA architecture

The rest of the section further discusses the two MASSA subsystems: MASSAPipe (see Section 3.1) and MASSAInference (see Section 3.2).

3.1 MASSAPipe

MASSAPipe manages the annotation pipeline. It executes tools and scripts that search in SDBs to get likely candidates for the annotation. It is also responsible for storing the information obtained and updating some of the default SDBs and files from their original sources.

MASSAPipe includes three types of agents:

- The *Tool Agent* (TA) wraps tools and scripts. It also accesses remote databases (e.g., Entrez Protein [17] and CDD [14]) and external files in order to get relevant information that may increase the accuracy of the annotation.
- The *Search Database Updater Agent* (SDBUA) updates the SDBs from its source remote servers. It controls resource usage for these tasks, the availability of the remote sources, and performs error recovery.

The activities of the MASSAPipe subsystem start when the LA creates the CA and its *environment* to meet a request. This includes creating a RDB, where the *task* outcomes will be stored, and launching TAs for the *tasks* in the request, though limited by the system workload and resources.

The CA checks whether all input files specified in the CF are available. Then, it processes the *tasks* to perform. These *tasks* are mainly searches in SDBs through tools like BLAST [2] or InterProScan [20]. As they are independent, the CA processes them in parallel. Each *task* is decomposed into *subtasks* that are stored in a queue and assigned to the available TAs. The TAs gather their information on candidate annotations as files in the GFF format [26], and after completing the assigned *task* they remove themselves from the *container*.

When the TAs complete the search *tasks*, the CA uploads the resulting files into the RDB of the *project*. Then, it sends a message to the TA responsible for executing Orthostrapper [23]. This TA uses some of the RDB information to predict orthology and updates the RDB with the obtained outcome. After that, the IA informs the user that this step is finished. The CA, in turn, changes the status of the job to "Finished", and tells the last TA to finish itself.

3.2 MASSAInference

MASSAInference is the part of the system that assigns automatically functional annotations. Its key agent is the *Inference Agent* (InfA), which manages the RBES. The InfA provides the rule engine with data retrieved from the RDB on candidate annotations for a query sequence. It also accesses local information (files and a database) to get the GO terms used to enrich the inferred annotation.

The KB contains rules that take into account different candidate features. The rules score and assess these features, and sort the candidates. The best situation happens when the query sequence present similarities with data from protein

databases (i.e., there are similar homolog candidates), along with additional indicators such as orthology likelihood and domain information. If there is no match to homolog candidates, the system tries to infer the annotation based on domain alignments. When this also fails, the sequence is added to a "Not annotated" list. The resulting sorted list starts with the candidate that has the most likely and informative annotation.

This subsystem starts working when the CA is notified that all the agents in the pipeline have finished. Then, it notifies the LA that it can proceed to the next step, and the LA launches as many InfAs as defined in the CF. The CA divides the work between the InfAs based on the number of query sequences and InfAs available. An InfA queries the RDB one sequence at time, obtaining the features of the candidate, and processing them according to the rules in the KB. Based on these rules, the candidate lists are created. When an InfA completes its *task*, it notifies that to the CA.

As InfAs finish their *tasks*, the CA asks them to self-destroy. When all InfAs have finished, the CA informs the IA that the annotation is done, and removes its *container* (together with itself) from the platform. Then, the IA notifies the user and makes the outputs available, mainly the sorted list of candidate annotations.

MASSA was designed to be able to evolve. Its modular and flexible architecture, together with its well-supported base frameworks, make this goal feasible. Adding a new tool to the pipeline is straightforward. It just requires describing the new task (i.e., a new Java class for it) and adding in the CA the code to ask for its execution. If the task introduces a new feature for the annotation process, the InfA has also to be programmed to deal with it, and new rules have to be defined. The later can be done just by modifying the rules flat file of the KB.

4 Case Study

With the purpose of testing the system, 2128 annotated sequences from four phylogenetically distinct species - *Homo sapiens* (532), *Gallus gallus* (596), *Drosophila melanogaster* (500), and *Xenopus tropicalis* (500) - were submitted to the system. Seven *tasks* were executed: FASTA to GFF, BLASTX against NR, BLASTX against UniProt, InterProScan, RPS-BLAST against CDD, and Orthostrapper. Parameters like the e-value threshold ($<= 1.0E^{-20}$ for BLASTX and $<= 1.0E^{-05}$ for RPS-BLAST) and number of InfA to perform the prediction (6) were also set.

After getting the input, the IA sends a message containing this information to the LA. The LA reads the message, and creates the container (*Container-1*) and its respective RDB (*massa_Container_1*). Based on the message, the LA launches on *Container-1* a CA and seven TAs (one for each task). Then, the LA forwards the message received to the CA and informs this agent that it can start the MASSAPipe workflow.

The CA checks the existence of the input files and splits each *task* into *subtasks*. For instance, the RPS-BLAST *task* is divided into a list of *subtasks*: *FastaTranslate*, *RPS-BLAST*, and *BLAST2GFF*. This *task* is carried out as follows.

At first, the CA informs the TA *Container-1_rpsBlast_1*, in charge of the RPS-BLAST *task*, that it should execute the *FastaTranslate subtask*. After finishing this *subtask*, the TA sends a message to the CA reporting the completed status of the *subtask*, and waits for the next command. The CA, in turn, checks the *subtask* list and assigns the *RPS-BLAST subtask* to the *Container-1_rpsBlast_1* TA. When this job is done, this agent informs the CA, which sends it the last *subtask*, *BLAST2GFF*. While performing it, the agent accesses online information from CDD and the local GO data in order to improve the quality of the information. At the GFF formatting stage, just target sequences with "informative" annotations are included into the GFF, that is, terms like "unnamed protein" or "unknown domain" are ignored. After completing the last *subtask*, the TA sends a message to the CA, which asks it to remove itself from the container. This procedure is done for all defined *tasks* (i.e., FASTA to GFF, BLASTX against NR, BLASTX against UniProt, InterProScan, and RPS-BLAST against CDD) in parallel, except for Orthostrapper.

When all the parallelizable *tasks* are done, the CA uploads the GFF files generated into the *massa_Container_1 database*. Once the data transference is completed, the CA informs the TA responsible for the Orthostrapper *task* (i.e., *Container-1_orthology*) it can start. *Container-1_orthology* accesses the RDB in order to get the information to accomplish its goal, and then uploads the result obtained into the RDB. After that, this agent sends a message to the CA informing its job is done and leaves the container. The CA, in turn, informs the IA and the LA that MASSAPipe has finished, and the IA forwards this message to the user. The LA launches four InfAs, and sends a message to the CA telling it can start MASSAInference.

The CA queries the RBD to get the number of query sequences to be annotated (e.g., 532 for *Homo sapiens*), divides the work based on the InfA number set in the CF (i.e., 6 InfAs) and sends a message to each InfA with the range of sequences they have to annotate. For example, *Container-1_Inference_1* is in charge of the first 90 sequences, *Container-1_Inference_2* annotates the next 90, and so on. An InfA infers one annotation at time, but all InfAs work in parallel.

The annotation inference is performed based on the rules described in the KB. These rules take into consideration the orthology, the domains found in the sequence, the conserved sites, the existence of GO terms, the bit score, the e-value, and the percentage of identity. The best-case scenario uses all these features to infer an annotation.

The results from MASSA were manually compared with the original ones by an expert, and 93.7% of the sequences were predicted correctly using the homology candidate approach. 0.28% of the sequences were annotated only with domain information, and 0.47% of the sequences could not be annotated. The rest of the sequences, 5.55%, was not satisfactorily annotated. This issue can be caused by sequences that are not correctly annotated or have questionable annotations because of the lack of consensus in the biological community. These results are promising according to experts, but additional comparison with human experts and tools is required.

5 Related Work

Nowadays, a range of tools that support the functional annotation process are publicly available. In general, these tools are standalone programs that do not communicate with each other, what encumbers the whole process. In order to overcome this hurdle, systems that integrate some of these tools have been developed using different approaches.

Systems like the Ensembl Analysis Pipeline (EAP) [19] and FIGENIX [10] are RBEs developed to accomplish the functional prediction. Although they are quite successful in this task, they constrain users because of their design and considered requirements. For instance, EAP can only deal with complete genomes, thus it is not suitable for DNA sequences out of this context. FIGENIX is only available through a Web service, presenting all the limitations related to this approach (e.g., applicable tools, parameters, and databases, and small input size), which precludes expert users from taking advantage of all their expertise.

More complex systems that combine MASs and RBEs have also been developed, though this approach is less popular. Examples of them are GeneWeaver [6], BioMas [8], and EDITtoTrEMBL [15]. They are mainly focused on wrapping a variety of tools and databases, but pay less attention to develop ESs that integrate knowledge. Their ESs are more related to managing the tool pipeline than to biological issues. Moreover, some of these systems do not integrate true ESs, but components that apply expert knowledge. For instance, BioMas includes an algorithm for deducing appropriate electronic GO annotations by mapping terms from different ontologies [8]. However, this knowledge is hard-coded in a component and not available in a KB as in true ESs [9].

Another issue is the use of infrastructures with limited support, or even developed *ad-hoc* for a particular system. For instance, EDITtoTrEMBL integrates a RBES based on logic programming with Well-Founded Semantics eXtended for explicit negation (WFSX) [1]. This is a less extended formalism than those present in, for instance, Drools [25] or the Semantic Web Rule Language (SWRL) [5], which have bigger communities supporting them. This support brings important benefits regarding development, maintenance, and extension of tools. Nevertheless, there are not studies evaluating whether experts have more or less difficulties to work with different formalisms, so the choice of a suitable one remains an open issue.

Finally, there are also methodological aspects. It is well-known that systematic approaches from Software and Knowledge Engineering facilitate the development of complex systems, but the literature does not document their application for the aforementioned systems. The lack of engineering methodologies does not only affect the development of systems, but also the repeatability, understanding, and analysis of these processes, as well as their functionalities and outcomes.

6 Conclusions

This work presents MASSA, a MAS with a RBES for functional annotation. It addresses three key problems of current annotation tools. Firstly, it uses the

RBES to mimic expert reasoning at certain points of the process, which allows generating more precise outcomes and reducing expert workload. Secondly, the explicit and declarative representation of knowledge as rules facilitates a greater involvement of experts in their specification and validation. Thirdly, applying the agent paradigm facilitates overtaking the environment hurdles of this problem (i.e., distribution, heterogeneity, and high pace of evolution in tools and data sources), and integrating the knowledge management.

MASSA does not only facilitate the annotation process, but also presents other remarkable features to boost and improve the process. It is able to deal with different databases, maintain the data up to date, take advantage of the available computational resources, and report all the reasoning process applied to come to the annotation. Regarding knowledge, it takes into consideration, among others, orthology, existence of domains, conserved sites, level of relevance of the annotation, and GO terms. All these features aggregate quality to the prediction. MASSA was able to produce accurate annotations for 93.7% of the 2128 sequences tested, what is a very encouraging result.

As far as we can ascertain, this approach has not been widely used to tackle this problem, since most of the MAS annotators developed to date lack of ESs. Also, some of the considered features appear in previous systems, but they do not do it in an integrated way. Moreover, the combination of MASs and RBESs seems to be quite suitable and advantageous for several Bioinformatics problems. Therefore, this work does not only intend to propose a possible solution to the functional annotation problem, but also to encourage the application of similar strategies in this field.

MASSA is ongoing work with several open lines for improvement. The system still has to be tested and assessed more extensively for other sequences and species. Besides, more complex performance tests should be carried out as well. This will be facilitated by making the system, its code, and results publicly available for the community. MASSA also needs to incorporate support for additional resources, such as methods for identifying conserved residues that affect function and for predicting transmembrane regions. In line with this, new rules will be added to the KB. These will allow representing the expert knowledge still missing regarding the annotation process in general, and also integrating properly the new resources. Another improvement could come from specializing the InfAs in different subtypes that work with different KBs representing the perspectives of multiple human experts. This differentiation would also require setting up some negotiation mechanism among agents that allow them arriving to a common (or at least most recommended) annotation. Regarding the system interface, we intend to follow the workflow management system trend, like in [27], to allow the user easily define the pipeline for each project.

Acknowledgments. This work has been done in the context of the projects "Aquagenomics" (CSD2007-00002) funded by the Consolider-Ingenio 2010 program of the Spanish Ministry of Education and Science, and "Social Ambient Assisting Living - Methods (SociAAL)" (TIN2011-28335-C02-01) supported

by the Spanish Ministry for Economy and Competitiveness. Also, we acknowledge support from the "Red Científico-Tecnológica en Ciencias de los Servicios" (TIN2011-15497-E) and the "Programa de Creación y Consolidación de Grupos de Investigación" (UCM-BSCH GR35/10-A).

References

1. Alferes, J.J., Pereira, L.M.: Reasoning with Logic Programming. LNCS, vol. 1111. Springer, Heidelberg (1996)
2. Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research 25(17), 3389–3402 (1997)
3. Bellifemine, F., Poggi, A., Rimassa, G.: JADE: a FIPA2000 compliant agent development environment. In: Proceedings of the 5th International Conference on Autonomous Agents (AGENTS 2001), pp. 216–217. ACM (2001)
4. Benson, D.A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Sayers, E.W.: GenBank. Nucleic Acids Research 41(D1), D36–D42 (2013)
5. Bodenreider, O., Stevens, R.: Bio-ontologies: current trends and future directions. Briefings in Bioinformatics 7(3), 256–274 (2006)
6. Bryson, K., Luck, M., Joy, M., Jones, D.T.: Agent interaction for Bioinformatics data management. Applied Artificial Intelligence 15(10), 917–947 (2001)
7. Crescenzi, P., Goldman, D., Papadimitriou, C.H., Piccolboni, A., Yannakakis, M.: On the complexity of protein folding. In: 2nd Annual International Conference on Computational Molecular Biology (RECOMB 1998), pp. 61–62. ACM (1998)
8. Decker, K., Khan, S., Schmidt, C., Situ, G., Makkena, R., Michaud, D.: BioMAS: a multi-agent system for genomic annotation. International Journal of Cooperative Information Systems 11(03n04), 265–292 (2002)
9. Giarratano, J.C., Riley, G.: Expert Systems: Principles and Programming. Computer Science Series. PWS Publishing Company (1998)
10. Gouret, P., Vitiello, V., Balandraud, N., Gilles, A., Pontarotti, P., Danchin, E.G.: FIGENIX: Intelligent automation of genomic annotation: expertise integration in a new software platform. BMC Bioinformatics 6, 198 (2005), `http://www.ncbi.nlm.nih.gov/pubmed/16083500`
11. Lehninger, A.L., Nelson, D.L., Cox, M.M.: Lehninger Principles of Biochemistry, 5th edn. W. H. Freeman & Company (2008)
12. Luck, M., Merelli, E.: Agents in Bioinformatics. Knowledge Engineering Review 20(2), 117–125 (2005)
13. Magrane, M., UniProt Consortium: UniProt knowledgebase: a hub of integrated protein data. Database 2011, bar009 (2011)
14. Marchler-Bauer, A., Lu, S., Anderson, J.B., Chitsaz, F., Derbyshire, M.K., DeWeese-Scott, C., Fong, J.H., Geer, L.Y., Geer, R.C., Gonzales, N.R., Gwadz, M., Hurwitz, D.I., Jackson, J.D., Ke, Z., Lanczycki, C.J., Lu, F., Marchler, G.H., Mullokandov, M., Omelchenko, M.V., Robertson, C.L., Song, J.S., Thanki, N., Yamashita, R.A., Zhang, D., Zhang, N., Zheng, C., Bryant, S.H.: CDD: conserved domains and protein three-dimensional structure. Nucleic Acids Research 41(D1), D348–D352 (2013)
15. Möller, S., Leser, U., Fleischmann, W., Apweiler, R.: EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. Bioinformatics 15(3), 219–227 (1999)

16. NCBI: http://www.ncbi.nlm.nih.gov, (accessed December 20, 2013)
17. NCBI Resource Coordinators: Database resources of the National Center for Biotechnology Information. Nucleic Acids Research 41(D1), D8–D20 (2013)
18. Pavón, J., Gómez-Sanz, J.J., Fuentes, R.: The INGENIAS methodology and tools. In: Agent-Oriented Methodologies, pp. 236–276. Idea Group Publishing (2005)
19. Potter, S.C., Clarke, L., Curwen, V., Keenan, S., Mongin, E., Searle, S.M.J., Stabenau, A., Storey, R., Clamp, M.: The Ensembl Analysis Pipeline. Genome Research 14(5), 934–941 (2004)
20. Quevillon, E., Silventoinen, V., Pillai, S., Harte, N., Mulder, N., Apweiler, R., Lopez, R.: InterProScan: protein domains identifier. Nucleic Acids Research 33(2), W116–W120 (2005)
21. Remm, M., Storm, C.E.V., Sonnhammer, E.L.L.: Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. Journal of Molecular Biology 314(5), 1041–1052 (2001)
22. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., Van de Velde, W., Wielinga, B.: Knowledge Engineering and Management – The CommonKADS Methodology. MIT Press, Cambridge (2000)
23. Storm, C.E.V., Sonnhammer, E.L.L.: Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. Bioinformatics 18(1), 92–99 (2002)
24. The Gene Ontology Consortium: Gene Ontology: tool for the unification of Biology. Nature Genetics 25(1), 25–29 (2000)
25. The JBoss Drools Team: Drools expert user guide (May 2012), http://docs.jboss.org/drools/release/5.4.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf (accessed December 20, 2013)
26. Welcome Trust Sanger Institute: GFF (General Feature Format) specifications document. (December 2012), http://www.sanger.ac.uk/resources/software/gff/spec.html (accessed December 20, 2013)
27. Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., de la Hidalga, A.N., Balcazar Vargas, M.P., Sufi, S., Goble, C.: The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. Nucleic Acids Research 41(W1), W557–W561 (2013)
28. Xavier, D., Morán, F., Fuentes-Fernández, R., Pajares, G.: Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology. Expert Systems with Applications 40(10), 3943–3952 (2013)

5.3. A rule-based expert system for inferring functional annotation

5.3.1. Citation

Xavier, D., Crespo, B. and Fuentes-Fernández, R. A rule-based expert system for inferring functional annotation. *Applied Soft Computing*, vol. 35, pages 373-385, 2015.

5.3.2. Abstract

Functional annotation is the process that assigns a biological functionality to a Deoxyribonucleic Acid (DNA) sequence. It requires searching in huge data sets for candidates, and inferring the most appropriate features based on the information found and expert knowledge. When humans perform most of these tasks, results are of a high quality, but there is a bottleneck in processing; when experts are largely replaced by automated tools, annotation is faster but of poorer quality. Combining the automatic annotation with Expert Systems (ESs) can enhance the quality of the annotation, while effectively reducing experts'workload. This paper presents INFAES, a Rule-Based ES developed for mimicking the human reasoning in the inference stage of the functional annotation. It integrates knowledge on Biology and heuristics about the use of Bioinformatics tools. Its development adopts state-of-the-art methodologies to facilitate the acquisition and integration of new knowledge. INFAES showed a high performance when compared to the systems developed for the first large-scale community-based Critical Assessment of protein Function Annotation (CAFA) (Radivojac et al., 2013).

5.3.3. References

(Radivojac et al., 2013), (Potter et al., 2004), (Gouret et al., 2005), (Domselaar et al., 2005), (Chen et al., 2012), (Aniba et al., 2009), (Hayes-Roth, 1985), (Studer et al., 1998), (Zacharias, 2008), (Weiss, 1999), (Xavier et al., 2013), (Schreiber et al., 2000), (The JBoss Drools Team, 2012), (Remm et al., 2001), (Sali et al., 2003), (Crescenzi et al., 1998a), (Lehninger et al., 2008), (Lee et al., 2007), (Li and Homer, 2010), (Altschul et al., 1997), (Rust et al., 2002), (Storm and Sonnhammer, 2002), (Jones et al., 2014), (Flicek et al., 2014), (Benson et al., 2013), (NCBI Resource Coordinators, 2013), (The UniProt Consortium, 2014), (Marchler-Bauer et al., 2013), (Hunter et al., 2012), (Finn et al., 2014), (Letunic et al., 2012a), (Haft et al., 2013), (Kahn et al., 2008), (Mi et al., 2013), (Ashburner et al., 2000), (Pavón et al., 2005), (Bellifemine et al., 2001), (Pontius et al., 2003), (NCBI - National Center for Biotechnology Information, 2013), (Welcome Trust Sanger Institute, 2012), (NCBI, 2007), (Cozzetto et al., 2013), (Falda et al., 2012), (Rice et al., 2000), (Koski et al., 2005), (Azé et al., 2008), (Cadag et al., 2007), (Conesa et al., 2005), (Bryson et al., 2001), (Decker et al., 2002), (Möller et al., 2001), (Punta et al., 2012), (Brachman and Levesque, 2004), (Andrade et al., 1999), (García-Magariño et al., 2009), (Kumar et al., 2009), (Krogh et al., 2001), (Petersen et al., 2011).

Applied Soft Computing 35 (2015) 373–385

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A rule-based expert system for inferring functional annotation

Daniela Xavier[a,*,1], Berta Crespo[b], Rubén Fuentes-Fernández[c]

[a] GARP (Genomic and RNA Profiling Core), Department of Molecular and Human Genetics, Baylor College of Medicine, One Baylor Plaza, 77030 Houston, USA
[b] Department of Fish Physiology and Biotechnology, Instituto de Acuicultura de Torre la Sal, Consejo Superior de Investigaciones Científicas (CSIC), Torre la Sal, C/ Ribera de Cabanes s/n, 12595 Castellón, Spain
[c] Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, C/ Profesor José García Santesmases 9, 28040 Madrid, Spain

ARTICLE INFO

Article history:
Received 23 January 2014
Received in revised form
18 December 2014
Accepted 9 May 2015
Available online 3 July 2015

Keywords:
Functional annotation
Rule-based expert system
Bioinformatics

ABSTRACT

Functional annotation is the process that assigns a biological functionality to a deoxyribonucleic acid (DNA) sequence. It requires searching in huge data sets for candidates, and inferring the most appropriate features based on the information found and expert knowledge. When humans perform most of these tasks, results are of a high quality, but there is a bottleneck in processing; when experts are largely replaced by automated tools, annotation is faster but of poorer quality. Combining the automatic annotation with expert systems (ESs) can enhance the quality of the annotation, while effectively reducing experts' workload. This paper presents INFAES, a rule-based ES developed for mimicking the human reasoning in the inference stage of the functional annotation. It integrates knowledge on Biology and heuristics about the use of Bioinformatics tools. Its development adopts state-of-the-art methodologies to facilitate the acquisition and integration of new knowledge. INFAES showed a high performance when compared to the systems developed for the first large-scale community-based critical assessment of protein function annotation (CAFA) [1].

1. Introduction

One of the most challenging tasks of Genomics is predicting the biological function of deoxyribonucleic acid (DNA) sequences, a procedure called *functional annotation*. Its outcome has to be as reliable and accurate as possible, as it will be used in further researches, including to predict new annotations.

The functional annotation is currently a complex, labor-intensive, and time-consuming task for experts. It requires a high degree of expertise to use the proper tools, algorithms, and databases in order to collect relevant information, and to make the pertinent decisions. The amount of genomic data that has been produced, especially in the last years, makes this *manual approach* feasible just for small data sets or reference genomes. Besides, it may produce conflicting interpretations of the analysis [2].

The alternative *automatic approach* uses tools able to process large volumes of data consistently without (almost any) user intervention. Its main drawback is that tools only use limited expert knowledge, so their results are less precise than those of human experts. For instance, only a few tools, like Figenix [3], take orthology knowledge into consideration, what increases the reliability of the annotation. Moreover, tools usually lack the flexibility to adapt to different needs and an ever evolving environment. For example, many of them restrict the kind of query sequences they support (e.g., sequences inside the genomic context [2] or bacterial sequences [4]), and they integrate only a limited and fixed set of data sources to search [5].

A possible way to preserve the quality of the manual annotation without running into its drawbacks is applying expert systems (ESs) to emulate the expert reasoning in certain parts of the process. Among the variety of ESs for annotation [6], rule-based ones [7] are particularly well-suited because of several reasons. First, rules are a natural way of representing knowledge about procedures and heuristics [7], as that applied to a large extent in functional annotation. Second, there are multiple knowledge elicitation techniques [8] to guide rule specification with experts. Third, since rules are more easily understandable by experts than code, their usage promotes system evolution through user involvement [9].

Despite these advantages, existing rule-based ESs (RBESs) for annotation present several issues. Their development is not usually related to standard good practices. Literature does not report

* Corresponding author. Tel.: +34 91 394 7548.
E-mail addresses: xavier@bcm.edu (D. Xavier), ruben@fdi.ucm.es (R. Fuentes-Fernández).
[1] This work was partially done while working at the Department of Biochemistry and Molecular Biology I, Universidad Complutense de Madrid, Spain.

http://dx.doi.org/10.1016/j.asoc.2015.05.055
1568-4946/

the application of any engineering methodology, so key decisions are neither explained nor documented. Moreover, they frequently rely on ad-hoc technologies poorly supported. For instance, there is no documentation on the development approach of the Ensembl Analysis Pipeline (EAP) [2], and it uses its own inference engine. This way of working makes systems difficult to maintain and evolve. Regarding the traditional limitations of annotation tools previously mentioned, RBESs facilitate their solving, but designers need to address them explicitly. For instance, EAP [2] and Figenix [3] are designed to integrate other tools, but only considering dataflow management. There are no guidelines, either general for RBESs or particular for these systems, on how to integrate the new tools in their ESs regarding knowledge, so this integration relies on designers' expertise.

To address these issues, this work proposes a RBES for inferring the functional annotation of DNA sequences called INFAES. INFAES is part of a wider research project called MASSA, a multiagent system (MAS) to Support functional Annotation. This MAS is a community of Intelligent Agents (IAs) [10] that work together. They implement a flexible pipeline of Bioinformatics tools that collects candidates and clues for the prediction task. Then, INFAES uses this information to evaluate the candidates and infer the most likely function.

Although there are already some ESs and RBESs for this task, INFAES was specifically developed to overcome several of their limitations. In particular, it provides an integration of knowledge and analyses previously scattered among different tools, and mechanisms (i.e., an architecture and development guidelines) to facilitate further evolution of the system in order to keep it up to date with emerging research.

As for the annotation process, INFAES is capable of assigning accurate functional annotations to DNA sequences regardless of the species, and whether they are or not complete genomes. Moreover, INFAES rules comprise knowledge that other systems do not consider, what increases the annotation effectiveness. Its rules are able to mine additional data related to the information from the pipeline, and compare the candidate annotations to come to a conclusion. These comparisons apply heuristics that integrate analysis variables from the pipeline tools (e.g., e-value, bit score, identity, and homology likelihood), and Biological knowledge (e.g., the orthology relationship between sequences, the domains and families, and the level of conservation of important sites). This knowledge has been extracted from several sources [11], and pursues modeling the Biologist expertise at the inference stage. These biological concepts are explained later in Section 2.1.

Since INFAES has a special focus on evolution, its architecture and development consider requirements for maintenance. These have not been explicitly taken into account in related works, but they must be in order to keep tools up to date in a domain with a fast changing pace.

INFAES knowledge is structured around the computation of scores and their interpretation. This facilitates considering new knowledge. It appears as new sets of rule to compute additional scores, that specific rules combine with existing ones.

Addressing evolution is not only an issue of system design. The development process also has to consider it. INFAES improves this aspect compared to existing tools for annotation. It follows CommonKADS [12], a well-known methodology for ESs, to build its Knowledge Base (KB) and document the process [11]. Moreover, it adopts widespread technologies, such as Java and Drools [13], which reduces development costs. These decisions allow designers and experts to focus on eliciting and managing the specific domain knowledge required for the annotation process, while facilitating the examination and validation of results.

The rest of the paper discusses these aspects in detail. Section 2 presents the background of the functional annotation problem. MASSA is briefly described in Section 3, while INFAES and the methodology used to tackle the problem are introduced in Section 4. Section 5 exemplifies the annotation and evaluates the system performance. The state of the art in systems for annotation is reviewed in Section 6. Finally, Section 7 discusses conclusions about the work and its results.

2. Background

In most living beings, the hereditary information is stored in macromolecules of DNA. Such molecules comprise two long complementary strands composed of small molecules called nucleotides. *Genes* are sections of these strands that detain the information to produce the proteins that participate in different biological processes. Determining the correspondences between genes and biological processes is the goal of *functional annotation*. This is a complex task, as there is not a one-to-one correspondence between genes and their functions, and the involved techniques are generally expensive in effort and resources.

Next sections provide further insights into this problem. Section 2.1 presents its biological basis, and Section 2.2 the current techniques applied in functional annotation and their main features.

2.1. Biological concepts

DNA molecules, and in particular their genes, contain the genetic information required to synthesize functional cellular components. This process is called *gene expression*, and has two parts. The first one is the *transcription*, where a complementary strand of nucleotides of a gene is transcribed into a messenger ribonucleic acid (mRNA); the second one is the *translation*, where the mRNA is translated into proteins. Since the DNA can be transcribed from both strands, a total of six *reading frames* (three from each strand, as they are always translated grouped by triplets) are possible for further translation into proteins.

During the transcription, different parts of the gene can be used to form distinct mRNAs. This phenomenon is known as *alternative splicing*, and it is the reason why a single gene can code different proteins.

In a simplistic way, a protein can be seen as a large molecule composed of amino acid (AA) chains. The *primary structure* of a protein is the linear sequence of these AAs. This structure can fold into itself forming two-dimensional organizations (e.g., helices, sheets, and turns) known as *secondary structure*. The components of this structure in turn, are folded into compact globules that form the *tertiary structure*.

The protein function is directly linked to its tertiary structure. However, some portions of the primary structure can vary substantially without changing the protein role. In fact, the AA sequence contains sections called *conserved residues* or *regions*, which are responsible for the functionality of the protein. Among them are *domains*.

Domains are compact, local, semi-independent units in proteins. Their existence and function is not tied to specific proteins, and their sequences tend to be more conserved than those of other regions. For these reasons, they are widely used to establish the functions of proteins. A protein may have one or more domains, and proteins that have the same domains are generally classified into the same *family*.

A protein family is a set of evolutionary-related proteins that share a significant degree of similarity. The members of a family are called *homologs*, and they descend from the same ancestor. Usually, homologs are 25% or more identical throughout their sequences. Homologs can be classified as *orthologs* or *paralogs*. The first ones

Table 1
Examples of publicly available biological databases.

Data type	Database name
Genome and genomic information	Ensembl [24]
Gene	GenBank [25]
Protein	Entrez protein [26] and UniProt [27] (composed of Swiss-Prot[a] and TrEMBL[b])
Protein domain and family	Conserved Domains Database (CDD) [28], InterPro [29], Pfam [30], Smart [31], TIGRFAM [32], ProDom [33], and Panther [34]
Ontology	Gene Ontology (GO) [35]

[a] Manually annotated and reviewed by experts.
[b] Automatically annotated and not reviewed by experts.

Table 2
Examples of SDB publicly available at NCBI.

SDB name	Content
NR	Database of non-redundant protein sequences, with entries from GenPept (translations for each of the coding sequences within the GenBank [25]), Swiss-Prot, and other sources
NT	Database of non-redundant nucleotide sequences, with entries from all traditional divisions of GenBank, EMBL[a], and DDBJ[b]
CDD	NCBI[c]-curated domains plus domain models from Smart, Pfam, TIGRFAM, and other sources

[a] http://www.embl.de.
[b] http://www.ddbj.nig.ac.jp.
[c] National Center for Biotechnology Information http://www.ncbi.nlm.nih.gov.

are found in different species due to speciation events, while the second ones are the result of gene duplication. Since orthologs often preserve their biological role, identifying them allows transferring functional information between genes from different organisms with a high degree of reliability [14].

2.2. Predicting the protein function

There are different approaches to predict the function of a protein. Some techniques are based on the direct study of the three-dimensional (3D) structure of the protein, e.g., X-ray crystallography and nuclear magnetic resonance. These are usually expensive, and require providing the appropriate conditions for the polypeptide chain of the protein to fold properly [15]. The tertiary structure of the protein can also be predicted based on its primary structure, but this is a NP-complete problem [16]. Nevertheless, the sequence of AA alone offers insights into that 3D structure and its function, cellular location, and evolution [17]. In this line, the most common, and generally more accessible, approach to functional annotation is "inheritance through homology" [18], i.e., what can be inferred based on the similarities between a protein of interest and previously studied proteins.

A common way of finding similarities between sequences is aligning them. A great variety of aligners and sequence comparison tools is publicly available [19]. One of the most popular is BLAST [20]. It searches in a pre-compiled database (i.e., Search Database (SDB)) for target sequences that have a high homology with a given query sequence. Its variants can be used for many purposes such as: finding protein family members, and predicting a protein function or its 3D structure (i.e., BLASTP); discovering genes in a genome or a protein encoded in a sequence (i.e., BLASTX); or identifying conserved domains in a protein sequence (i.e., RPS-BLAST).

In order to characterize a gene function, its sequence should be mapped to known genes, and if a good match is found a prediction is assigned, or its protein domains should be identified [21]. Nevertheless, the transference of an annotation from a sequence to another using the "inheritance through homology" approach should not relay only on the similarities between sequences. Additional information should be collected to support (or to question) the proposed annotation.

Additional tools and databases are used to get that information. Examples of tools are Orthostrapper [22] and InParanoid [14] for ortholog detection, and InterProScan [23] and RPS-BLAST for family/domain detection. Tables 1 and 2 show some databases with biological information.

After the previous steps, a shared vocabulary, like GO [35], should be used to give the proper description of the functions predicted. This allows experts to represent the annotation in terms of elements from the categories cellular components (CC), molecular function (MF), and biological process (BP).

All these tasks enhance the results of the functional annotation, but make this process demanding and fairly labor-intensive. At the moment, tools tend to focus on specific parts of the process, automating routine and repetitive tasks over huge preliminary data sets.

3. MASSA

MASSA (*MAS to Support functional Annotation*) is an integrated system for functional annotation. As a MAS, it is composed of a set of interacting IA [10] that perceive their environment and act upon it. This acting is autonomous and based on the knowledge an agent holds about its environment and itself. The adoption of this paradigm is justified by its suitability to satisfy the three main goals leading the MASSA design: to facilitate the introduction of components that apply expert knowledge or specific tools in different tasks of the annotation process; to promote the distribution of tasks across the available computational resources; to reduce the users' workload by delegating in the system most of routine tasks. MASSA has been designed following the INGENIAS [36] methodology for MASs.

MASSA has two main subsystems (see Fig. 1): *MASSAPipe* and *MASSAInference*. MASSAPipe is a pipeline that executes tools to search information that is used in the inference process of the annotation. These are mainly publicly available tools that implement routine tasks of the annotation process. MASSAPipe currently integrates BLAST [20], RPS-BLAST against CDD [28], InterProScan [23], and Orthostrapper [22]. There are also scripts specifically developed for MASSAPipe that mine online information throughout different databases. This subsystem is also in charge of keeping its data sources up to date. Once that MASSAPipe has acquired all the information for a query, it stores this data into the project Result Database (RDB). MASSAInference uses that information to assign the most likely annotation to sequences. *InferenceAgents* (InfAs), which encapsulate the INFAES RBES, are responsible for this task. The *InterfaceAgent* receives the user requests, maintains users informed of the status of their jobs, and makes the outcomes available to users.

All the previous information (i.e., the user's request, the data from the pipeline, and the information generated by the RBES), with additional internal information to manage these tasks in MASSA, constitutes a *project*. MASSA can process multiple projects in parallel.

MASSA is implemented in Java with the Jade framework [37] for MASs. Most of existing Bioinformatics scripts used in MASSAPipe are written in Perl. The RDB is supported by MySQL.[2]

[2] http://www.mysql.com.

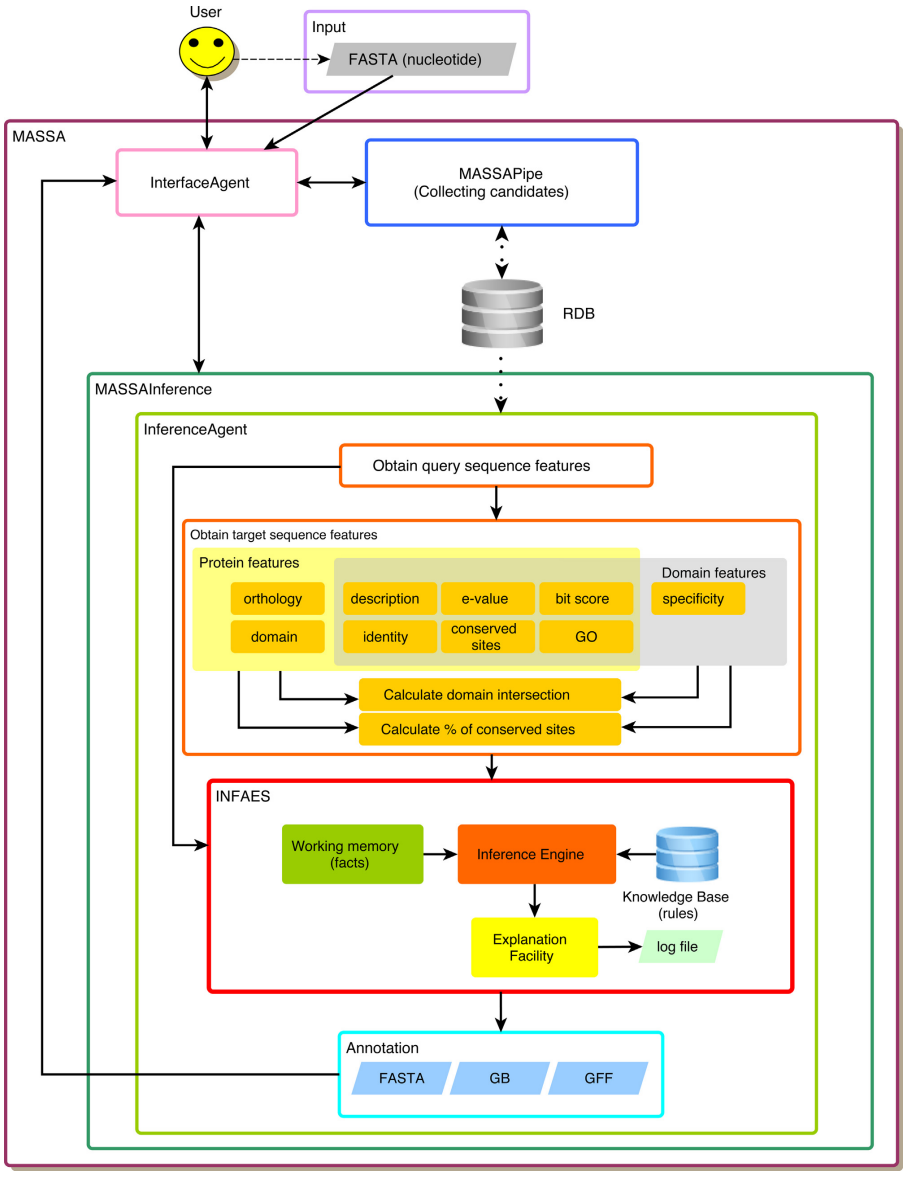


Fig. 1. MASSA architecture.

Table 3
Candidate attributes.

Id.	Attribute	Description	Homolog sequence	Domain or family
1	Candidate identifier	Unique identifier of the sequence. Its format depends on the database	YES	YES
2	Candidate description	Annotation given to the target sequence	YES	YES
3	Candidate source	Database where the candidate is stored. This value varies from 0 to 1, according to the source database features (e.g., data from automatic or manual annotation, and cross references to GO terms and other databases). Manual curated databases, such as Swiss-Prot, have higher *Candidate source* values than not reviewed ones, like TrEMBL	YES	NO
4	Strand	Strand transcribed. This value can be + or −	YES	YES
5	Frame	Reading frame. This value varies from 0 to 2	YES	YES
6	Query start	First coordinate of the query sequence that aligns with the target sequence/domain/family	YES	YES
7	Query end	Last coordinate of the query sequence that aligns with the target sequence/domain/family	YES	YES
8	Target start	First coordinate of the target sequence/domain/family that aligns with the query sequence	YES	YES[a]
9	Target end	Last coordinate of the target sequence/domain/family that aligns with the query sequence	YES	YES[a]
10	*E*-value	The expectation value represents the likelihood of a match had occurred by chance. The closest to 0, the more significant is the match	YES	YES
11	Bit score	This value is derived from the raw alignment score, taking statistical properties of the scoring system into account. It indicates how good the alignment is: the higher the bit score is, the better is the alignment. Since this value is normalized, bit scores from different alignments can be compared [38]	YES	YES[a]
12	Identity	The percentage of residues that match in the alignment. This value varies from 0 to 1	YES	YES[a]
13	Orthology	The score given by Orthostrapper [22] to each target sequence. The highest this score is, the more likely the orthology relationship. If the target sequence was not classified as a possible ortholog, this value is −1	YES	NO
14	Query Sequence Domains or Families (QSDF)	The domains/families found in the query sequence through similarity searches	NO	YES
15	Target Sequence Domains or Families (TSDF)	The domains/families that belong to the target sequence according to its source database	YES	NO
16	Domain or Family Intersection (DFI)	The intersection between the domains/families existent in the target sequence (15) and those found for the query sequence (14), i.e., $QSDF \cap TSDF$. This intersection takes into account: strand (4), frame (5), and query and target coordinates (6–9)	YES	NO
17	Specificity	A RPS-BLAST correspondence (i.e., *hit*) between a query sequence and a conserved domain has a level of confidence. A hit must meet or exceed the Threshold Bit Score (TBS) to be considered *specific* [39]. This represents a high-confidence association, and thus, in the inferred function of the protein query sequence. The specificity is 1 if the CDD domain/family bit score is greater or equal than the TBS, and 0 otherwise	NO	YES[a]
18	Percentage of Conserved Sites (PCS)	Some sequences have important sites associated to them described in databases such as CDD [28], Entrez Protein [26], and UniProt [27]. The percentage of conserved sites is the percentage of these sites found preserved in the query sequence after aligning it with the target sequence	YES	YES
18	GO	GO[35] terms associated to the homolog sequence or domain/family	YES	YES

[a] Not for InterProScan data.

The previous architecture puts most of the applied expert knowledge into INFAES. The MASSA approach tries to facilitate its evolution to adapt it to new information and techniques for functional annotation. For this purpose, this research bases its development on the well-known methodology CommonKADS [12] for knowledge-based systems, as described in [11].

4. INFAES expert system

INFAES is the RBES of MASSA. It performs the inference task of the functional annotation. This section discusses in detail its structure and components (see Section 4.1), the decisions and data flow implemented by its rules (see Section 4.2), and some development and implementation issues (see Section 4.3).

4.1. Structure

RBESs [7] have two main components that contain their information (i.e., rules and facts), the *working memory* and the *KB*. An engine triggers and interprets rules according to the available information, and updates the working memory. Next sections discuss these components for INFAES, with Section 4.1.1 for the working memory and Section 4.1.2 for the KB. The engine is provided by Drools [13] (see Section 4.3).

INFAES also includes an explanation component that summarizes the inference process (e.g., available facts, rule activation, and agenda). It takes the information provided by the inference engine, summarizes it in a log file, and delivers this to users through the MASSA interface after completing the annotation process.

4.1.1. Working memory

The working memory comprises all the information related to projects. It includes the initial data provided by MASSAPipe (see Section 3), and the intermediate results generated by rules (for these, see next sections). The management of this information is organized around the concept of candidate annotation. An INFAES *candidate* is an object that has an identifier, a sequence, and a description (i.e., the annotation) among other attributes.

MASSAPipe obtains candidates using a similarity search with BLASTX [20] against different databases. These results go through a first filter based on content. This filter uses regular expressions associated to "uninformative" annotations. Terms such as "unnamed protein product", "protein of unknown function", or "putative uncharacterized protein" are considered functionally uninformative, so their associated candidates are discarded. After this filter, information like e-value, bit score, percentage of identity, and the alignment of the candidate sequence with others are collected. Other algorithms, such as those of RPS-BLAST or InterProScan [23], are also used to fetch domain and family information. Specific information (e.g., conserved sites and their coordinates, and GO terms) is acquired from databases such as CDD [28], Entrez Protein [26], UniProt [27], and InterPro [29]. Besides, evidence of orthology is calculated by Orthostrapper [22] and combined with the previous data. Table 3 shows the attributes a candidate may have after the previous tasks according to its type: homolog sequence (i.e., proteins or nucleotides) or domain/family.

4.1.2. Knowledge base

The knowledge extracted for the annotation problem following CommonKADS [12] was converted into rules. These rules are mainly constructed out of heuristics on sequence attributes that can indicate whether a target sequence is a good candidate or not. MASSAPipe gets the majority of these features during the gathering stage (see Table 3), though some of them are collected or calculated by the RBES itself.

Table 4
Scores calculated for a candidate.

Score type	Description/how is calculated
Source score	This value is equal to the *Candidate source* (see attribute (3) in Table 3)
Orthology score	This values is equal to the *Orthology* (see attribute (13) in Table 3)
Domain score	This score summarizes the similarity between the domains/families of the query sequence and the candidate. It is calculated as: *number of elements in DFI / number of elements in TSDF*. This value varies between 0 and 1. If all the domains/families predicted for the query belong to the candidate, it has the maximum value, 1. If the query sequence is apparently a fragment of the target sequence, it will not have some of its domains/families, and the value is higher than 0 but less than 1
Specificity score	It is the arithmetic mean of the *Specificities* (see attribute (17) in Table 3) of all domains/families that belong to the DFI, that is, *sum of Specificity of all DFI elements/ number of DFI elements*. It ranges from 0 to 1
Conserved sites score	This value is equal to the PCS (see attribute (18) in Table 3)

The rules of the INFAES KB perform two main processes at the inference stage. Firstly, they process and assess the previous features for each target sequence, and based on them, they calculate some scores (see Table 4). Secondly, they compare the different candidates using their features and the additional scores. The sequence that best fits their criteria is chosen as result, and its annotations transferred to the query sequence. This process is explained in detail in Section 4.2.

There are currently more than sixty rules in the KB. They are structured and organized to follow the steps of the reasoning process of experts and to facilitate the management and addition of new knowledge. Rules are grouped according to the features and scores they evaluate (e.g., source, orthology, domain, or conserved sites). INFAES uses a combination of different strategies to tackle conflict resolution among them: specificity, rule ordering, and context limiting. The first two are standard for RBESs. The third one is used to impose certain expert constraints, such as processing *Specificity score* rules only after exhausting *Domain* related rules, since it is necessary to identify a domain before calculating its *Specificity score*.

4.2. Inferring the annotation

INFAES uses the information acquired in MASSAPipe (see Section 3) and the calculated scores (done by the rules in the KB, see Section 4.1.2) to infer the best annotation for a query. For this purpose, it applies heuristics based on biological and tool knowledge over that information. Fig. 2 summarizes this inference process.

In the best-case scenario, previous stages found for the query sequence similarities with data from protein databases (Fig. 2(2)), along with orthology likelihood (Fig. 2(6)), domain information (Fig. 2(10) and (13)), and conserved sites information (Fig. 2(16)). Sometimes there is no match to homolog candidates, and then the best bet is trying to infer the function based on the domain alignments (Fig. 2(19)). Nevertheless, some sequences do not match to domains either (Fig. 2(26)). In a case like that, the sequence is added to a "Not annotated" list.

The system can therefore take two ways to annotate the data: based on homolog sequences (see Fig. 2(a) or on domains (see Fig. 2(b). The annotation based on homolog sequences is the one that can result in a more reliable and explanatory outcome, because it takes into consideration all attributes and scores described before

D. Xavier et al. / Applied Soft Computing 35 (2015) 373–385 379

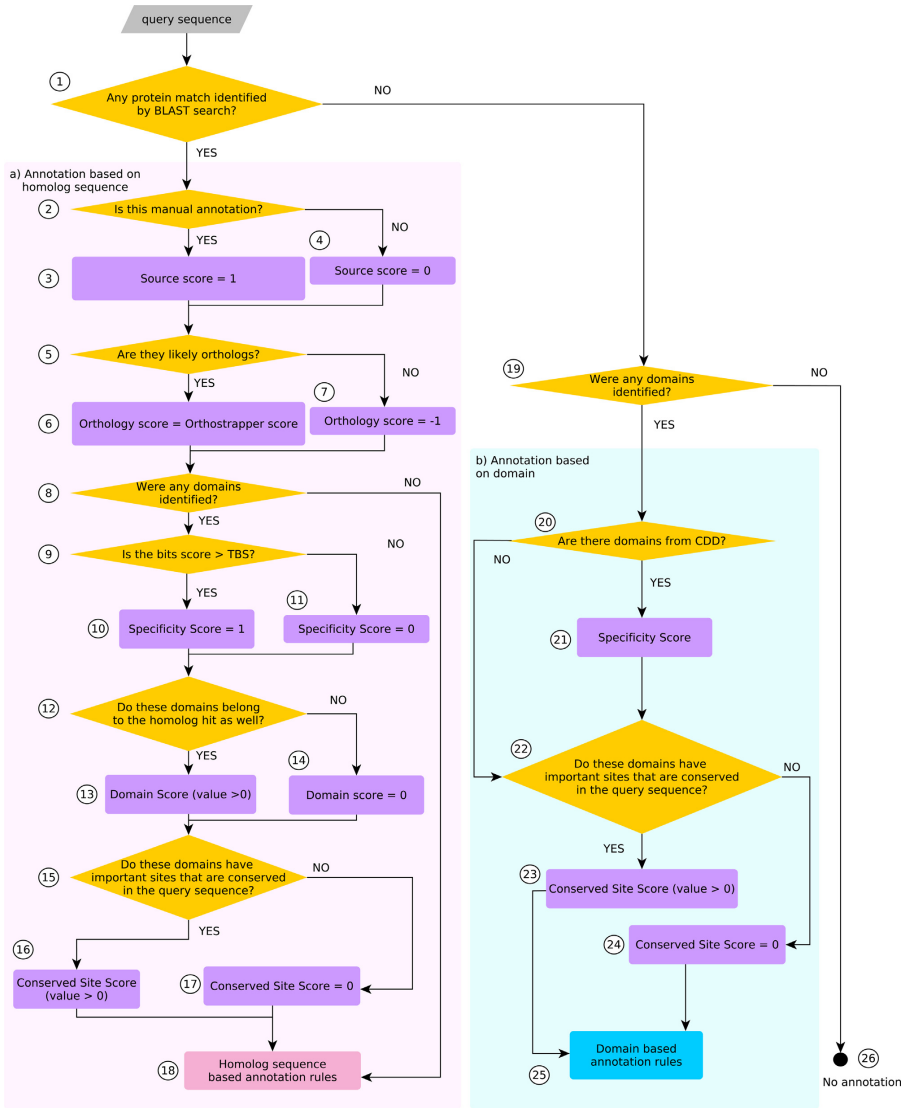


Fig. 2. Workflow for candidate evaluation.

(when they exist). On the other side, annotation based on domains relies only on the domain similarities and the conserved sites if any.

The previous choices correspond to the assessment of the attributes *Candidate source*, *Orthology*, *Specificity*, *Domain Intersection*, and *Conserved Sites*, and setting the related scores. After that, the system goes to the proper annotation rule set: *Homolog sequence based annotation rules* (see Fig. 2(18)) or *Domain based annotation rules* (see Fig. 2(25)). The candidates are compared based on the selected rule set, and sorted in a list of potential candidates that starts with the most likely and informative annotation.

In order to create this list, the first candidate processed is set as the first element of the list. Then, every new candidate scores,

along with some attributes, are compared to those of the elements in the list until the new candidate is correctly arranged. The best annotation, that is, the one from the first element in the list, is transferred to the query sequence.

During this last stage of the inference (see Fig. 2(18) and (25)), the GO, bit score, *e*-value, and identity attributes (see Table 3) play the role of decision values. When all scores are equal in a comparison, the first rule to identify the best candidate prioritizes the candidate containing GO terms; if both candidates have or do not have GO terms, the decision is made based on the highest bit score. If this attribute does not decide the "winner", the candidate with lowest e-value is chosen. In the case the candidates have the same e-value, the identity is taken into consideration.

Finally, the sorted list containing all the candidates is provided to users. In this way, they can check all the possible annotations and verify the prediction.

4.3. Development and implementation

Bioinformatics is a constantly evolving domain, so its tools should be able to keep up with this feature, besides attempting to solve the problems the field faces. Unfortunately, the majority of these systems are just result and performance oriented, not taking into account other aspects like software evolution. This ability to evolve is directly connected to the system's computational approach and architecture, the methodologies used to model the system as a whole and the knowledge it contains, and the technologies used to implement it.

INFAES is encapsulated in a modular and flexible system that follows the agent paradigm. Moreover, the RBES approach helps to separate the knowledge from the procedural code. As its rules represent knowledge in a more human-like way, it also makes the reasoning process more intuitive to experts, facilitates troubleshooting, and encourages their involvement in the production of new rules.

The knowledge used to develop INFAES (and MASSA) was extracted through structured interviews with experts in Biology and Bioinformatics [11], and complemented with information from other sources such as [18,21]. This knowledge was modeled through the state-of-the-art methodology CommonKADS [12], and pointed out to a classification task: the annotation problem can be viewed as one of finding the protein "class" that best characterizes a certain sequence.

Regarding the system implementation, INFAES uses the facilities provided by Drools [13]. These include an inference engine, components to output the trace of the decisions made, and development libraries and tools.

The Drools engine uses forward chaining. That is, the system starts with the data of the initial evidences, and applies rules to them to obtain intermediate data and infer all the possible conclusions. This approach is in line with that used by human experts in functional annotation.

Drools can be easily integrated with Java. This facilitates interactions between INFAES and the rest of MASSA, as this is developed with the Java-based framework Jade [37] for MASs. During the inference process, Java objects are used to manipulate several types of facts in the working memory. A sorted list *PotentialCandidateList* stores all the target sequences for a query. Each target sequence is related to a *PotentialCandidate* object composed of all the attributes (see Table 3) that have been collected about it. A *PotentialCandidate* object, in turn, may contain a list of *DomainCandidate* objects, which describe all the features related to each domain existing in the sequence. Both *PotentialCandidate* and *DomainCandidate* may have a list of *ConservedSite* as attribute. A *ConservedSite* object stores the information about a conserved site. Adding new features to consider in these facts is straightforward, as it just requires introducing new attributes to the classes and editing the relevant rules.

5. Experimentation

This section illustrates the actual use of INFAES. Section 5.1 discusses how INFAES annotates a given sequence. In Section 5.2, INFAES performance is evaluated using the data and methodology of the first large-scale community-based critical assessment of protein function annotation (CAFA) [1].

5.1. Inference example

In order to illustrate the INFAES reasoning process, the human sequence *Q8IVL6* (*Prolyl 3-hydroxylase 3*) was selected from CAFA's data. The MASSAPipe pipeline used to obtain the candidates involved: BLASTX against NR[3] and Swiss-Prot[2], RPS-BLAST against CDD[4], InterProScan[5], and Orthostrapper [22]. Aiming to perform a realistic experiment, all sequences related to human were removed from the SDBs. Then, the outcomes were processed by MASSAInference and inferred by INFAES.

The pipeline yields 11 homolog candidates and 4 domains/families (IPR006620, IPR005123, PF13640, and SM00702). Since homolog sequences were found (see Fig. 2(2)), the system used the *Annotation based on homolog sequence* (see Fig. 2(a)) approach to predict the query function. For the sake of conciseness, the rest of the inference process is only described for the potential target *Q8CG70*.

Firstly, the *Candidate source* is assessed and the *Source score* (see Fig. 2(3) and (4)) is set. In this case, since Swiss-Prot is a curated database, its candidates get a score of 1, whereas NR candidates receive 0. After this, an *Orthology score* is given based on the Orthostrapper prediction (see Table 5). Then, as the system has identified domains which bit scores exceed their respective TBSs (see Table 6, rows 5–6), it gets the *Specificity score* (see Fig. 2(10)) by calculating the arithmetic mean of the *Specificities* from all domains in the DFI (see Table 3 for both attributes).

The *Specificity* value is propagated to the domains with the same *Candidate identifier* (see Table 3, row 1) and coordinates (i.e., *strand*, *frame*, query/target *start* and *end*) (see Table 3, rows 4–9) but different *sources* (see Table 6, rows 3–4). In the present case, the *Specificity score* is 1 (see Table 4) because the defined *Specificity* (i.e., not "NA") of all the QSDF that belong to the DFI is 1. Besides, since all the TSDF (see Table 7) are also predicted for the query sequence (DFI = TSDF), the *Domain score* (see Fig. 2(13)) is 1 as well. Four conserved sites, related to "*Metal binding*" and "*Active site*", are described on *Q8CG70* records, and all of them are preserved in the query sequence. Based on that, the *Conserved site score* (see Fig. 2(16)) is 1.

The next step is adding the already scored candidate to the *PotentialCandidateList*. If the list is empty, the new candidate is the first element; otherwise it is compared with the other elements and sorted according to the *Homolog sequence based annotation rules* (see Fig. 2(18)). Table 5 shows the 11 candidates sorted based on these rules.

After comparing all candidates, the system predicts that *Q8CG70* has the best chances to be an accurate annotation: it comes from a curated source, it was classified as an ortholog, the domains predicted in the query sequence have highest bit score than the TBS and also exist in the target sequence (see Tables 6 and 7), the important sites described are conserved, and it has GO terms

[3] Downloaded on 16th July 2014.
[4] Last modification on 12th February 2014. Downloaded on 30th May 2014.
[5] Version 5.4-47.0, running the following analyses: TIGRFAM, ProDom, Panther, SMART, and Pfam.

Table 5
Candidates and their scores and some attributes for the query sequence *Q8IVL6*.

Accession	Description	Source score	Orth. score	Domain score	Specif. score	Cons. site score	Bit score	*E*-value	Identity
(1) sp|Q8CG70	Prolyl 3-hydroxylase 3	1	1	1	1	1	965	0	0.7
(2) gi|426371415	Prolyl 3-hydroxylase 3	0	1	1	1	0	1114	0	0.79
(3) sp|Q3V1T4	Prolyl 3-hydroxylase 1	1	0	1	1	1	449	$1.00e^{-145}$	0.45
(4) sp|Q6JHU8	Prolyl 3-hydroxylase 1	1	0	1	1	1	429	$1.00e^{-137}$	0.44
(5) sp|Q6JHU7	Prolyl 3-hydroxylase 2	1	0	1	1	1	408	$1.00e^{-130}$	0.42
(6) gi|441670306	Prolyl 3-hydroxylase 3	0	0	1	1	0	1102	0	0.79
(7) gi|635063073	Prolyl 3-hydroxylase 3 isoform X1	0	0	1	1	0	1100	0	0.79
(8) gi|297261676	Prolyl 3-hydroxylase 3-like isoform 6	0	0	1	1	0	1099	0	0.79
(9) sp|Q9R1J8	Prolyl 3-hydroxylase 1	0	0	0.5	1	1	447	$1.00e-144$	0.44
(10) sp|Q8CG71	Prolyl 3-hydroxylase 2	0	0	0.5	1	1	397	$1.00e-126$	0.41
(11) sp|Q4KLM6	Prolyl 3-hydroxylase 2	0	0	0.5	1	1	397	$1.00e-126$	0.41

Table 6
Domains found in the query *Q8IVL6* according to InterproScan[5] and RPS-BLAST against CDD. Attributes that play an important role in the reasoning process are described for each domain.

Domain	Source	TBS	Bit score	Start	End	Specificity
(1) IPR006620	InterproScan	NA	$2.80e-48$	464	674	NA
(2) IPR005123	InterproScan	NA	$8.60e-14$	581	673	NA
(3) PF13640	InterproScan	NA	$8.60e-14$	581	673	1[a]
(4) SM00702	InterproScan	NA	$2.80e-48$	464	674	1[a]
(5) SM00702	RPS-BLAST	102	$1.00e-24$	474	674	1
(6) PF13640	RPS-BLAST	54	$1.00e-8$	581	674	1

[a] Propagated specificity.

Table 7
Domains (related to the analysis executed[5]) that belong to the target sequence *Q8CG70* according to the UniProt database.

Domain	Start	End
IPR006620	460	671
IPR005123	557	671
PF13640	577	669
SM00702	460	670

associated (GO:0003674, GO:0005506, GO:0005575, GO:0005783, GO:0008285, GO:0016491, GO:0016702, GO:0016705, GO:0016706, GO:0019797, GO:0031418, GO:0046872, and GO:0055114).

MASSA transfers this best candidate as the annotation for the query sequence. The outcome is written in GenBank [25], GFF [40], and FASTA [41] formats.

The automatic annotation predicted by INFAES was compared to the one in UniProt through the protein-centric metrics described in CAFA [1]. Section 5.2.1 briefly explains these metrics. For a fixed threshold of 1, the system predicted the *BP* with *precision* ($pr_{Q8IVL6}(1)$) and *recall* ($rc_{Q8IVL6}(1)$) of 1 and 0.844 respectively, resulting in $F\text{-}measure_{BP} = 0.915$ (out of 1). The *MF* inference scored 1 for both metrics, and consequently $F\text{-}measure_{MF} = 1$.

5.2. Evaluation

Assessing the accuracy of systems for functional annotation is not a trivial task, and depending on the level of complexity of the annotation, it can be quite subjective. Although many works compare the results of their systems with others, the research community has not agreed yet a formal benchmark for this purpose. CAFA [1] proposed a methodology to assess different algorithms based on the GO terms obtained from the annotation for MF and BP categories. Section 5.2.1 describes its measurement method, and Section 5.2.2 applies it to INFAES.

5.2.1. The methodology

The CAFA methodology uses metrics of *precision* and *recall*, and the *F-measure* to calculate the performance of algorithms. The *precision-recall* approach has been extensively used in fields such as pattern recognition and information retrieval. It takes into account the relevance of the results, and its outcome is fairly easy to interpret. For a given annotation and GO category, *precision* is the fraction of GO terms that were correctly inferred rather than incorrectly, whereas *recall* is the fraction of GO terms that were correctly inferred rather than missed. *Precision* and *recall* can be combined in a single performance measure, called *F-measure*. The *F-measure* is calculated as follows:

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}, \; where\, precision = \frac{\sum(CPGO \cap EDGO)}{\sum CPGO},$$

$$recall = \frac{\sum(CPGO \cap EDGO)}{\sum EDGO}$$

CPGO is the set of GO terms predicted for a given category, and EDGO is the set of GO terms determined as true experimentally.

CAFA evaluated 54 tools on a target set of 866 sequences from 11 organisms with this methodology. These tools use a variety of computational approaches and algorithms, and sometimes different biological knowledge. Their results were translated in terms of F_{max}[6], where $F_{max} = 1$ characterizes a perfect predictor. They were ranked based on this value, and the 10 top-performing tools where presented in the CAFA's work [1]. The best 3 were: Jones-UCL [42], Argot2 [43], and PANNZER[7] (see Section 6.1).

5.2.2. INFAES evaluation

In order to evaluate the RBES presented in this work, the 866 phylogenetically diverse sequences from CAFA [1] were annotated

[6] *F-measure*'s maximum value over all algorithm thresholds.
[7] http://ekhidna.biocenter.helsinki.fi/pannzer.

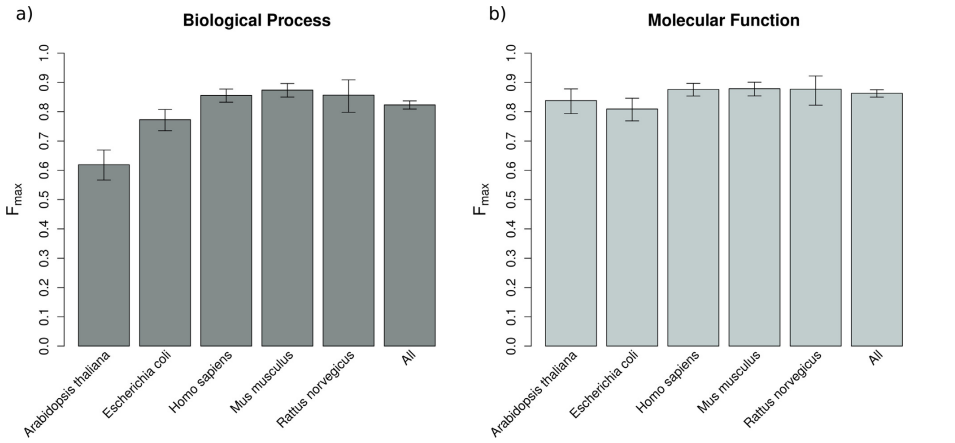


Fig. 3. Performance assessment for (a) BP and (b) MF. Only species with more than 30 sequences were plotted separately. The last bar of each plot contains all the 866 sequences examined. Confidence Intervals (95%) were calculated through bootstrap with $n = 10,000$ iterations on the set of query sequences. Annotations containing only "protein binding" as MF were excluded from the analysis due to its uninformative character [1].

using INFAES. The sequences were downloaded from UniProt [27], written as nucleic acid sequences using *backtranseq* [44], and submitted to MASSAPipe. The pipeline used to annotate these data was the same of Section 5.1. Since the SDBs used in this evaluation may already contain the query sequences and their respective annotations, they were purged. MASSAPipe was executed separately for each organism, and all the sequences related to these organisms were removed from the SDBs. This approach creates a more realistic scenario, as it prevents in the experiment a sequence to be annotated with its annotation from databases if that is already available. Then, INFAES annotated the query sequences, and the results were assessed based on the protein-centric benchmark presented in CAFA (Fig. 3).

For the system evaluation, the GO terms produced were grouped in MF and BP, and a F_{max} was calculated for each category. Both F_{max} yielded values greater than 0.8, what indicates that INFAES has a high performance. Aiming at following all the methodology described in CAFA, a domain analysis was also carried out. Although the annotations can have domains from different sources, the majority of them (95%) are from Pfam. As depicted in Fig. 4, most of the sequences are associated to a single-domain, what was already highly expected [1].

The results produced in the INFAES evaluation were compared to the ones described in CAFA and, according to them, INFAES outperformed the 10 top-ranked tools evaluated in this experiment. The reasons of this result are a combination of the system accuracy and the evolution of the databases. For instance, from September 2010 to June 2014, 26,188 sequences (4.8% of the current entries) have been manually curated and added to the Swiss-Prot database.[8][9] It is hard to establish the level of impact these new data have on our experiment. Our system and its tools (as it also happens with the tools in the CAFA experiments) access a range of local and remote databases, where it is not possible to reproduce the exact setting used for CAFA [1] four years ago.

[8] ftp://ftp.uniprot.org/pub/databases/uniprot/previous_releases/release-2010_09/knowledgebase/Swiss-Prot_statistics.html.
[9] ftp://ftp.uniprot.org/pub/databases/uniprot/previous_releases/release-2014_06/knowledgebase/UniProtKB_SwissProt-relstat.html.

6. Related work

Currently, there is a variety of tools to support experts in the process of functional annotation. These tools present relevant differences across two orthogonal dimensions: the biological one, which includes the knowledge they consider, the algorithms to manipulate it, and their goals for annotation (see Section 6.1); and the developmental one, with aspects such as the paradigm and process adopted to build them (see Section 6.2).

6.1. Knowledge base and goals

The process of functional annotation has currently been solved only partially. There are no tools that incorporate all kind of knowledge potentially applicable to it, or that can be used for any type of query.

Regarding the applicable knowledge, most tools consider at least homolog sequences combined with domains and GO information [4,5,42,45–48]. Some tools go further and add additional knowledge on: transmembrane topology prediction [2,49–51]; ortholog inference [3]; gene expression and protein–protein interaction information allied to probabilistics [42]; or semantic distances and structure in ontologies [43]. Moreover, the majority of tools has a small and fixed range of databases to search. For instance, FastAnnotator [5] only uses NR as SDB for homology and Pfam [52] for domain identification, and Argot[2] [43] uses UniProt [27] and Pfam for the same purposes. Although more information seems to be always useful for a prediction, sometimes it does not imply a real contribution for the annotation [1], like the case of the high-throughput data-integration in [42].

These tools also employ different algorithms for the inference stage of annotation. For instance, the three best performing tools in CAFA [1] were PANNZER[7], Argot[2] [43], and Jones-UCL [42]. PANNZER combines several statistical methods (i.e., regression, Term Frequency – Inverse Document Frequency, and Gene Set Z-score) and hierarchical clustering. Clustering is also used by Argot[2]

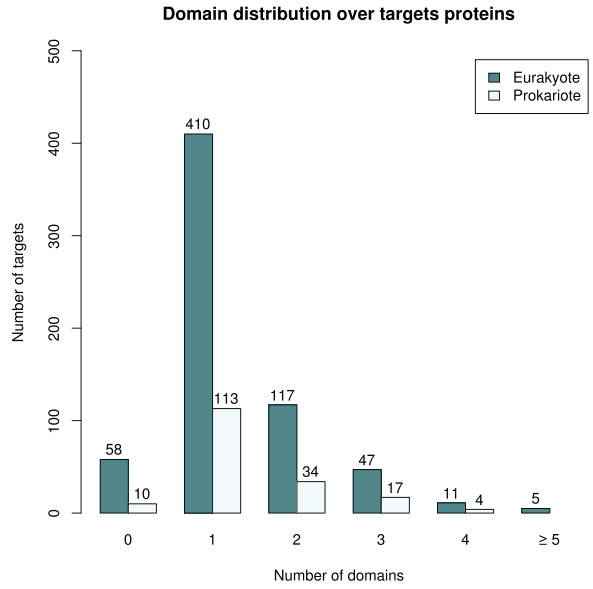


Fig. 4. Distribution of single-domain and multi-domains from Pfam over eukaryotes and prokaryotes sequences. Eukaryotes distribution is very similar to that presented in CAFA.

to select the most accurate GO terms for a given target protein. These selection is based on the semantic similarity measure and according to a *Z*-score. Jones-UCL, on the other hand, uses a network propagation algorithm based on the GO graph structure to combine predictions from different sources. To gather information before the inference tasks, these tools use pipelines of more basic tools.

The target of the tools also differentiate them. There are tools that were developed to annotate complete genomes [2], so they are not suitable when the goal is annotating sequences that are only part of this scope. Moreover, some tools focus only on a specific biological domain, such as bacterial [4] or transcript [5] annotations.

INFAES was developed to deal with DNA sequences regardless their genome context. Currently, it considers ortholog relationship between sequences, and the existence of domains, families, and conserved sites. In order to obtain functionally relevant annotations, it prioritizes curated databases, and entries containing informative descriptions and GO terms. In this way, it takes into account most of the knowledge considered in the most popular tools. Moreover, since INFAES manages data as *attributes* of *objects*, it can process data from any source that MASSAPipe incorporates to its flexible pipeline.

6.2. Development approach

The development of tools for functional annotation has adopted multiple approaches. Differences include the paradigm, the computational approach, the model followed to guide their development, and the way they are made available.

The paradigm is concerned with the style of programming. There are two main groups here, imperative and declarative approaches. The first one merges the control and logics of the program; the second one focuses on logics without specifying the control, which is responsibility of some available engine. For the case of functional annotation, examples of tools developed under imperative paradigms are those that use Perl [2,5] or Java [49,50]. Examples of declarative tools are those developed with languages and environments such as Prolog [3], Jess [47], and Drools [13] (this work). The choice among these approaches implies certain trade-offs. As declarative approaches focus on problem logics, they are usually considered closer to domain experts than imperatives. Experts can examine and understand the knowledge these systems embed in an easier way than in the case of imperative programs. However, imperative approaches are usually more efficient from the computational point of view, and they allow programmers manually fine tuning the control of systems.

As pointed out before, there are multiple declarative approaches. The choice between them is largely a question of the kind of knowledge and inferences usual in the problem domain [53]. Logic programming languages, like Prolog [3], are better suited to deal with deductive processes. Rule-based approaches, like Jess [47] and Drools [13], are oriented toward heuristic and problem-solving knowledge. The choice should also depend on the existence of a strong supporting community that guarantees the continuity of the selected tools. Some annotation systems depend on less popular languages, like EDITtoTrEMBL [51] with Well-Founded Semantics with eXplicit negation (WFSX), what may compromise the software evolution.

The way the inference is made is also important. The clearer it is, the easier to test and explain it, and evolve the system. For instance, although machine learning can play a positive role in functional annotation, its process could be difficult to interpret [1]. On the other hand, rules are well-known to be a natural way of specifying knowledge for experts, facilitating their comprehension and contributions during system evolution.

Another issue to consider is the support for the development process itself. Developing complex systems requires the adoption of systematic approaches from Software and Knowledge Engineering, including process models and support tools. Although the related literature contains multiple examples of methodologies applicable in this context, this is an issue not considered (or at least not documented) for most of the studied systems. Here, the use of well-known methodologies in the development (INGENIAS [36] for MASSA and CommonKADS [12] for INFAES) can be considered a novelty in the area. The adoption of these methodologies does not only facilitate the development of systems, but also the repeatability of the process and the analysis of its tasks and outcomes. This can contribute to a higher involvement of domain experts and their collaboration with engineers, as all of them will find easier examining the developed systems.

The way these tools are made available should also be taken into account. Although the web service approach seems very attractive, specially for Biologists, it imposes constraints on, for instance, the type, version, and parametrization of the software, the considered databases, or task durations [54], not to mention possible security issues. Furthermore, it usually limits the frequency of queries, so in general it is not suitable for a whole annotation project. Moreover, users have no control on how the inference is carried out. Annotation tools such as [3–5,43] are only available in this format. MASSA, together with INFAES, is going to be publicly available in the near future.

7. Conclusions

This paper has presented INFAES, a RBES for functional annotation. Functional annotation is currently a challenging process that involves the use of specialized tools and databases, and demands intensive expert participation. Support software tools for it present common limitations. On one side, there are those related to the annotation process, including the use of only part of the knowledge potentially useful for the task, and their limited scope that precludes annotating certain query sequences. On the other side, the issues pertaining to their development, such as the absence of documented engineering approaches supporting it, and the difficulties to maintain and evolve these systems. To address these problems, INFAES works on its development approach, design, and knowledge, with this paper focused on this last aspect.

INFAES presents a KB design intended not only to produce accurate annotations, but also to facilitate the gradual integration of the knowledge elicited from experts. Its organization around scores promotes creating groups of rules to address specific areas of knowledge, and combine their results through additional rules directly related to the decision process.

INFAES also brings improvements in the methodological approach to build ESs for annotation. It applies existing state-of-the-art methodologies to build both the embedding MAS (with INGENIAS [36]), and INFAES (with CommonKADS [12]) (see [11]). The content of the KB combines knowledge from different tools and experts that was extracted using knowledge elicitation techniques. The application of these methodologies to build the system facilitates its systematic study and maintenance, and thus its evolution according to users' needs and new knowledge and technologies.

Regarding its design, INFAES is part of a wider functional annotation project called MASSA. MASSA implements a flexible pipeline of Bioinformatics tools to gather data related to potential candidates for the annotation, and encapsulates INFAES for the inference stage. This architecture provides flexibility to integrate new tools and the required knowledge to manage them.

INFAES performance was validated and assessed using the metrics described in CAFA [1]. The 866 sequences from this large-scale experiment were processed with MASSA and INFAES, and the results compared to the 10 top-performing algorithms described in CAFA. INFAES got a F_{max} above 0.8 (out of 1) for the two GO categories tested (*BP* and *MF*), outperforming the best algorithms presented in CAFA, what is a very encouraging outcome.

INFAES, and also the overall MASSA, are ongoing work, and there are several open issues that could improve them. Regarding INFAES, immediate work is targeted to enrich the KB with additional information. For instance, specific rules that take into account the different types of protein could provide more accurate annotations for some queries. This growth in the number of rules, along with different criteria to guide the annotation process, will lead to incorporate different types of *InferenceAgents*, each one containing instances of INFAES with different KBs. These agents will negotiate in order to suggest an agreed annotation. Research in agents have already addressed these negotiation issues [55]. We also intend to add semi-supervised learning to the ES, allowing it to calibrate scores according to the users' feedback. At the level of MASSAPipe, more resources can be added to feed INFAES with additional evidences. Examples of these resources are: SIFT [56], to identify conserved residues that affect function; TMHMM [57], for the prediction of transmembrane regions; and SignalP [58], for signal peptide prediction.

Acknowledgements

This work has been done in the context of the projects "Aquagenomics" (CSD2007-00002) funded by the Consolider-Ingenio 2010 program of the Spanish Ministry of Education and Science, and "Social Ambient Assisting Living - Methods (SociAAL)" (TIN2011-28335-C02-01) supported by the Spanish Ministry for Economy and Competitiveness. Also, we acknowledge support from the "Red Científico-Tecnológica en Ciencias de los Servicios" (TIN2011-15497-E) and the "Programa de Creación y Consolidación de Grupos de Investigación" (UCM-BSCH GR35/10-A).

References

[1] P. Radivojac, W.T. Clark, T.R. Oron, A.M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, G. Pandey, J.M. Yunes, A.S. Talwalkar, S. Repo, M.L. Souza, D. Piovesan, R. Casadio, Z. Wang, J. Cheng, H. Fang, J. Gough, P. Koskinen, P. Toronen, J. Nokso-Koivisto, L. Holm, D. Cozzetto, D.W.A. Buchan, K. Bryson, D.T. Jones, B. Limaye, H. Inamdar, A. Datta, S.K. Manjari, R. Joshi, M. Chitale, D. Kihara, A.M. Lisewski, S. Erdin, E. Venner, O. Lichtarge, R. Rentzsch, H. Yang, A.E. Romero, P. Bhat, A. Paccanaro, T. Hamp, R. Kaszner, S. Seemayer, E. Vicedo, C. Schaefer, D. Achten, F. Auer, A. Boehm, T. Braun, M. Hecht, M. Heron, P. Honigschmid, T.A. Hopf, S. Kaufmann, M. Kiening, D. Krompass, C. Landerer, Y. Mahlich, M. Roos, J. Bjorne, T. Salakoski, A. Wong, H. Shatkay, F. Gatzmann, I. Sommer, M.N. Wass, M.J.E. Sternberg, N. Skunca, F. Supek, M. Bosnjak, P. Panov, S. Dzeroski, T. Smuc, Y.A.I. Kourmpetis, A.D.J. van Dijk, C.J.F.t. Braak, Y. Zhou, Q. Gong, X. Dong, W. Tian, M. Falda, P. Fontana, E. Lavezzo, B. Di Camillo, S. Toppo, L. Lan, N. Djuric, Y. Guo, S. Vucetic, A. Bairoch, M. Linial, P.C. Babbitt, S.E. Brenner, C. Orengo, B. Rost, S.D. Mooney, I. Friedberg, A large-scale evaluation of computational protein function prediction, Nat. Methods 10 (3) (2013) 221–227.
[2] S.C. Potter, L. Clarke, V. Curwen, S. Keenan, E. Mongin, S.M.J. Searle, A. Stabenau, R. Storey, M. Clamp, The Ensembl analysis pipeline, Genome Res. 14 (2004) 934–941.
[3] P. Gouret, V. Vitiello, N. Balandraud, A. Gilles, P. Pontarotti, E.G. Danchin, FIGENIX: Intelligent automation of genomic annotation: expertise integration in a new software platform, BMC Bioinform. 6 (2005) 198.
[4] G.H.V. Domselaar, P. Stothard, S. Shrivastava, J.A. Cruz, A. Guo, X. Dong, P. Lu, D. Szafron, R. Greiner, D.S. Wishart, BASys: a web server for automated bacterial genome annotation, Nucl. Acids Res. 33 (2005) W455–W459.
[5] T.-W. Chen, R.-C.R. Gan, T.H. Wu, P.-J. Huang, C.-Y. Lee, Y.-Y.M. Chen, C.-C. Chen, P. Tang, FastAnnotator – an efficient transcript annotation web tool, BMC Genomics 13 (2012) S9.
[6] M.R. Aniba, S. Siguenza, A. Friedrich, F. Plewniak, O. Poch, A. Marchler-Bauer, J.D. Thompson, Knowledge-based expert systems and a proof-of-concept case study for multiple sequence alignment construction and analysis, Brief. Bioinform. 10 (2009) 11–23.
[7] F. Hayes-Roth, Rule-based systems, Commun. ACM 28 (1985) 921–932.
[8] R. Studer, V.R. Benjamins, D. Fensel, Knowledge engineering: principles and methods, Data Knowl. Eng. 25 (1998) 161–197.

[9] V. Zacharias, Development and verification of rule based systems – a survey of developers, in: N. Bassiliades, G. Governatori, A. Paschke (Eds.), Rule Representation, Interchange and Reasoning on the Web – International Symposium, RuleML 2008, Lecture Notes in Computer Science, vol. 5321, Springer Berlin Heidelberg, 2008, pp. 6–16.

[10] G. Weiss (Ed.), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press, 1999.

[11] D. Xavier, F. Morán, R. Fuentes-Fernández, G. Pajares, Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology, Expert Syst. Appl. 40 (2013) 3943–3952.

[12] G. Schreiber, H. Akkermans, A. Anjewierden, R. de Hoog, N. Shadbolt, W. Van de Velde, B. Wielinga, Knowledge Engineering and Management – The CommonKADS Methodology, MIT Press, Cambridge, MA, 2000.

[13] The JBoss Drools Team, Drools expert user guide, 2012 http://docs.jboss.org/drools/release/5.4.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf (Online: accessed 20.12.13).

[14] M. Remm, C.E.V. Storm, E.L.L. Sonnhammer, Automatic clustering of orthologs and in-paralogs from pairwise species comparisons, J. Mol. Biol. 314 (2001) 1041–1052.

[15] A. Sali, R. Glaeser, T. Earnest, W. Baumeister, From words to literature in structural proteomics, Nature 422 (2003) 216–225.

[16] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, M. Yannakakis, On the complexity of protein folding, J. Comput. Biol. 5 (1998) 423–465.

[17] A.L. Lehninger, D.L. Nelson, M.M. Cox, Lehninger Principles of Biochemistry, 5th edition, W. H. Freeman & Company, 2008.

[18] D. Lee, O. Redfern, C. Orengo, Predicting protein function from sequence and structure, Nat. Rev. Mol. Cell Biol. 8 (2007) 995–1005.

[19] H. Li, N. Homer, A survey of sequence alignment algorithms for next-generation sequencing, Brief. Bioinform. 11 (2010) 473–483.

[20] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, D.J. Lipman, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucl. Acids Res. 25 (1997) 3389–3402.

[21] A.G. Rust, E. Mongin, E. Birney, Genome annotation techniques: new approaches and challenges, Drug Discov. Today 7 (2002) S70–S76.

[22] C.E.V. Storm, E.L.L. Sonnhammer, Automated ortholog inference from phylogenetic trees and calculation of orthology reliability, Bioinformatics 18 (2002) 92–99.

[23] P. Jones, D. Binns, H.-Y. Chang, M. Fraser, W. Li, C. McAnulla, H. McWilliam, J. Maslen, A. Mitchell, G. Nuka, S. Pesseat, A.F. Quinn, A. Sangrador-Vegas, M. Scheremetjew, S.-Y. Yong, R. Lopez, S. Hunter, Interproscan 5: genome-scale protein function classification, Bioinformatics 30 (2014) 1236–1240.

[24] P. Flicek, M.R. Amode, D. Barrell, K. Beal, K. Billis, S. Brent, D. Carvalho-Silva, P. Clapham, G. Coates, S. Fitzgerald, L. Gil, C.G. Girón, L. Gordon, T. Hourlier, S. Hunt, N. Johnson, T. Juettemann, A.K. Kähäri, S. Keenan, E. Kulesha, F.J. Martin, T. Maurel, W.M. McLaren, D.N. Murphy, R. Nag, B. Overduin, M. Pignatelli, B. Pritchard, E. Pritchard, H.S. Riat, M. Ruffier, D. Sheppard, K. Taylor, A. Thormann, S.J. Trevanion, A. Vullo, S.P. Wilder, M. Wilson, A. Zadissa, B.L. Aken, E. Birney, F. Cunningham, J. Harrow, J. Herrero, T.J.P. Hubbard, R. Kinsella, M. Muffato, A. Parker, G. Spudich, A. Yates, D.R. Zerbino, S.M.J. Searle, Ensembl 2014, Nucl. Acids Res. 42 (2014) D749–D755.

[25] D.A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, E.W. Sayers, GenBank, Nucl. Acids Res. 41 (2013) D36–D42.

[26] NCBI Resource Coordinators, Database resources of the National Center for Biotechnology Information, Nucl. Acids Res. 41 (2013) D8–D20.

[27] The UniProt Consortium, Activities at the Universal Protein Resource (UniProt), Nucl. Acids Res. 42 (2014) D191–D198.

[28] A. Marchler-Bauer, C. Zheng, F. Chitsaz, M.K. Derbyshire, L.Y. Geer, R.C. Geer, N.R. Gonzales, M. Gwadz, D.I. Hurwitz, C.J. Lanczycki, F. Lu, S. Lu, G.H. Marchler, J.S. Song, N. Thanki, R.A. Yamashita, D. Zhang, S.H. Bryant, CDD: conserved domains and protein three-dimensional structure., Nucl. Acids Res. 41 (2013) 348–352.

[29] S. Hunter, P. Jones, A. Mitchell, R. Apweiler, T.K. Attwood, A. Bateman, T. Bernard, D. Binns, P. Bork, S. Burge, E. de Castro, P. Coggill, M. Corbett, U. Das, L. Daugherty, L. Duquenne, R.D. Finn, M. Fraser, J. Gough, D. Haft, N. Hulo, D. Kahn, E. Kelly, I. Letunic, D. Lonsdale, R. Lopez, M. Madera, J. Maslen, C. McAnulla, J. McDowall, C. McMenamin, H. Mi, P. Mutowo-Muellenet, N. Mulder, D. Natale, C. Orengo, S. Pesseat, M. Punta, A.F. Quinn, C. Rivoire, A. Sangrador-Vegas, J.D. Selengut, C.J.A. Sigrist, M. Scheremetjew, J. Tate, M. Thimmajanarthanan, P.D. Thomas, C.H. Wu, C. Yeats, S.-Y. Yong, InterPro in 2011: new developments in the family and domain prediction database, Nucl. Acids Res. 40 (2012) D306–D312.

[30] R.D. Finn, A. Bateman, J. Clements, P. Coggill, R.Y. Eberhardt, S.R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E.L.L. Sonnhammer, J.G. Tate, M. Punta, The Pfam protein families database, Nucl. Acids Res. 42 (2014) D222–D230.

[31] I. Letunic, T. Doerks, P. Bork, SMART 7: recent updates to the protein domain annotation resource, Nucl. Acids Res. 40 (2012) D302–D305.

[32] D.H. Haft, J.D. Selengut, R.A. Richter, D. Harkins, M.K. Basu, E. Beck, TIGRFAMs and genome properties in 2013, Nucl. Acids Res. 41 (2013) D387–D395.

[33] D. Kahn, C. Rezvoy, F. Vivien, Parallel large scale inference of protein domain families, in: ICPADS, IEEE, 2008, pp. 72–79.

[34] H. Mi, A. Muruganujan, P.D. Thomas, PANTHER in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees, Nucl. Acids Res. 41 (2013) 377–386.

[35] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, et al., Gene Ontology: tool for the unification of biology, Nat. Genet. 25 (2000) 25–29.

[36] J. Pavón, J.J. Gómez-Sanz, R. Fuentes, Agent-Oriented Methodologies, Idea Group Publishing, 2005, pp. 236–276.

[37] F. Bellifemine, A. Poggi, G. Rimassa, JADE: a FIPA2000 compliant agent development environment, in: Proceedings of the 5th International Conference on Autonomous Agents (AGENTS 2001), ACM, 2001, pp. 216–217.

[38] J.U. Pontius, L. Wagner, G.D. Schuler, The NCBI Handbook, National Center for Biotechnology Information, 2003.

[39] NCBI, NCBI Conserved Domain Database (CDD) Help, 2013 http://www.ncbi.nlm.nih.gov/Structure/cdd/cdd_help.shtml (Online: accessed 20.12.013).

[40] Welcome Trust Sanger Institute, GFF (General Feature Format) Specifications Document, 2012 http://www.sanger.ac.uk/resources/software/gff/spec.html (Online: accessed 20.12.2013).

[41] NCBI, Web BLAST page options, 2007 http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml (Online: accessed 20.12.13).

[42] D. Cozzetto, D.W.A. Buchan, K. Bryson, D.T. Jones, Protein function prediction by massive integration of evolutionary analyses and multiple data sources, BMC Bioinform. 14 (2013) S1.

[43] M. Falda, S. Toppo, A. Pescarolo, E. Lavezzo, B. Di Camillo, A. Facchinetti, E. Cilia, R. Velasco, P. Fontana, Argot2: a large scale function prediction tool relying on semantic similarity of weighted gene ontology terms, BMC Bioinform. 13 (Suppl. 4) (2012).

[44] P. Rice, I. Longden, A. Bleasby, EMBOSS: the European molecular biology open software suite, Trends Genet.: TIG 16 (2000) 276–277.

[45] L.B. Koski, M.W. Gray, B.F. Lang, G. Burger, AutoFACT: An automatic functional annotation and classification tool, BMC Bioinform. 6 (2005) 151.

[46] J. Azé, L. Gentils, C. Toffano-Nioche, V. Loux, J.-F. Gibrat, P. Bessières, C. Rouveirol, A. Poupon, C. Froidevaux, Towards a semi-automatic functional annotation tool based on decision-tree techniques, BMC Proc. 2 (2008) S3.

[47] E. Cadag, B. Louie, P.J. Myler, P. Tarczy-Hornoch, Biomediator data integration and inference for functional annotation of anonymous sequences, in: R.B. Altman, A.K. Dunker, L. Hunter, T. Murray, T.E. Klein (Eds.), Pacific Symposium on Biocomputing, World Scientific Publishing Co., 2007, pp. 343–354.

[48] S. Götz, J.M. García-Gómez, J. Terol, T.D. Williams, S.H. Nagaraj, M.J. Nueda, M. Robles, M. Talón, J. Dopazo, A. Conesa, High-throughput functional annotation and data mining with the Blast2GO suite, Nucl. Acids Res. 36 (2008) 3420–3435.

[49] K. Bryson, M. Luck, M. Joy, D.T. Jones, Agent interaction for bioinformatics data management, Appl. Artif. Intell. 15 (2001) 917–947.

[50] K. Decker, S. Khan, C. Schmidt, G. Situ, R. Makkena, D. Michaud, BioMAS: a multi-agent system for genomic annotation, Int. J. Coop. Inf. Syst. 11 (2002) 265–292.

[51] S. Möller, M. Schroeder, R. Apweiler, Consistent integration of non-reliable heterogeneous information resources applied to the annotation of transmembrane proteins, Comput. Chem. 26 (2001) 41–49.

[52] M. Punta, P.C. Coggill, R.Y. Eberhardt, J. Mistry, J. Tate, C. Boursnell, N. Pang, K. Forslund, G. Ceric, J. Clements, A. Heger, L. Holm, E.L.L. Sonnhammer, S.R. Eddy, A. Bateman, R.D. Finn, The Pfam protein families database, Nucl. Acids Res. 40 (2012) D290–D301.

[53] R.J. Brachman, H.J. Levesque, Knowledge Representation and Reasoning, Morgan Kaufmann, 2004.

[54] M.A. Andrade, N.P. Brown, C. Leroy, S. Hoersch, A. de Daruvar, C. Reich, A. Franchini, J. Tamames, A. Valencia, C. Ouzounis, C. Sander, Automated genome sequence analysis and annotation, Bioinformatics 15 (1999) 391–412.

[55] I. García-Magariño, J.J. Gómez-Sanz, J.R. Pérez-Agüera, A complete-computerised Delphi process with a multi-agent system, in: K.V. Hindriks, A. Pokahr, S. Sardina (Eds.), Programming Multi-Agent Systems, Lecture Notes in Computer Science, vol. 5442, Springer Berlin Heidelberg, 2009, pp. 120–135.

[56] P. Kumar, S. Henikoff, P.C. Ng, Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm, Nat. Protoc. 4 (2009) 1073–1081.

[57] A. Krogh, B. Larsson, G. von Heijne, E.L.L. Sonnhammer, Predicting transmembrane protein topology with a hidden markov model: application to complete genomes, J. Mol. Biol. 305 (2001) 567–580.

[58] T.N. Petersen, S. Brunak, G. von Heijne, H. Nielsen, SignalP 4.0: discriminating signal peptides from transmembrane regions, Nat. Methods 8 (2011) 785–786.

5.4. An analysis of tool requirements for collaborative annotation

5.4.1. Citation

An analysis of tool requirements for collaborative annotation. Daniela Xavier, Rubén Fuentes-Fernández. Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Calabria, 6-8 May 2015.

Xavier, D. and Fuentes-Fernández, R. An analysis of tool requirements for collaborative annotation. In *Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (edited by G. Fortino, W. Shen, J.-P. Barthès, J. Luo, W. Li, S. Ochoa, M.-H. Abel, A. Guerrieri and M. Ramos), pages 212-217. IEEE, Calabria, Italy, 2015.

5.4.2. Abstract

In the last years, the amount of genomic data produced has increased significantly. These data need to be processed and associated to biological information in order to make a meaningful use of them. In the case of functional annotation, automated tools support these tasks, but their results have to be curated by experts, what creates a bottleneck. Although some community annotation efforts have attempted to address this issue, they have not been overly successful. This paper carries out a social analysis of these efforts through the Activity Theory framework, looking for their limitations and potential solutions for them. The conclusions of this analysis are used to propose requirements for an actual annotation tool, MASSA (Multi-Agent System to Support functional Annotation). The case study reports a preliminary evaluation of their implementation and the analysis of the tool performance according to a state-of-the-art benchmark.

5.4.3. References

(Friedberg, 2006), (Schnoes et al., 2009), (Reeves et al., 2009), (Mazumder et al., 2009), (Waldrop, 2008), (Leontiev, 1978), (Xavier et al., 2014), (Elsik et al., 2006), (Reinhard et al., 1994), (Engeström, 1987), (Gustin, 1973), (Arias et al., 2000), (Miao et al., 2000), (Portela et al., 2012), (Radivojac et al., 2013).

Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)

An analysis of tool requirements for collaborative annotation

Subtitle as needed (*paper subtitle*)

Daniela Xavier
GARP (Genomic and RNA Profiling Core),
Department of Molecular and Human Genetics
Baylor College of Medicine
Houston, United States
xavier@bcm.edu

Rubén Fuentes-Fernández
Department of Software Engineering and Artificial Intelligence
Universidad Complutense de Madrid
Madrid, Spain
ruben@fdi.ucm.es

***Abstract*—In the last years, the amount of genomic data produced has increased significantly. These data need to be processed and linked to biological information in order to make a meaningful use of them. Automated tools support these tasks, but their results have to be curated by experts, what creates a bottleneck. Although some collaborative *community annotation* efforts have attempted to address this issue, they have not been overly successful. This paper carries out a social analysis of them and their tools through the Activity Theory framework, looking for limitations and potential solutions. Its conclusions are used to propose requirements for an actual annotation tool, MASSA (Multi-Agent System to Support functional Annotation), which works in *functional annotation* (i.e., assigning biological functions). The case study reports a preliminary evaluation of their implementation and the analysis of the tool performance according to a state-of-the-art benchmark.**

***Keywords*—genomic data; annotation; functional annotation; community annotation; social analysis; Activity Theory; tools requirements; multi-agent system; MASSA.**

I. Introduction

With the advent of new sequencing techniques, the amount of genomic data produced in the past years has been growing significantly. However, these data are generally meaningless if they are not processed and associated with biological information [1].

Annotation is a complex, labor-intensive, and time-consuming task that aims to assign biological features to genomic sequences. For instance, *functional annotation* focuses on biological functions. There are multiple tools to support the annotation process, but a curator manually makes the final decision in order to have an accurate outcome. The curator is an expert who detains the proper knowledge on Biology and tools. This practice creates a bottleneck due to the great volume of data and the reduced number of curators. Aiming to avoid it, some automatic approaches have been proposed, but their outcomes are less precise [2]. Given the importance of annotation results for further works, researchers have been promoting the creation and support of high-quality manually curated databases [2].

Community annotation [3] [4] has been proposed as a possible solution to these issues. This approach aims to encourage the involvement of users of the annotated results as experts in the annotation process. It can adopt different organizations [4], such as supervised/unsupervised dispersed, jamboree, or student workshop. Although it seems promising and useful to solve the curator bottleneck, the efforts made until now have failed in different ways. The difficulties to involve a critical mass of users [4] and their short-term commitment [3] frequently impair this approach. Apparently, only those cases limited in time and with instant rewards seem to be successful [3]. Nevertheless, there have been just a few attempts [3] [5] to extract knowledge related to social organization and tool support from these experiences, so there are not guidelines for future ones.

In this work, we analyze these community annotation issues in the light of the Activity Theory (AT) [6], aiming to identify the reasons for such problems and possible solutions. The AT philosophical and analytical framework from Social Sciences revolves around the concept of *activity*. An *activity* is a human process where the individual and social levels are considered interconnected, as well as its changes over time. According to this theory, the tensions (i.e., *contradictions*) present in every human setting propel its evolution.

The findings of the previous analysis are used to propose requirements for annotation tools. This is an effective way of proposing to communities actual mechanisms to evaluate that could improve their work. Moreover, the pervasive use of tools in these tasks allows using them to promote useful practices for this type of annotation.

This work uses MASSA (Multi-Agent System to Support Functional Annotation) [7] as the annotation tool to illustrate these requirements. There are two main reasons for this choice. First, MASSA is a Multi-Agent System (MAS), so its design already considers issues of distribution and knowledge management that also appear in community annotation. Second, its development is based on requirement elicitation techniques and well-known methodologies for MASs and Expert Systems (ESs). This facilitates understanding its design decisions and modifying the system if needed.

978-1-4799-2001-3/15/$31 ©2015 IEEE

The case study reports the evaluation of a set of sequences from three organisms with a modified version of MASSA. Several experts were asked to assess the implemented functionality for community annotation in that context.

The rest of the paper includes the following sections. Section II introduces community annotation, and Section III the main concepts from AT used in its analysis. Then, Section IV describes the contradictions present in the current setting of this type of annotation, and their potential solutions. These are evaluated in the case study in Section V. Finally, Section VI discusses the conclusions and future work.

II. COMMUNITY ANNOTATION

Community annotation [4] is a collaborative task involving mainly users of databases with annotated gene sequences, but also in smaller numbers expert curators and tool designers. Its purpose is distributing the annotation effort among a broad base of annotators coming from users. Curators only review critical and conflicting entries, or random ones to check quality. This division of tasks should remove the bottleneck that is the reduced number of expert curators working with those databases. The actual way of organizing the effort varies among the variants of this type of annotation [4], e.g., supervised and unsupervised dispersed community annotation, jamborees, and student workshops.

The reasoning behind these efforts considers that as users are the main beneficiaries of these data, they will be willing to collaborate in their improvement. As most of these users are experts that use annotations in their domains, they have knowledge to annotate and review some entries. However, the actual results of these efforts show a partial success [3]. Only the types limited in time, and with immediate benefits for participants, are clearly successful. Therefore, the approach fails to set up a large community of users committed for a long period, which they need to become expert annotators.

Literature has discussed the reasons for this failure. However, the findings are not conclusive, and they are even sometimes contradictory. Some works [5] have pointed out the lack of recognition of this labor in the community. However, others [4] found this is not as important for users as knowing they are participating in a relevant challenge for Mankind and Science. In the same way, the availability of users' time to devote to this task is seen in some cases as one of the main obstacles [4], but it is minimized in others [8].

Special consideration in this analysis deserve the resources supporting annotation. Researchers need to work with large sets of data (mainly sequences) and bodies of knowledge (e.g., applicable literature, terms from ontologies for annotation, or protein domains), scattered among multiple sources. The preliminary repetitive processing of this information is only possible with the help of tools. These allow [3][7], for instance, comparing sequences, predicting transmembrane regions, or browsing data according to their biological features.

Community annotation also uses resources to enable collaboration. Examples of these [5] [8] are those intended to share knowledge on tasks and homogenize results (e.g., Standard Operating Procedures (SOPs) and examples of annotated sequences), and those that facilitate sharing other collaborative resources (e.g., wikis or e-mail lists).

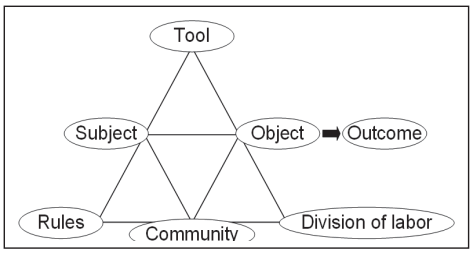
Both groups of resources have shown relevant limitations when applied in settings of community annotation. The former is focused on the work of the individual annotator, so it does not facilitate the collaborative construction of annotations and the related knowledge. In the second group, there are general collaborative tools not customized for annotation, or artefacts that are difficult to integrate in semi-automated and distributed workflows. However, an effective support for collaborative tasks demands the features of these resources being directly related to the different aspects of the considered teamwork [9].

III. ACTIVITY THEORY

The Activity Theory (AT) [6] is a framework from Social Sciences for the analysis of people behavior. Its main unit of analysis is the *activity*, which is a human action in a social context over time. AT regards every human activity as carried out by individuals. These and the artefacts (physical or mental) they use are the results of their societies or groups acting in an environment. As a consequence of these actions, people change their environment and their artefacts.

The context of an activity is known as its *activity system*. Fig. 1 shows its traditional depiction in AT studies [10]. Every vertex of the triangles is labeled with one of the main conceptual categories of this context. Triangles represent *mediation relationships*. For instance, the *community* is said to *mediate* (i.e. organize or affect) the relationship between *subjects* and *objects*.

Fig. 1. AT depiction of an activity system [10].



The individual level of the context corresponds to the upper triangle of the figure. The *subject* is the active element of the activity system. S/he executes the activity in order to satisfy certain relevant needs. The *outcome* is the result of the activity, and it is able to satisfy those needs. It is obtained from the transformation of *objects* using *tools*.

The social level is characterized by the entities in the lower triangles. The *community* is the set of subjects related directly or indirectly to the object, or more widely to the overall activity. The *division of labor* establishes the organization of the community in the activity. *Rules* represent social artefacts with influence in the activity. Their origin is outside the activity itself, for instance in culture, religion, or country laws. Although their origin and scope is different, both the division

of labor and rules comprehend similar artefacts, such as norms, laws, culture, prejudices, or power relationships.

Activity systems are not isolated units. They are interconnected by shared artefacts. For instance, the outcome of an activity system can be a paper that becomes a tool in an instance of the activity system for community annotation.

The analysis of the evolution of these networks of activity systems is organized around the concept of *contradiction*. Contradictions are tensions between elements in these networks. They are classified according to the components of activity systems involved in them. *Primary contradictions* appear in components that need to satisfy contradictory goals, or between different components playing the same role in the activity system. For instance, annotators competing for relevance can harm the community effort. *Secondary contradictions* emerge between components playing different roles in the activity system, usually because one of them does not meet the expected features. An example is the tensions between the division of labor of collaborative effort and rules for funding based on individual results. *Tertiary contradictions* happen between an activity system and its evolutions. These could appear between traditional and community-based annotation activities competing to attract users and annotators. Finally, *quaternary contradictions* are those among activities in a network. For instance, a learning activity that is not producing annotators with suitable profiles to participate in an activity of community annotation is in conflict with the latter.

Subjects in a network must face its contradictions. In order to fix them, they try to change the network, but at the same time, they start processes that will produce new tensions. This continuous dynamics causes evolution.

IV. A Social Analysis of Community Annotation

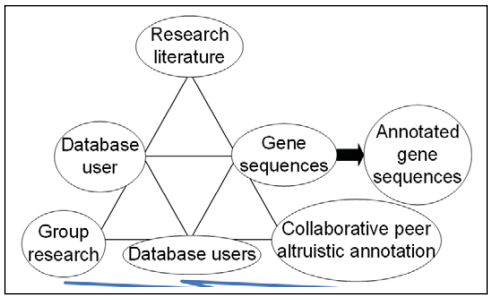
Community annotation is in conflict with some practices and resources of traditional annotation activities. This section analyzes these contradictions and their potential solutions using the AT. The study focuses on issues appearing around the annotation community (see Section IV.A) and its resources (see Section IV.B).

A. The Annotation Community

The advances in automated sequencing techniques have produced a vast amount of data. The main concern for Bioinformatics and related disciplines is becoming that the meaningful use of these data requires their annotation [2]: sequences have to be linked to additional information on, for instance, their structure and biological function, according to research literature. The scarcity of experts able to carry out these tasks, and the limitations of automated support tools, have led to look for alternatives to traditional models of annotation. Community annotation [8] is one of them. It tries to engage users of annotated sequences as long-term part-time annotators, as they are experts and the main beneficiaries of the results. However, these efforts have failed in this goal largely.

Fig. 2 shows the activity system corresponding to this situation. It focuses on the activity for community annotation and the secondary contradiction around its community.

Fig. 2. Secondary contradiction in the activity system for community annotation regarding its community.



Regarding the individual level of the activity system, the main active subjects of this collaborative task are the *database user*s, who should become annotators that help in the addition of new knowledge. Expert curators can be taken into account through communities (they share objects with users) or neighbor activities (they curate the outcome of this activity). The objects transformed in this process are the *gene sequences* in databases. Annotators enrich them by linking entries to biological information, tied back when possible to research literature. There are multiple tools applied in this process, but the key one here is the *research literature* containing the knowledge (e.g., on Biology or Genetics) to annotate. It constrains relevant parts of the process, as suitable annotators need to have an educational level and experience that allow them to understand, abstract, and codify that knowledge in a suitable way for databases.

At the social level, the community includes *database users*, and possibly expert curators. The division of labor describes the rules of this annotation effort, which are determined mainly by the type of organization adopted for community annotation. There are variations among the different types, but in all of them the long-term commitment of users is voluntary. This is characterized as *collaborative peer altruistic annotation*. In this case, the most immediate rules come from the neighbor research activities, characterized by the *group research* element. Researchers, as members of the community of database users, are the potential annotators. However, rules highlight that annotation is not their work, and participating in it diverts effort from their main tasks [4].

Fig. 2 depicts the secondary contradiction in this activity system as a lightning bolt between its division of labor and rules. For the *database user* subjects that participate in the annotation process, there is a conflict between participating in community annotation efforts (following the norms of the division of labor) and making their usual research activities (according to their rules). The *collaborative peer altruistic annotation* division of labor corresponds to communities of people with the same hierarchical power, pursuing a common goal, and not obtaining an immediate personal benefit of the task. An example of this is the peer-review of scientific literature [4]. On the other side, the *group research* rules correspond to a context where researchers are devoted to their

own group research, and focused on results that can bring funding to their groups. In these research activities, databases are tools, and developing them is subordinated to other tasks.

From the AT perspective, these tensions correspond to an *exchange value* contradiction [6]. This appears when a subject produces an outcome for the community, but the subject does not perceive any contribution of that outcome to one of her/his needs. Thus, there is a decoupling between the activity and its objectives. The common solution to this contradiction is introducing a feedback loop that allows the community activities contributing to the subject's objectives.

Literature offers advice on the kind of outcomes that can be useful to reward subjects in this context of research and collaborative efforts. Studies on the motivation of scientists [11] show the strong impact of recognition (e.g., through promotion in job, awards to their research, and citations to their works and contributions) and charisma (e.g., being involved with researchers, projects, or activities perceived as highly relevant in their field or even for Mankind).

Tools for community annotation can contribute to support those social patterns in several ways. A credit-assignment system that acknowledges authors' contributions is the basic element [5]. Such systems are already present in research literature (i.e., paper authorship) and peer production (e.g., wikis contributors). This mechanism makes possible citing contributions, which in turn is a useful mechanism to give recognition to authors and to trace annotation sources. Traceability would allow following chains of related annotations, which facilitates their correction when researchers detect misannotations [2]. Highly recognized contributors could also be proposed for relevant roles in the community, with extra capabilities in the organization or permissions in tools. These rankings are already applied in communities like the Wikipedia[1], where contributors consider them when voting who should play a given responsibility.

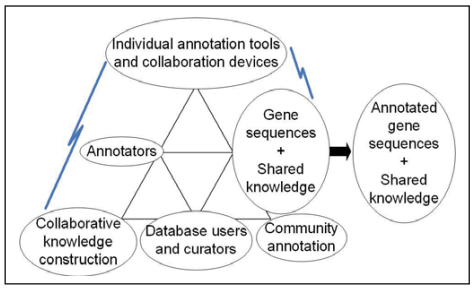
B. Resources

Community annotation makes use of a variety of resources. Some of them support the individual tasks of annotators, and others have been adopted to facilitate the collaborative work. In both cases, they have not been specifically designed for that setting. Community annotation is a process of collaborative knowledge construction. It requires resources to share and work in the community over the knowledge to build, its background, and the rationale behind this construction [12]. These features are missed or barely considered currently.

For this case, the activity system also corresponds to community annotation (see Fig. 3). However, this perspective is focused on its tools and the related secondary contradictions.

The subjects are the *annotators*, who can be database users collaborating in the annotation or expert curators. As a process of collaborative knowledge construction, there are two types of object. The primary one, shared with the previous perspective on community annotation (see Section IV.A), is the *gene sequences* to annotate. They are the main component of the *annotated gene sequences* that are the key outcome of this activity. The other object is the *shared knowledge* that subjects use and build to enable their collaborative effort. It comprehends resources such as literature, operation procedures, or descriptions of the steps followed to annotate a given entry. Tools should support manipulating that *shared knowledge* (with the *collaboration devices*) and the *annotated gene sequences* (mainly using the *individual annotation tools*). The division of labor that organizes the community of annotators (i.e., *database users and curators*) depends on the type of *community annotation* adopted [4]. The relevant rules here are those of activities of *collaborative knowledge construction*. They require procedures and tools to work jointly with the description and solution of the problem [12].

Fig. 3. Secondary contradiction in the activity system for community annotation regarding its support.



The secondary contradiction in this activity system is an instance of the *contradictory information* problem [13]. This appears when the artifacts that need to support the collaborative effort do not meet these requirements. They need to be redesigned in order to integrate properly the different perspectives. Their functionality should enable the discussion on the contradictory shared artifacts, the synthesis of different perspectives, and the representation of this process and facets.

Subjects in the activity systems of community annotation need at least to share information on the sequences to annotate, the information linked to them, and the process followed to assign it. This process can adopt standard procedures and use automated tools that access databases of heterogeneous information [1] [8]. The application of tools can be easily described from their actual inputs and outputs. In the case of procedures, communities frequently develop SOPs and examples to get results of homogenous quality [8]. Experts in the community develop these resources and review them according to the new knowledge and experience gained in the annotation effort.

The previous way of working makes difficult the integration in the shared artefacts of the different perspectives in the community. First, experts mainly describe resources (i.e., SOPs and examples) in natural language. This reduces the availability of automated guidance in their application, and their interpretation by individuals is source of inconsistencies and misunderstandings. Other fields have adopted formal modeling languages to describe their workflows in order to

[1] http://www.wikipedia.org/

deal with this problem. Examples of this approach are the pervasive use in Software Engineering of modeling languages for workflows and processes [13] [14]. Nevertheless, it would be important to use specialized languages easy to understand for people involved in annotation. Moreover, the use of well-defined modeling languages will facilitate the development of automated *wizard* tools to guide and assist in their processes. Examples of these wizards appear, for instance, in integrated development environments like Eclipse to guide programmers in artefact creation [15]. Second, all the members of the community should be able to contribute to the analysis and description of the shared knowledge [12]. This is not only an issue of having means to describe the information, but also that they must be accessible to the different profiles and backgrounds in the community. For instance, experts develop SOPs for target novices, but only these really know their problems and doubts in the annotation tasks. Thus, the means to describe procedures need to be integrated in the usual activities of users of different expertise levels, so they can incorporate the actual users' experience [12].

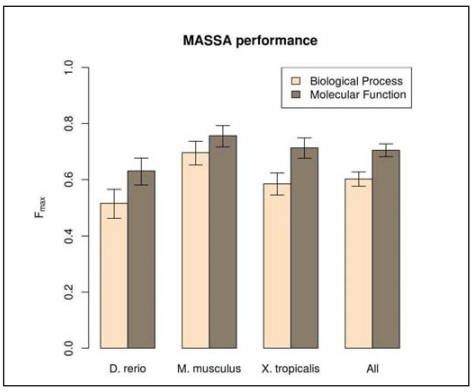
Annotation tools can incorporate the previous functionality in different ways. For instance, tools can monitor users' activities in order to gain insights on their actual behavior. The design of workflows can be made with collaborative and online design tools where multiple users can contribute.

V. Case Study: MASSA Evaluation

In order to evaluate the previous findings, this work considers the features and performance of an existing tool for individual annotation, MASSA [7]. It is a MAS that includes two subsystems: a flexible pipeline of basic annotation tools that provides raw data and access to databases, and a Rule-Based Expert System (RBES) that infers the annotation of sequences from the previous data. Both subsystems share a result database for the intermediate data of the annotation process. Users communicate with the system through *interface agents* that manage the graphical interface. They introduce in this interface the gene sequences to annotate, and the system reports there the result. MASSA can manage simultaneously multiple requests and their information as *projects*.

The requirements from the social analysis of community annotation (see Section IV) considered here are the following. For the contradictions regarding the community (see Section IV.A), the system needs to allow the participation of multiple users and the recognition of their contributions. In this case, MASSA is already a multi-user system. For the second requirement, the system will record with the modifications of annotations the user who did them and their timestamp. This authoring information can be exported to a file and imported in a spreadsheet software to show statistics on MASSA use and contributions. For the contradictions regarding resources (see Section IV.B), the rule engine of the RBES can provide a log of the activation of the decision rules in its knowledge base. It includes their identifier, order of activation, parameters, and new information produced. Experts are informed of the weights, thresholds, and scores of these rules, and they can adjust them. This information is stored and aggregated in order to support experts in the evaluation of the system setup over multiple requests.

Fig. 4. MASSA performance evaluation based on CAFA's benchmark. A bootstrap on the set of query sequences, with n = 10,000 iterations, was used to calculate the Confidence Intervals (95%).



The modifications and performance of MASSA are tested with an experiment that involves the annotation of multiple sequences from different organisms. The annotation is organized in two rounds. Experts have a first round to adjust the values in rules. This round follows the experiment in [7]. The second round is to evaluate the system performance and is described next.

The second round considers 661 random sequences from three phylogenetic different organisms (*Xenopus tropicalis* (241), *Mus musculus* (200), and *Danio rerio* (220)). These were initially processed with a MASSA pipeline that includes BLASTX against the Non-Redundant protein sequences database and UniProt, RPS-BLAST against Conserved Domains Database, InterProScan, and Orthostrapper. More information on these tools and databases of the MASSA pipeline can be found in [7]. The data from different species were processed separately. All the sequences related to the organisms in the experiment were removed from the respective databases, enabling a more realistic experiment.

The MASSA outcome was evaluated following the procedure of the Critical Assessment of Functional Annotation (CAFA) experiment [16]. It requires that the annotation process assigns Gene Ontology (GO) terms to each of the analyzed entries. Then, these are grouped by their GO category in Molecular Function (MF) or Biological Process (BP). From this information, the method evaluates the performance of annotation tools based on *precision-recall* and *F-measure*, and calculates a single value, F_{max}, that allows comparisons between methods.

MASSA obtained in the experiment an average F_{max} of 0.6 and 0.7 for MF and BP, respectively (see Fig. 4). Considering the fact that a perfect prediction has a $F_{max} = 1$, and that MASSA outperformed the 10 top-performing algorithms presented in CAFA[16] [16], results are very encouraging.

Regarding the evaluation of features to address the contradictions, the participant experts were interviewed. The

information on the authors of annotations and users proposing values for rules was regarded as necessary to allow discussion when differences arise. Additional information justifying the choice of the value, or the change of the previous one, was viewed as a must. Experts advocated for integrated tool functionality to discuss these issues without abandoning the annotation workflows. There were also concerns about how to guarantee the correct attribution of work, particularly as information evolves over time. A history of changes, as that of wikis, was pointed out as a possible solution. The information provided by tools on their actual functioning was considered useful. The kind of step-by-step description of the process in the RBES was also validated as a potential description for annotator workflows. It would allow to the rest of the community knowing which decisions were made when annotating a given sequence. All the experts missed a tighter integration of tools with the annotation procedures and workflows, which could provide more guidance in the process. Wizards, as those available in development tools like Eclipse [15], were perceived as too rigid for the annotation task. Experts regarded the process as a graph of tasks with recommendations for each one and requirements (i.e., conditions or artefacts) to enable them.

VI. Conclusions and Future Work

This paper has analyzed community annotation [4], and the reasons behind its partial failure on building stable communities of annotator users. This study has been based on the sociological framework of AT [6].

The analysis of the activities involved has shown problems at two different levels. First, the community of database users lack of incentives to participate in the process. Successful efforts in other domains have available mechanisms to reward contributions, mainly using different forms of recognition. Second, this kind of activity is a process of co-construction of knowledge. Such activities require means to externalize their results and the process that generates them. Current resources do not consider these aspects, so that information becomes an individual asset or requires important efforts to share it.

The previous problems are largely due to an approach to resource development focused on traditional forms of annotation. This conflict constitutes itself a form of quaternary contradiction between the development and community annotation activities. An effective support to community annotation requires a higher involvement of its users in the definition and evolution of those resources.

The previous findings have been addressed through the recommendations found in AT. The result has been a set of potential requirements for annotation tools that this work has illustrated with modifications to MASSA [7]. Experts have evaluated these features in an experiment with positive comments. Nevertheless, they considered that higher levels of integration between tools and processes are needed in order to facilitate annotation tasks. In particular, the formal definition of workflows and support tools for them, and their collaborative development in the community, is future work.

Acknowledgment

This work has been done in the context of the projects "Aquagenomics" (CSD2007-00002) funded by the Consolider-Ingenio 2010 program of the Spanish Ministry of Education and Science, "Social Ambient Assisting Living - Methods (SociAAL)" (TIN2011-28335-C02-01) supported by the Spanish Ministry for Economy and Competitiveness, and "Programa de Creación y Consolidación de Grupos de Investigación" (UCM-BSCH GR35/10-A).

References

[1] I. Friedberg, "Automated protein function prediction – The genomic challenge," Briefings in Bioinformatics, vol. 7(3), pp. 225-242, 2006.

[2] A. M. Schnoes, S. D. Brown, I. Dodevski, and P. C. Babbitt, "Annotation error in public databases: misannotation of molecular function in enzyme superfamilies," PloS Computational Biology, vol. 5(12), pp. e1000605, 2009.

[3] G. A. Reeves, D. Talavera, and J. M. Thornton, "Genome and proteome annotation: organization, interpretation and integration," Journal of the Royal Society Interface, vol. 6(31), pp.129-147, 2009.

[4] R. Mazumder, D. A. Natale, J. A. E. Julio, L.-S. Yeh, and C. H. Wu, "Community annotation in biology," Biology Direct, vol. 5, pp. 12, 2009.

[5] M. Waldrop, "Big data: wikiomics," Nature, vol. 455(7209), pp.22-25, 2008.

[6] V. Kaptelinin and B. A. Nardi, Acting with technology - Activity theory and interaction design. Cambridge, MA, USA: MIT Press, 2006.

[7] D. Xavier, B. Crespo, R. Fuentes-Fernández, and J. J. Gómez-Sanz, "MASSA: Multi-agent system to support functional annotation," in Advances in Practical Applications of Heterogeneous Multi-Agent Systems - The PAAMS Collection, Lecture Notes in Computer Science, vol. 8473, Y. Demazeau, F. Zambonelli, J. M. Corchado, and J. Bajo, Eds. Springer-Verlag, 2014, pp. 291-302.

[8] C. Elsik, K. C. Worley, L. Zhang, N. V. Milshina, H. Jiang, J. T. Reese, K. L. Childs, A. Venkatraman, C. M. Dickens, G. M. Weinstock, and R. A. Gibbs, "Community annotation: procedures, protocols, and supporting tools," Genome Research, vol. 16(11), pp. 1329-1333, 2006.

[9] W. Reinhard, J. Schweitzer, G. Völksen, and M. Weber, "CSCW tools: Concepts and architectures," Computer, vol. 27(5), pp., 28-36, 1994.

[10] Y. Engeström, Learning by expanding - An activity-theoretical approach to developmental research. Helsinki, Finland: Orienta-Konsultit, 1987.

[11] B. H. Gustin, "Charisma, recognition, and the motivation of scientists," American Journal of Sociology, vol. 78(5), pp. 1119-1134, 1973.

[12] E. Arias, Ernesto, H. Eden, G. Fischer, A. Gorman, and E. Scharff, "Transcending the individual human mind - creating shared understanding through collaborative design," ACM Transactions on Computer-Human Interaction, vol. 7(1), pp. 84-113, 2000.

[13] Y. Miao, S. Holst, T. Holmer, J. M. Fleschutz, and P. Zentel, "An activity-oriented approach to visually structured knowledge representation for problem-based learning in virtual learning environments," in Designing Cooperative Systems- The Use of Theories and Models, Proceedings of the 4th International Conference on the Design of Cooperative Systems (COOP 2000), R. Dieng, A. Giboin, L. Karsenty, and G. De Michelis, Eds. IOS Press, 2000, pp. 303-320.

[14] C. Portela, A. Vasconcelos, A. Silva, A. Sinimbú, E. Silva, M. Ronny, W. Lira, and S. Oliveira, "A comparative analysis between BPMN and SPEM modeling standards in the software processes context," Journal of Software Engineering and Applications, vol. 5(5), pp. 19428, 2012.

[15] J. D'Anjou, The Java developer's guide to Eclipse. Boston, MA, USA: Addison-Wesley Professional, 2005.

[16] A. M. Lisewski, S. Erdin, E. Venner, O. Lichtarge, R. Rentzsch, H. Yang, et al., "A large-scale evaluation of computational protein function prediction," Nature Methods, vol. 10, pp. 221-227, 2013.

Chapter 6

Conclusions and future work

The End is not only an End,
This is the Beginning of a new Start.

Gaurav GRV Sharma

This final chapter summarizes the research and highlights the main contributions. It also presents the next lines of research and future work.

6.1. Main contributions

This work presents MASSA, a system developed to produce accurate functional annotations and to address some key issues of the annotation problem. It was designed based on the requirements obtained in the study of the domain, the problem, and the proposed solutions available, both with a review of literature and interviewing experts specifically for this work. Besides supporting annotation tasks, the overall design of the system was aimed at facilitating its usage, understanding, maintenance, and evolution. These are key features given the fast pace of evolution in the domain, that affects the considered knowledge, available tools, and even the organization of work among experts.

MASSA proposes an architecture that integrates a RBES in a MAS. This allows improving the quality of the inference and dealing with the domain singularities.

The MAS is composed of two sub-systems: MASSAPipe and MASSAInference. The first one is responsible for executing supporting tools to gather information about potential candidates. The second one processes this information and infers the annotation using the RBES. Currently MASSAPipe supports some of the most popular tools (e.g., BLAST (Altschul et al., 1997), InterProScan (Jones et al., 2014), and RPS-BLAST (Altschul et al., 1997)) and data sources (e.g., NR, UniProtKB (Magrane and Consortium, 2011), and CDD (Marchler-Bauer et al., 2013)) in this domain. Moreover, the pipeline is flexible enough to handle any database that can be formatted as a SDB, and it is possible to create custom pipelines by adding/removing databases and tools. MASSAPipe also

has an agent in charge of updating the SDBs. This functionality is essential as data are frequently updated.

The RBES mimics the expert reasoning process at the inference stage, which allows generating more precise outcomes and reducing the expert workload. The system combines the basic knowledge (i.e., homology, domain/family identification, and GO terms) the majority of systems apply with orthology and additional expertise that can improve the annotation quality. The way the knowledge is structured pursues facilitating its understanding and that of the annotation process. It is described with rules based on scores. These scores are calculated from the basic information provided by the pipeline. They summarize expert judgment on the relevance of features, their values, and their relationships. Using scores also facilitates exploring alternative reasoning workflows.

Besides, MASSA can deal with any kind of input, regardless its context or related specie. The system also presents the annotation results in a very descriptive and complete way, intelligible for both human and machines. The output is provided in different formats (e.g., GenBank (see Sectio A.1.2), GFF (see Section A.3.1), and FASTA (see Section A.2.1), so the expert can choose the one that suits best her/his needs.

MASSA also integrates some community annotation requirements. Among other features, it is a multi-user system and provides a log of its reasoning process with author and timestamp information.

These features allow meeting the previously mentioned requirements of usability and maintenability. They decouple the pipeline that uses existing tools to provide preliminary data from the part that applies the expert knowledge. The RBES and the internal organization of its KB are intuitive for experts, so they can contribute effectivey to knowledge management. Its explanation facility gives a description of all decisions made. On its side, the agent approach eases overtaking the environmental hurdles of the domain (e.g., distribution, heterogeneity, and constant evolution), and integrating knowledge management. Tool dependencies are described using FSMs to facilitate integration. This overall approach allows also organizing the system in a very intuitive way for experts, similar to their own processes. Thereby, they can have a better comprehension of the whole annotation process.

The combination of MASs and RBESs seems to be quite suitable and advantageous for several Bioinformatics problems, although not very explored. Thus, this work intends to encourage the application of similar strategies in the field.

This work also pays particular attention to the way of addressing the development process. Two state-of-the-art methodologies were applied: INGENIAS (Pavón et al., 2005) for the MAS and CommonKADS (Schreiber et al., 2000) for the RBES. Both approaches provide guidelines that not only facilitate the development process, but also the maintenance and extension. Also taking into account these concerns, the system was implemented using well-known paradigms and frameworks. Java was used to develop the system. This allows using two well-established frameworks, Jade (Bellifemine et al., 2001) for the MAS and Drools (The JBoss Drools Team, 2012) for the RBES. The decision of using these frameworks was taken, among other reasons, based on their widespread use and support and their portability.

MASSA was assessed following the benchmark proposed by CAFA (Radivojac et al., 2013). This evaluation yielded very encouraging results and showed that MASSA performs better than the 10 best preforming tools presented in this experiment. This assessment also shows the importance of having gold standards to follow when evaluating a system.

Besides proposing a solution to the annotation problem, this work aims to highlight some good practices that seem to be sometimes disregarded in the domain. First, in a constantly evolving domain, systems should be developed to be able to follow this progress. Second, well-developed systems are more robust, flexible, and easy to maintain and extend. This affects the system design, but also how it is developed. Software Engineering has many guidelines that facilitate the system development, maintenance, and evolution. Knowledge Engineering comprises methodologies to obtain, organize, structure, and model the knowledge in a way it can be easily understood by people involved in the process. Third, choosing well-supported paradigms and tools to implement the system is essential for its development and evolution. It gives the chance of having a community of people collaborating with system developers to support the required infrastructure.

6.2. Future lines of research and work

MASSA is an ongoing work and has several open lines for improvement. This section summarizes the main ones.

Regarding its knowledge, MASSA still has to incorporate support for additional resources. Some of them are tools to recognize conserved residues that affect function (e.g., SIFT (Kumar et al., 2009)), or to predict transmembrane regions (e.g., TMHMM (Krogh et al., 2001)) or peptide signal (e.g., SignalP (Petersen et al., 2011)). In line with this, new rules will be added to the KB. These will allow representing the expert knowledge still missed regarding the annotation process in general, and also the proper integration of these new resources.

The setup of the system can also be improved using some semi-automated techniques. Currently, experts set manually the scores of the KB. Semi-supervised learning can be used to calibrate these scores based on the users' feedback.

The RBES currenly just represents one possible inference process, though supported by the experience of multiple experts. There are sometimes differences in this process among experts. The inference agent could be specialized in different subtypes. Each subtype would work with its own KB to represent different perspectives of experts. This differentiation would require setting up some negotiation mechanism among agents that allows them arriving to a common (or at least most recommended) annotation.

Another issue is facilitating the pipeline customization by users. In this line, MASSA interface is going to be reformulated following the workflow management system trend, like in (Wolstencroft et al., 2013) and (Goecks et al., 2010). This approach has shown to improve system usability regarding the description of workflows.

The support for community annotation is still incomplete. The analysis carried out

points out the need of providing a reward mechanism to encourage the experts' collaboration. Besides, the system has to facilitate the exchange of information, and the standardization of annotations. In this line, an agent can be implemented to monitor users' activities in order to understand their behavior and decisions. This would allow acquiring more expertise to be crystallized as part of the KB and new requirements for the system. After incorporating these improvements, an extensive assessment of the features that support the community annotation will be put into practice.

Finally, the system and its source code will be made publicly available. Its functionality will also be provided as web-services. These initiatives are expected to promote the involvement of users, so extensive feedback can be obtained on the system features and knowledge. It would also allow sharing the current development experience with the community.

Part I

Appendix

Appendix A

File formats

Divide each difficulty into as many parts as is feasible and necessary to resolve it.

Rene Descartes

A.1. Database Files

A.1.1. EMBL format

UniProt (Magrane and UniProt Consortium, 2011) information can be retrieved in different formats: flat text (EMBL sequence format), FASTA (see Section A.2.1), XML, or RDF/XML [1]. MASSA was developed to connect to the UniProt database (Magrane and UniProt Consortium, 2011) and read the EMBL files related to target proteins. This format (see Format A.1) contains information about related publications, taxonomy, alternative names, cross-reference, domains and families, conserved sites, GO terms, and so on.

Format A.1: EMBL file for *Expansin-A7*[2]

```
ID  EXPA7_ARATH       Reviewed;     262 AA.
AC  Q9LN94; Q9LN88;
DT  05-MAR-2002, integrated into UniProtKB/Swiss-Prot.
DT  01-OCT-2000, sequence version 1.
DT  24-JUN-2015, entry version 107.
DE  RecName: Full=Expansin-A7;
DE         Short=AtEXPA7;
DE  AltName: Full=Alpha-expansin-7;
DE         Short=At-EXP7;
DE         Short=AtEx7;
DE  AltName: Full=Ath-ExpAlpha-1.26;
DE  Flags: Precursor;
GN  Name=EXPA7; Synonyms=EXP7; OrderedLocusNames=At1g12560;
GN  ORFNames=F5O11.30, T12C24.10, T12C24_8;
OS  Arabidopsis thaliana (Mouse-ear cress).
OC  Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
OC  Spermatophyta; Magnoliophyta; eudicotyledons; Gunneridae;
OC  Pentapetalae; rosids; malvids; Brassicales; Brassicaceae; Camelineae;
```

[1] http://www.uniprot.org/help/retrieve_sets. [Online: accessed 5-September-2015]

[2] http://www.uniprot.org/uniprot/Q9LN94.txt. [Online: accessed 5-September-2015]

```
OC  Arabidopsis.
OX  NCBI_TaxID=3702;
RN  [1]
RP  NUCLEOTIDE SEQUENCE [LARGE SCALE GENOMIC DNA].
RC  STRAIN=cv. Columbia;
RX  PubMed=11130712; DOI=10.1038/35048500;
RA  Theologis A., Ecker J.R., Palm C.J., Federspiel N.A., Kaul S.,
RA  White O., Alonso J., Altafi H., Araujo R., Bowman C.L., Brooks S.Y.,
RA  Buehler E., Chan A., Chao Q., Chen H., Cheuk R.F., Chin C.W.,
RA  Chung M.K., Conn L., Conway A.B., Conway A.R., Creasy T.H., Dewar K.,
RA  Dunn P., Etgu P., Feldblyum T.V., Feng J.-D., Fong B., Fujii C.Y.,
RA  Gill J.E., Goldsmith A.D., Haas B., Hansen N.F., Hughes B., Huizar L.,
RA  Hunter J.L., Jenkins J., Johnson-Hopson C., Khan S., Khaykin E.,
RA  Kim C.J., Koo H.L., Kremenetskaia I., Kurtz D.B., Kwan A., Lam B.,
RA  Langin-Hooper S., Lee A., Lee J.M., Lenz C.A., Li J.H., Li Y.-P.,
RA  Lin X., Liu S.X., Liu Z.A., Luros J.S., Maiti R., Marziali A.,
RA  Militscher J., Miranda M., Nguyen M., Nierman W.C., Osborne B.I.,
RA  Pai G., Peterson J., Pham P.K., Rizzo M., Rooney T., Rowley D.,
RA  Sakano H., Salzberg S.L., Schwartz J.R., Shinn P., Southwick A.M.,
RA  Sun H., Tallon L.J., Tambunga G., Toriumi M.J., Town C.D.,
RA  Utterback T., Van Aken S., Vaysberg M., Vysotskaia V.S., Walker M.,
RA  Wu D., Yu G., Fraser C.M., Venter J.C., Davis R.W.;
RT  "Sequence and analysis of chromosome 1 of the plant Arabidopsis
RT  thaliana.";
RL  Nature 408:816-820(2000).
RN  [2]
RP  GENOME REANNOTATION.
RC  STRAIN=cv. Columbia;
RG  The Arabidopsis Information Resource (TAIR);
RL  Submitted (APR-2011) to the EMBL/GenBank/DDBJ databases.
RN  [3]
RP  NOMENCLATURE.
RX  PubMed=15604683; DOI=10.1007/s11103-004-0158-6;
RA  Kende H., Bradford K.J., Brummell D.A., Cho H.-T., Cosgrove D.J.,
RA  Fleming A.J., Gehring C., Lee Y., McQueen-Mason S.J., Rose J.K.C.,
RA  Voesenek L.A.C.;
RT  "Nomenclature for members of the expansin superfamily of genes and
RT  proteins.";
RL  Plant Mol. Biol. 55:311-314(2004).
CC  -!- FUNCTION: Causes loosening and extension of plant cell walls by
CC      disrupting non-covalent bonding between cellulose microfibrils and
CC      matrix glucans. No enzymatic activity has been found (By
CC      similarity). {ECO:0000250}.
CC  -!- SUBCELLULAR LOCATION: Secreted, cell wall. Membrane; Peripheral
CC      membrane protein.
CC  -!- SIMILARITY: Belongs to the expansin family. Expansin A subfamily.
CC      {ECO:0000305}.
CC  -!- SIMILARITY: Contains 1 expansin-like CBD domain.
CC      {ECO:0000255|PROSITE-ProRule:PRU00078}.
CC  -!- SIMILARITY: Contains 1 expansin-like EG45 domain.
CC      {ECO:0000255|PROSITE-ProRule:PRU00079}.
CC  -!- SEQUENCE CAUTION:
CC      Sequence=AAF88078.1; Type=Erroneous gene model prediction; Evidence={ECO:0000305};
CC  -!- WEB RESOURCE: Name=EXPANSIN homepage;
CC      URL="http://homes.bio.psu.edu/expansins/";
CC  ---------------------------------------------------------------------
CC  Copyrighted by the UniProt Consortium, see http://www.uniprot.org/terms
CC  Distributed under the Creative Commons Attribution-NoDerivs License
CC  ---------------------------------------------------------------------
DR  EMBL; AC025416; AAF79645.1; -; Genomic_DNA.
DR  EMBL; AC025417; AAF88078.1; ALT_SEQ; Genomic_DNA.
DR  EMBL; CP002684; AEE28897.1; -; Genomic_DNA.
DR  PIR; F86259; F86259.
DR  RefSeq; NP_172717.1; NM_101127.3.
DR  UniGene; At.42070; -.
DR  ProteinModelPortal; Q9LN94; -.
DR  SMR; Q9LN94; 35-261.
DR  STRING; 3702.AT1G12560.1; -.
DR  PaxDb; Q9LN94; -.
DR  PRIDE; Q9LN94; -.
DR  EnsemblPlants; AT1G12560.1; AT1G12560.1; AT1G12560.
DR  GeneID; 837813; -.
DR  KEGG; ath:AT1G12560; -.
DR  TAIR; AT1G12560; -.
DR  eggNOG; NOG119598; -.
DR  HOGENOM; HOG000220793; -.
DR  InParanoid; Q9LN94; -.
DR  OMA; PAFMKMA; -.
DR  PhylomeDB; Q9LN94; -.
DR  PRO; PR:Q9LN94; -.
DR  Proteomes; UP000006548; Chromosome 1.
DR  GO; GO:0005618; C:cell wall; IEA:UniProtKB-SubCell.
DR  GO; GO:0005576; C:extracellular region; IEA:UniProtKB-KW.
```

```
DR  GO; GO:0016020; C:membrane; IEA:UniProtKB-SubCell.
DR  GO; GO:0009664; P:plant-type cell wall organization; IEA:InterPro.
DR  GO; GO:0048767; P:root hair elongation; IMP:TAIR.
DR  Gene3D; 2.40.40.10; -; 1.
DR  Gene3D; 2.60.40.760; -; 1.
DR  InterPro; IPR014733; Barwin-like_endoglucanase.
DR  InterPro; IPR007118; Expan_Lol_pI.
DR  InterPro; IPR002963; Expansin.
DR  InterPro; IPR007112; Expansin/allergen_DPBB_dom.
DR  InterPro; IPR007117; Expansin_CBD.
DR  InterPro; IPR009009; RlpA-like_DPBB.
DR  Pfam; PF03330; DPBB_1; 1.
DR  Pfam; PF01357; Pollen_allerg_1; 1.
DR  PRINTS; PR01226; EXPANSIN.
DR  PRINTS; PR01225; EXPANSNFAMLY.
DR  SMART; SM00837; DPBB_1; 1.
DR  SUPFAM; SSF49590; SSF49590; 1.
DR  SUPFAM; SSF50685; SSF50685; 1.
DR  PROSITE; PS50843; EXPANSIN_CBD; 1.
DR  PROSITE; PS50842; EXPANSIN_EG45; 1.
PE  3: Inferred from homology;
KW  Cell wall; Cell wall biogenesis/degradation; Complete proteome;
KW  Disulfide bond; Membrane; Reference proteome; Secreted; Signal.
FT  SIGNAL     1    30     {ECO:0000255}.
FT  CHAIN     31   262     Expansin-A7.
FT                         /FTId=PRO_0000008688.
FT  DOMAIN    55   167     Expansin-like EG45. {ECO:0000255|PROSITE-
FT                         ProRule:PRU00079}.
FT  DOMAIN   177   257     Expansin-like CBD. {ECO:0000255|PROSITE-
FT                         ProRule:PRU00078}.
FT  DISULFID  58    86     {ECO:0000255|PROSITE-ProRule:PRU00079}.
FT  DISULFID  89   162     {ECO:0000255|PROSITE-ProRule:PRU00079}.
FT  DISULFID  94   100     {ECO:0000255|PROSITE-ProRule:PRU00079}.
SQ  SEQUENCE 262 AA; 28791 MW; 9F1523339AB55664 CRC64;
    MGPISSSWSF NKFFSIVFVV FAISGEFVAG YYRPGPWRYA HATFYGDETG GETMGGACGY
    GNLFNSGYGL STAALSTTLF NDGYGCGQCF QITCSKSPHC YSGKSTVVTA TNLCPPNWYQ
    DSNAGGWCNP PRTHFDMAKP AFMKLAYWRA GIIPVAYRRV PCQRSGGMRF QFQGNSYWLL
    IFVMNVGGAG DIKSMAVKGS RTNWISMSHN WGASYQAFSS LYGQSLSFRV TSYTTGETIY
    AWNVAPANWS GGKTYKSTAN FR
//
```

A.1.2. GenBank format

The GenBank sequence format (GB) is a text-based format for describing sequences, their associated annotations, and any relevant related information, such as taxonomy, references, or Coding DNA Sequences (CDSs). This format is employed by the NCBI[3] and DDJB[4], and can be easily parsed by BioPerl's Bio::SeqIO module.

Format A.2: Genbak entry for protein *alpha-expansin family protein*[5]

```
LOCUS    NP_172717          262 aa       linear PLN 22-JAN-2014
DEFINITION alpha-expansin family protein [Arabidopsis thaliana].
ACCESSION NP_172717
VERSION  NP_172717.1 GI:15222017
DBSOURCE REFSEQ: accession NM_101127.3
KEYWORDS RefSeq.
SOURCE   Arabidopsis thaliana (thale cress)
  ORGANISM Arabidopsis thaliana
         Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
         Spermatophyta; Magnoliophyta; eudicotyledons; Gunneridae;
         Pentapetalae; rosids; malvids; Brassicales; Brassicaceae;
         Camelineae; Arabidopsis.
REFERENCE 1 (residues 1 to 262)
  AUTHORS Theologis,A., Ecker,J.R., Palm,C.J., Federspiel,N.A., Kaul,S.,
         White,O., Alonso,J., Altafi,H., Araujo,R., Bowman,C.L.,
```

[3] http://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html. [Online: accessed 5-September-2015]

[4] http://www.ddbj.nig.ac.jp/sub/ref10-e.html. [Online: accessed 5-September-2015]

[5] http://www.ncbi.nlm.nih.gov/protein/NP_172717.1. [Online: accessed 5-September-2015]

```
         Brooks,S.Y., Buehler,E., Chan,A., Chao,Q., Chen,H., Cheuk,R.F.,
         Chin,C.W., Chung,M.K., Conn,L., Conway,A.B., Conway,A.R.,
         Creasy,T.H., Dewar,K., Dunn,P., Etgu,P., Feldblyum,T.V., Feng,J.,
         Fong,B., Fujii,C.Y., Gill,J.E., Goldsmith,A.D., Haas,B.,
         Hansen,N.F., Hughes,B., Huizar,L., Hunter,J.L., Jenkins,J.,
         Johnson-Hopson,C., Khan,S., Khaykin,E., Kim,C.J., Koo,H.L.,
         Kremenetskaia,I., Kurtz,D.B., Kwan,A., Lam,B., Langin-Hooper,S.,
         Lee,A., Lee,J.M., Lenz,C.A., Li,J.H., Li,Y., Lin,X., Liu,S.X.,
         Liu,Z.A., Luros,J.S., Maiti,R., Marziali,A., Militscher,J.,
         Miranda,M., Nguyen,M., Nierman,W.C., Osborne,B.I., Pai,G.,
         Peterson,J., Pham,P.K., Rizzo,M., Rooney,T., Rowley,D., Sakano,H.,
         Salzberg,S.L., Schwartz,J.R., Shinn,P., Southwick,A.M., Sun,H.,
         Tallon,L.J., Tambunga,G., Toriumi,M.J., Town,C.D., Utterback,T.,
         Van Aken,S., Vaysberg,M., Vysotskaia,V.S., Walker,M., Wu,D., Yu,G.,
         Fraser,C.M., Venter,J.C. and Davis,R.W.
  TITLE   Sequence and analysis of chromosome 1 of the plant Arabidopsis
         thaliana
  JOURNAL Nature 408 (6814), 816-820 (2000)
   PUBMED 11130712
REFERENCE 2 (residues 1 to 262)
  AUTHORS Swarbreck,D., Lamesch,P., Wilks,C. and Huala,E.
  CONSRTM Arabidopsis TAIR10 Release
  TITLE   Direct Submission
  JOURNAL Submitted (18-FEB-2011) Department of Plant Biology, Carnegie
         Institution, 260 Panama Street, Stanford, CA, USA
COMMENT  REVIEWED REFSEQ: This record has been curated by TAIR. The
         reference sequence is identical to AEE28897.
         Method: conceptual translation.
FEATURES        Location/Qualifiers
    source      1..262
                /organism="Arabidopsis thaliana"
                /db_xref="taxon:3702"
                /chromosome="1"
                /ecotype="Columbia"
    Protein     1..262
                /product="alpha-expansin family protein"
                /calculated_mol_wt=28660
    Region      32..261
                /region_name="PLN00193"
                /note="expansin-A; Provisional"
                /db_xref="CDD:215097"
    Region      72..157
                /region_name="DPBB_1"
                /note="Rare lipoprotein A (RlpA)-like double-psi
                beta-barrel; cl04011"
                /db_xref="CDD:277442"
    Region      168..246
                /region_name="Pollen_allerg_1"
                /note="Pollen allergen; pfam01357"
                /db_xref="CDD:250557"
    CDS         1..262
                /gene="EXPA7"
                /locus_tag="AT1G12560"
                /gene_synonym="ATEXP7; ATEXPA7; ATHEXP ALPHA 1.26; EXP7;
                expansin A7; F5O11.30; F5O11_30"
                /coded_by="NM_101127.3:39..827"
                /inference="Similar to DNA sequence:INSD:BX815910.1"
                /inference="Similar to RNA sequence,
                EST:INSD:T76481.1,INSD:N37411.1"
                /note="expansin A7 (EXPA7); INVOLVED IN: plant-type cell
                wall modification involved in multidimensional cell
                growth, unidimensional cell growth, plant-type cell wall
                loosening; LOCATED IN: endomembrane system, extracellular
                region; EXPRESSED IN: sperm cell, root hair, root;
                CONTAINS InterPro DOMAIN/s: Barwin-related endoglucanase
                (InterPro:IPR009009), Pollen allergen, N-terminal
                (InterPro:IPR014734), Expansin (InterPro:IPR002963), Rare
                lipoprotein A (InterPro:IPR005132), Expansin/Lol pI
                (InterPro:IPR007118), Expansin 45, endoglucanase-like
                (InterPro:IPR007112), Pollen allergen/expansin, C-terminal
                (InterPro:IPR007117); BEST Arabidopsis thaliana protein
                match is: expansin A18 (TAIR:AT1G62980.1); Has 2192 Blast
                hits to 2189 proteins in 166 species: Archae - 0; Bacteria
                - 20; Metazoa - 0; Fungi - 53; Plants - 2083; Viruses - 0;
                Other Eukaryotes - 36 (source: NCBI BLink)."
                /db_xref="GeneID:837813"
                /db_xref="TAIR:AT1G12560"
ORIGIN
      1 mgpissswsf nkffsivfvv faisgefvag yyrpgpwrya hatfygdetg getmggacgy
     61 gnlfnsgygl staalsttlf ndgygcgqcf qitcsksphc ysgkstvvta tnlcppnwyq
    121 dsnaggwcnp prthfdmakp afmklaywra giipvayrrv pcqrsggmrf qfqgnsywll
    181 ifvmnvggag diksmavkgs rtnwismshn wgasyqafss lygqslsfrv tsyttgetiy
    241 awnvapanws ggktykstan fr
```

```
//
```

A.2. Program Files

A.2.1. FASTA format

FASTA (Pearson and Lipman, 1988) is a popular alignment tool which input format has become one of the standards in Bioinformatics to represent nucleotides or AA sequences. The FASTA format (see Format A.3) consists of a header that starts with ">" followed by an unique identifier. An optional description can be added after the identifier separated by a white space. This description is, in general, part of the functional annotation. The lines that come after the header are related to the sequence. It is recommended that all lines of text should be shorter than 80 characters, and no blanks lines are accepted in the middle of the file (NCBI - National Center for Biotechnology Information, 2007). This format can be parsed using BioPerl's Bio::SeqIO module.

Format A.3: FASTA for the human histone H1.1 amino acid sequence[6].

```
>gi|4885373|ref|NP_005316.1| histone H1.1 [Homo sapiens]
MSETVPPAPAASAAPEKPLAGKKAKKPAKAAAASKKKPAGPSVSELIVQAASSSKERGGV
SLAALKKALAAAGYDVEKNNSRIKLGIKSLVSKGTLVQTKGTGASGSFKLNKKASSVETK
PGASKVATKTKATGASKKLKKATGASKKSVKTPKKAKKPAATRKSSKNPKKPKTVKPKKV
AKSPAKAKAVKPKAAKARVTKPKTAKPKKAAPKKK
```

A.2.2. BLAST output format

BLAST (Altschul et al., 1997) can produce different types of output formats, from XML to tabular or Comma-Separated Values (CSV). These formats have essentially the same information, described with different levels of detail. BLAST's default output (see Format A.4) is the most complete of the reports. It can be processed by Bio::SearchIO, and it is the one used in this work.

The default output comprises a list of target sequences, which are similar to the query (*hits*). Each hit has its own accession name or identifier and description. The hits are ranked according to their *bit score* and *e-value*, which are described next. After the *hit* list, the alignments for each hit are displayed. Alignments comprise the following features:

- Query identifier and description
- **Length:** the lenght of the alignment, that is, how long the two sequences have matched. This alignment can include gaps.
- ***Bit score*:** The score is a value that estimates the overall quality of an alignment. The *bit score* is a normalized (i.e., log-scaled) score expressed in bits. It indicates

[6] http://www.ncbi.nlm.nih.gov/protein/4885373. [Online: accessed 5-September-2015]

the magnitude of the search space that should be searched in order to expect to find a score as good as or better than this one by chance. This value indicates the statistical significance of the alignment. The higher the value is, the more similar the sequences are. Values below 50 bits are considered unreliable (Claverie and Notredame, 2006).

- ***E-value*:** The *e-value*, or expectation value, provides the most important statistical information. It indicates the number of hits that can occur by chance when searching in a database of a certain size. Its value decreases exponentially as the *bit score* increases. The lower the *e-value*, that is, the closer it gets to zero, the more "significant" the match is. However, it is important to take into account that short sequences have high *e-values* because they are more likely to occur in the database purely by chance (NCBI - National Center for Biotechnology Information, 2015). Thus, in order to annotate a small sequence, more information is needed besides the BLAST report.
- **Percent of identity (Identities):** the number of identical residues divided by the number of matched residues. Gaps are ignored.
- **Positives:** the number of identical or similar residues (represented by "+") divided by the number of matched residues. Gaps are ignored.
- **Frame:** strand and reading frame (see Section 2.2) of the query sequence used in the alignment. This is just relevant when the target is an AA sequence.
- **Alignment:** the top sequence is the query (Query) and the bottom sequence is the *hit* (Sbjct). The sequence in between is the line that contains the identical and similar residues ("+"). The numbers on the right side of "Query" and "Sbjct" indicate the coordinates of the matching.

Format A.4: BLASTX format example for a query with 2 hits.

```
BLASTX 2.2.26 [Sep-21-2011]


Reference: Altschul, Stephen F., Thomas L. Madden, Alejandro A. Schaffer,
Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman (1997),
"Gapped BLAST and PSI-BLAST: a new generation of protein database search
programs", Nucleic Acids Res. 25:3389-3402.

Query= sequence1
       (3601 letters)

Database: uniprot_sprot
        518,337 sequences; 179,807,836 total letters

Searching.................................................done


                                                  Score  E
Sequences producing significant alignments:      (bits) Value

sp|Q3U034|METL4_MOUSE Methyltransferase-like protein 4 OS=Mus mu... 559 0.0
sp|Q8LFA9|METL2_ARATH Methyltransferase-like protein 2 OS=Arabid... 108 2e-23
```

```
>sp|Q3U034|METL4_MOUSE Methyltransferase-like protein 4 OS=Mus
         musculus GN=Mettl4 PE=2 SV=1
        Length = 471

 Score = 559 bits (1440), Expect = 0.0
 Identities = 283/394 (71%), Positives = 318/394 (80%)
 Frame = +2

Query: 800 MSVVHQLSAGWLLDHLSFINKINYQLHQHHEPCCRKKEFTTSVHFESLQMDSVSSSGVCA 979
          MSVVH L GWLLDHLSFINK+NYQL QH E C K T+SV+ +SLQ+D S G A
Sbjct: 1 MSVVHHLPPGWLLDHLSFINKVNYQLCQHQESFCSKNNPTSSVYMDSLQLDPGSPFGAPA 60

Query: 980 AFIASDSSTKPENDDGGNYEMFTRKFVFRPELFDVTKPYITPAVHKECQQSNEKEDLMNG 1159
            A D +T NDD G+ E+ T K+VFR ELF+VTKPYI PAVHKE QQSN+ E+L+
Sbjct: 61 MCFAPDFTTVSGNDDEGSCEVITEKYVFRSELFNVTKPYIVPAVHKERQQSNKNENLVTD 120

Query: 1160 VXXXXXXXXXXXXXXXCVVFNQGELDAMEYHTKIRELILDGSLQLIQEGLKSGFLYPLFE 1339
                      C+ FNQGELDAMEYHTKIRELILDGS +LIQEGL+SGFLYPL E
Sbjct: 121 YKQEVSVSVGKKRKR-CIAFNQGELDAMEYHTKIRELILDGSSKLIQEGLRSGFLYPLVE 179

Query: 1340 KQDKGSKPITLPLDACSLSELCEMAKHLPSLNEMEHQTLQLVEEDTSVTEQDLFLRVVEN 1519
          KQD S ITLPLDAC+LSELCEMAKHLPSLNEME QTLQL+ +D SV E DL +++EN
Sbjct: 180 KQDGSSGCITLPLDACNLSELCEMAKHLPSLNEMELQTLQLMGDDVSVIELDLSSQIIEN 239

Query: 1520 NSSFTKVITLMGQKYLLPPKSSFLLSDISCMQPLLNYRKTFDVIVIDPPWQNKSVKRSNR 1699
          NSSF+K+ITLMGQKYLLPP+SSFLLSDISCMQPLLN KTFD IVIDPPW+NKSVKRSNR
Sbjct: 240 NSSFSKMITLMGQKYLLPPQSSFLLSDISCMQPLLNCGKTFDAIVIDPPWENKSVKRSNR 299

Query: 1700 YSYLSPLQIQQIPIPKLAAPNCLLVTWVTNRQKHLRFIKEELYPSWSVEVVAEWHWVKIT 1879
          YS LSP QI+++PIPKLAA +CL+VTWVTNRQKHL F+KEELYPSWSVEVVAEW+WVKIT
Sbjct: 300 YSSLSPQQIKRMPIPKLAAADCLIVTWVTNRQKHLCFVKEELYPSWSVEVVAEWYWVKIT 359

Query: 1880 NSGEFVFPLDSPHKKPYEGLILGRVQEKTALPLR 1981
          NSGEFVFPLDSPHKKPYE L+LGRV+EKT L LR
Sbjct: 360 NSGEFVFPLDSPHKKPYECLVLGRVKEKTPLALR 393


>sp|Q8LFA9|METL2_ARATH Methyltransferase-like protein 2 OS=Arabidopsis
         thaliana GN=At1g19340 PE=2 SV=2
        Length = 414

 Score = 108 bits (269), Expect = 2e-23
 Identities = 58/166 (34%), Positives = 93/166 (56%), Gaps = 6/166 (3%)
 Frame = +2

Query: 1469 EDTSVTEQ---DLFLRVVENNSSFTKVITLMGQKYLLPPKSSFLLSDISCMQPLLNYRKT 1639
          E S EQ   +F +V N        ++Y++P S F +SD+ ++ L+ +
Sbjct: 165 EGESCNEQRVFQVFNNLVVNEIGEEVEAEFSNRRYIMPRNSCFYMSDLHHIRNLVPAKSE 224

Query: 1640 --FDVIVIDPPWQNKSVKRSNRYSYLSPLQIQQIPIPKLA-APNCLLVTWVTNRQKHLRF 1810
            +++IVIDPPW+N S + ++Y L +PI +LA A L+ WVTNR+K L F
Sbjct: 225 EGYNLIVIDPPWENASAHQKSKYPTLPNQYFLSLPIKQLAHAEGALVALWVTNREKLLSF 284

Query: 1811 IKEELYPSWSVEVVAEWHWVKITNSGEFVFPLDSPHKKPYEGLILG 1948
          +++EL+P+W ++ VA +W+K+ G + LD H KPYE L+LG
Sbjct: 285 VEKELFPAWGIKYVATMYWLKVKPDGTLICDLDLVHHKPYEYLLLG 330
```

A.2.3. InterProScan output format

InterProScan (Jones et al., 2014) output can be delivered as a XML, GFF3 (see Appendix A.3.1), HTML, SVG, or TSV file. All these formats can be easily processed using Perl or specific modules.

InterProScan's TSV output (see Formart A.5) presents 15 columns (being the last 4 optional) (InterPro, 2013):

1. **Protein Accession:** unique identifier of the protein sequence.
2. **Sequence MD5 digest:**
3. **Sequence Length:** length of the matching sequence.

4. **Analysis**: source of the signature (family or domain), e.g., Pfam (Finn et al., 2014), SMART (Letunic et al., 2012b), or PANTHER (Mi et al., 2013).
5. **Signature Accession:** unique identifier of the signature.
6. **Signature Description:** biological description of signature.
7. **Start location:** position of the first residue of the query in the alignment.
8. **Stop location:** position of the last residue of the query in the alignment.
9. **Score:** the e-value of the match reported by member database method. It follows the same idea of BLAST's *e-value* (see Appendix A.2.2).
10. **status:** true (T) for true positives, otherwise unknwon (?).
11. **Date:** date of the execution
12. **InterPro annotation accession:** unique identifier of the InterPro entry that is related to the signature. Optional.
13. **InterPro annotation description:** biological description of the InterPro entry that is related to the signature. Optional.
14. **GO annotations:** GO accession number related to the matched signature. Identifiers are separated by the symbol "|". Optional.
15. **Pathways annotations:** biological pathways related to the signature. Optional.

A.2.4. CD-Search output format

CD-Search (Marchler-Bauer and Bryant, 2004) generates a very visual output (Figure A.1) that depicts specific and non-specific hits (see Section 2.5.1.3), superfamilies, and important sites. It also shows more detail information of the domains matched (Figure A.2), including a colored version of the alingment.

A.2.5. PfamScan output format

PfamScan (Finn et al., 2014) generates a tab-delimited output (see Format A.6) similar to the InterProScan output (see Appendix A.2.3). It consist of 15 columns that contain information such as the sequence id; alignment start and end; Pfam (Finn et al., 2014) domain accession and name; *bit score* and *e-value*.

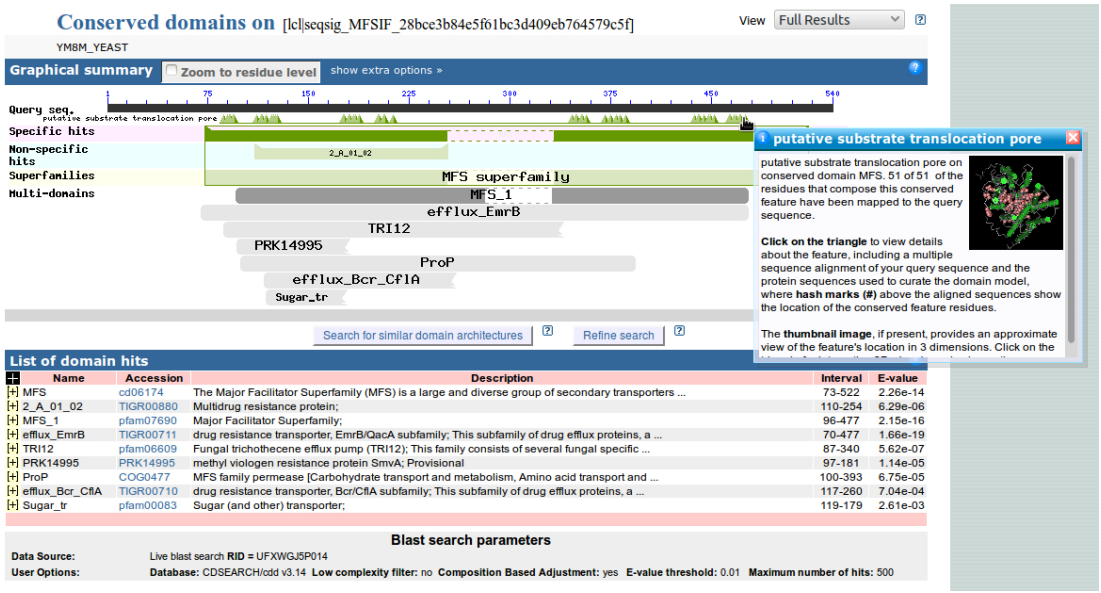
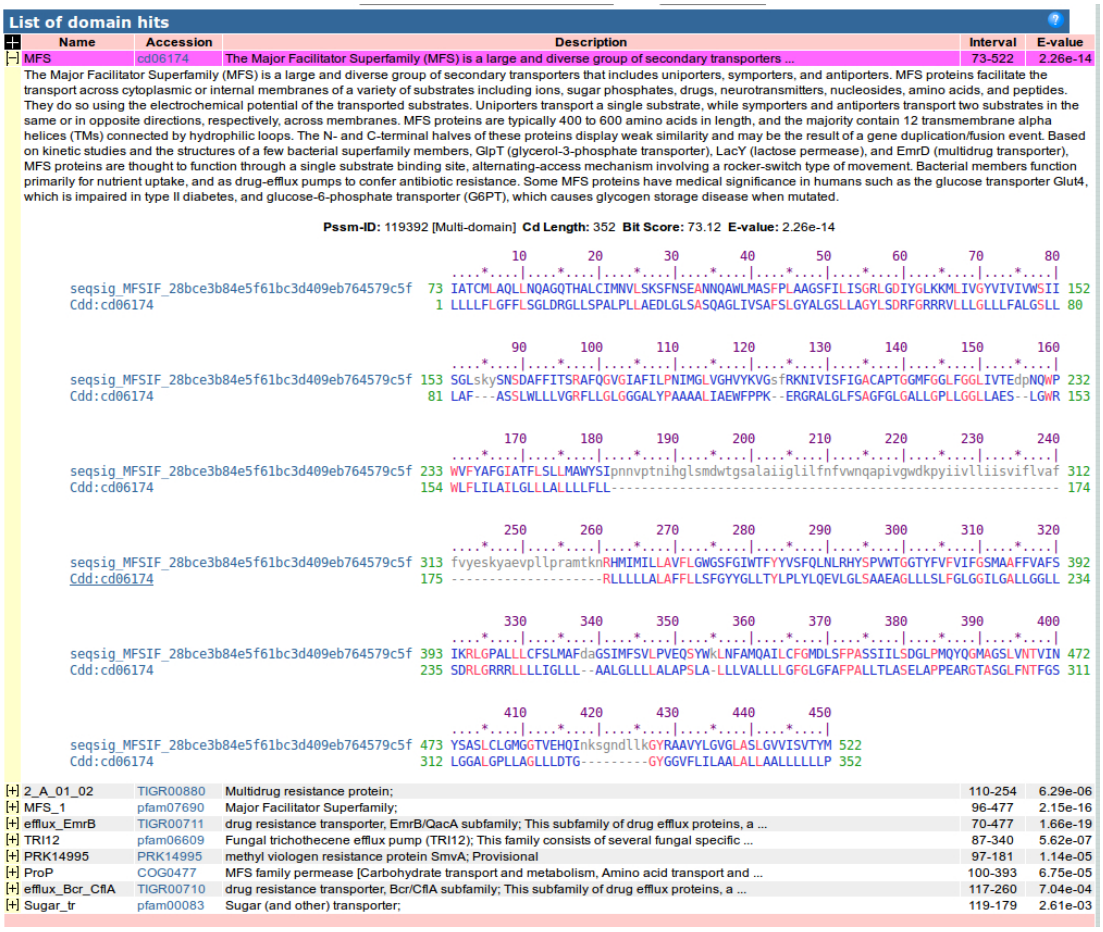


Figure A.1: Full result mode of CD-Search with important site pop-up information.



List of domain hits

Name	Accession	Description	Interval	E-value
MFS	cd06174	The Major Facilitator Superfamily (MFS) is a large and diverse group of secondary transporters ...	73-522	2.26e-14

The Major Facilitator Superfamily (MFS) is a large and diverse group of secondary transporters that includes uniporters, symporters, and antiporters. MFS proteins facilitate the transport across cytoplasmic or internal membranes of a variety of substrates including ions, sugar phosphates, drugs, neurotransmitters, nucleosides, amino acids, and peptides. They do so using the electrochemical potential of the transported substrates. Uniporters transport a single substrate, while symporters and antiporters transport two substrates in the same or in opposite directions, respectively, across membranes. MFS proteins are typically 400 to 600 amino acids in length, and the majority contain 12 transmembrane alpha helices (TMs) connected by hydrophilic loops. The N- and C-terminal halves of these proteins display weak similarity and may be the result of a gene duplication/fusion event. Based on kinetic studies and the structures of a few bacterial superfamily members, GlpT (glycerol-3-phosphate transporter), LacY (lactose permease), and EmrD (multidrug transporter), MFS proteins are thought to function through a single substrate binding site, alternating-access mechanism involving a rocker-switch type of movement. Bacterial members function primarily for nutrient uptake, and as drug-efflux pumps to confer antibiotic resistance. Some MFS proteins have medical significance in humans such as the glucose transporter Glut4, which is impaired in type II diabetes, and glucose-6-phosphate transporter (G6PT), which causes glycogen storage disease when mutated.

Pssm-ID: 119392 [Multi-domain] **Cd Length:** 352 **Bit Score:** 73.12 **E-value:** 2.26e-14

```
                                                  10        20        30        40        50        60        70        80
                                                  ....*....|....*....|....*....|....*....|....*....|....*....|....*....|....*....|
seqsig_MFSIF_28bce3b84e5f61bc3d409eb764579c5f  73 IATCMLAQLLNQAGQTHALCIMNVLSKSFNSEANNQAWLMASFPLAAGSFILISGRLGDIYGLKKMLIVGYVIVIVWSII 152
Cdd:cd06174                                     1 LLLLFLGFFLSGLDRGLLSPALPLLAEDLGLSASQAGLIVSAFSLGYALGSLLAGYLSDRFGRRRVLLLGLLLFALGSLL 80

                                                         90       100       110       120       130       140       150       160
                                                  ....*....|....*....|....*....|....*....|....*....|....*....|....*....|....*....|
seqsig_MFSIF_28bce3b84e5f61bc3d409eb764579c5f 153 SGLskySNSDAFFITSRAFQGVGIAFILPNIMGLVGHVYKVGsfRKNIVISFIGACAPTGGMFGGLFGGLIVTEdpNQWP 232
Cdd:cd06174                                    81 LAF---ASSLWLLLVGRFLLGLGGGALYPAAAALIAEWFPPK--ERGRALGLFSAGFGLGALLGPLLGGLLAES--LGWR 153

                                                        170       180       190       200       210       220       230       240
                                                  ....*....|....*....|....*....|....*....|....*....|....*....|....*....|....*....|
seqsig_MFSIF_28bce3b84e5f61bc3d409eb764579c5f 233 WVFYAFGIATFLSLLMAWYSIpnnvptnihglsmdwtgsalaiiglilfnfvwnqapivgwdkpyiivlliisviflvaf 312
Cdd:cd06174                                   154 WLFLILAILGLLLALLLLFLL---------------------------------------------------------- 174

                                                        250       260       270       280       290       300       310       320
                                                  ....*....|....*....|....*....|....*....|....*....|....*....|....*....|....*....|
seqsig_MFSIF_28bce3b84e5f61bc3d409eb764579c5f 313 fvyeskyaevpllpramtknRHMIMILLAVFLGWGSFGIWTFYYVSFQLNLRHYSPVWTGGTYFVFVIFGSMAAFFVAFS 392
Cdd:cd06174                                   175 --------------------RLLLLLALAFLLSFGYYGLLTYLPLYLQEVLGLSAAEAGLLLSLFGLGGILGALLGGLL 234

                                                        330       340       350       360       370       380       390       400
                                                  ....*....|....*....|....*....|....*....|....*....|....*....|....*....|....*....|
seqsig_MFSIF_28bce3b84e5f61bc3d409eb764579c5f 393 IKRLGPALLLCFSLMAFdaGSIMFSVLPVEQSYWkLNFAMQAILCFGMDLSFPASSIILSDGLPMQYQGMAGSLVNTVIN 472
Cdd:cd06174                                   235 SDRLGRRRLLLLIGLLL--AALGLLLLALAPSLA-LLLVALLLLGFGLGFAFPALLTLASELAPPEARGTASGLFNTFGS 311

                                                        410       420       430       440       450
                                                  ....*....|....*....|....*....|....*....|....*....|
seqsig_MFSIF_28bce3b84e5f61bc3d409eb764579c5f 473 YSASLCLGMGGTVEHQInksgndllkGYRAAVYLGVGLASLGVVISVTYM 522
Cdd:cd06174                                   312 LGGALGPLLAGLLLDTG---------GYGGVFLILAALLAALLLLLLP 352
```

2_A_01_02	TIGR00880	Multidrug resistance protein;	110-254	6.29e-06
MFS_1	pfam07690	Major Facilitator Superfamily;	96-477	2.15e-16
efflux_EmrB	TIGR00711	drug resistance transporter, EmrB/QacA subfamily; This subfamily of drug efflux proteins, a ...	70-477	1.66e-19
TRI12	pfam06609	Fungal trichothecene efflux pump (TRI12); This family consists of several fungal specific ...	87-340	5.62e-07
PRK14995	PRK14995	methyl viologen resistance protein SmvA; Provisional	97-181	1.14e-05
ProP	COG0477	MFS family permease [Carbohydrate transport and metabolism, Amino acid transport and ...	100-393	6.75e-05
efflux_Bcr_CflA	TIGR00710	drug resistance transporter, Bcr/CflA subfamily; This subfamily of drug efflux proteins, a ...	117-260	7.04e-04
Sugar_tr	pfam00083	Sugar (and other) transporter;	119-179	2.61e-03

Figure A.2: CD-Search detailed information of MFS domain (first *hit* on Figure A.1) and its alignment with the query sequence.

Format A.5: InterProScan format example.

```
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 SMART  SM00211   Thyroglobulin type I repeats.  248 296 2.4E-15 T 30-07-2014 IPR000716 Thyroglobulin type-1
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 SMART  SM00211   Thyroglobulin type I repeats.  115 162 1.1E-11 T 30-07-2014 IPR000716 Thyroglobulin type-1
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 Pfam  PF10591   Secreted protein acidic and  315 424 1.5E-12 T 30-07-2014 IPR019577 SPARC/Testican, calcium  GO:0005509|
                 rich in cysteine Ca binding                  -binding domain    GO:0005578|
                 region                              GO:0007165
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 PANTHER PTHR12352:SF10            1 431 0.0  T 30-07-2014
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 Pfam  PF07648   Kazal-type serine protease   47 87 1.5E-6 T 30-07-2014 IPR002350 Kazal domain     GO:0005515
                 inhibitor domain
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 PANTHER PTHR12352             1 431 0.0  T 30-07-2014
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 SMART  SM00280   Kazal type serine protease   42 87 1.5E-7 T 30-07-2014 IPR002350 Kazal domain     GO:0005515
                 inhibitors
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 Pfam  PF00086   Thyroglobulin type-1 repeat  227 292 4.8E-14 T 30-07-2014 IPR000716 Thyroglobulin type-1
query1 e7087b8520ef6784ae5e3b1907dd2ff1 434 Pfam  PF00086   Thyroglobulin type-1 repeat  95 158 1.2E-15 T 30-07-2014 IPR000716 Thyroglobulin type-1
```

Format A.6: PfamScan format example.

```
# <seq id> <alignment start> <alignment end> <envelope start> <envelope end> <hmm acc> <hmm name> <type> <hmm start> <hmm end> <hmm length> <bit score> <E-value> <significance> <clan>

ENV10_YEAST  1   171    1   171 PF05620.7 DUF788      Family  1  167  167   145.1 1.7e-42 1 No_clan
ENV10_YEAST 102  136   95   156 PF05101.9 VirB3       Family 27   61   84     9.5      1  0 No_clan
YD306_YEAST 115  172  114   175 PF12937.3 F-box-like  Domain  4   44   47    14.1    0.03 0 CL0271
YD306_YEAST 322  401  310   415 PF13306.2 LRR_5       Family 23  107  128    10.7    0.35 0 CL0022
SRF1_YEAST 271   419  267   421 PF01284.19 MARVEL     Domain  8  142  144    17.5  0.0028 0 CL0396
YM8M_YEAST  77   477   76   479 PF07690.12 MFS_1      Family  2  350  352   111.2 4.4e-32 1 CL0015
YM8M_YEAST 502   534  492   539 PF16082.1 Phage_holin_2_4 Family 35 65 81    7.5     3.5  0 CL0563
YCT1_YEAST  67   431   55   432 PF07690.12 MFS_1      Family  5  351  352    66.3 2.1e-18 1 CL0015
YCT1_YEAST 303   400  263   407 PF11833.4 DUF3353     Family 89  191  198    11.8    0.12 0 No_clan
```

Format A.7: GFF used in MASSA.

```
NAAA_HUMAN SOURCE Contig 36 357 . . . 454|MIRA "NAAA_HUMAN"
NAAA_HUMAN BLASTX|UniProt  Annotation 36 357 0.0 + 0 Description "N-acylethanolamine-hydrolyzing acid amidase" ;FracId 0.807453416149068 ;Bits 554.0 ; GO "GO:0005737 GO:0005764
     GO:0006629 GO:0008134 GO:0016787 GO:0016810" ; Source_score 1.0; Domain_score 1.0 ;Specificity_score 0.0 ;Conserved_score 1.0 ;Orthology_score 1.0 ; Target
     Protein:sp|Q5KTC7|NAAA_RAT 41 362
NAAA_HUMAN InterProScan|Pfam similarity 33 121 4.5E-21 + 0 Description "beta subunit of N-acylethanolamine-hydrolyzing acid amidase" ;Specificity 0 ; Target Domain:PF15508
NAAA_HUMAN InterProScan|Pfam similarity 126 292 8.7E-7 + 0 Description "Linear amide C-N hydrolases, choloylglycine hydrolase family" ;Specificity 0 ;GO "GO:0016787" ; Target
     Domain:PF02275
NAAA_HUMAN InterProScan|InterPro similarity 126 292 8.7E-7 + 0 Description "Choloylglycine hydrolase/NAAA C-terminal" ;Specificity 0 ; Target Domain:IPR029132
NAAA_HUMAN BLASTX|UniProt  Annotation 41 362 . + 0 Match sp|Q5KTC7|NAAA_RAT
NAAA_HUMAN BLASTX|UniProt  HSP 126 126 . + 0 Description "Nucleophile" ;Site_Type "ACT_SITE" ;Name "ACT_SITE" Target Site1:ACT_SITE_Nucleophile
NAAA_HUMAN RPS-BLAST|CDD  similarity 35 121 5.0E-26 + 0 Description "NAAA-beta" ;FracId 0.344827586206897 ;Bits 101.0 ;Specificity 0 ; Target Domain:PF15508 5 93
```

A.2.6. Clustal format

ClustalW (Larkin et al., 2007) creates a very straightforward output. It presents sequences aligned in blocks (see Format A.8), organized in three columns. The first one contains the sequences identifiers; the second one the sequences aligned; and the third one the start position of the alignment.

As the majority of multi-alignment outputs, gaps (e.g., not aligned residues) are represented as "-", and consensus symbols are used to indicate levels of conservation. An "*" (asterisk) indicates fully conserved sites; a ":" (colon) indicates strong similar properties, and a "." (period) weak similarity.

Format A.8: CLUSTAL output format.

```
ENV10_YEAST  ------------------------------------------------------------
YM8M_YEAST   ----MFSIFKKKTSVQGTDSEIDEKITVKAKDKVVVSTEDEEVTTIVSSTKSTQVTNDSP 56
SRF1_YEAST   MGDSNSSQEAYSDTTSTNASRIADQNQLNLNVDLEKNQTVRKSGSLEALQNAKIHVPKHS 60
YCT1_YEAST   -------------MSKVDVKIGADSISSSDEILVPSRLADVTLAFMEENDAAVPEITPE 46
YD306_YEAST  --------MANKSRPKKIKAPYRKYVAGEGFSSTRNDNKAKEFTITIPEDAELIETPQGS 52


ENV10_YEAST  ------------------------MAGKAGRKQASSN------AKIIQGLYKQVS----- 25
YM8M_YEAST   WQDPTYFSSFGKELMFIATCMLAQLLNQAGQTHALCI------MNVLSKSFNSEANNQAW 110
SRF1_YEAST   DGSPLDYPKLNTYTFVPTTVPPYVLEAQFDKLRLQDK------GTVDGNVTDDKNLPKEF 114
YCT1_YEAST   QEKKLKRKLFLTIFTFVSAINLLLYMDKATLSYDSILGFFE-DTGLTQNTYNTVN-TLFY 104
YD306_YEAST  YYYDETNDTIVKLTRLSNEKKDKKGRKQSPSSSSTSSSKGEKNGKVIESEEARMHSVSVK 112
                              :           :

ENV10_YEAST  LFLGMAIVR-LFISRKVTIG------QWIKLVALNVP-MFVALYIIVLSGKPKYDGN--- 74
YM8M_YEAST   LMASFPLAAGSFILISGRLG------DIYGLKKMLIVGYVIVIVWSIISGLSKYSNSDAF 164
SRF1_YEAST   KWGQFASTIGCHSAYTRDQNY-NPSHKSYDGYSLSSSTSSKNAALREILGDMCSEWGGEE 173
YCT1_YEAST   VGFAIGQFPGQYLAQKLPLG--KFLGGLLATWTILIFLSCTAYNFSGVVALRFFLGLTES 162
YD306_YEAST  MVLPWEIQHRIIHYLDIPEKEEKLNKTANGKKTTTGINMNYLLVCRNWYAMCLPKLYYAP 172
                                                .

ENV10_YEAST  -------RVVKQGIDLNDNTNLISYFFDL--------------------IYLSLFG---- 103
YM8M_YEAST   FITSRAFQGVGIAFILPNIMGLVGHVYKVGSFRKNIVISFIGACAPTGGMFGGLFGGLIV 224
SRF1_YEAST   RLEGVLHSEIGANLEFNTTEERKEWLQYIEKVKDFYYGDNKKNPESPESVHNKVYKSDWV 233
YCT1_YEAST   VVIPILITTMGMFFDASERAAAQPFFFAACMGSPIPTGFIAYGVLHITNPSISLWKIFTI 222
YD306_YEAST  ALTSKNFNGFVDTIIINKKKNLGHYVFELNLSTILQSGRNSFVSKLLRRCCSNLTKFIAP 232
                    .  :       .                 :

ENV10_YEAST  ------------------------------------------------NIGIIAFRTFKFW 116
YM8M_YEAST   TEDPNQWPWVFYAFGIATFLSLLMAWYSIPNNVPTNIHGLSMDWTGSALAIIGLILFNFV 284
SRF1_YEAST   N---------------------------------------ELNKEREKWRRLKQRKLQQW 254
YCT1_YEAST   IIGGLTFIMTVVVILWFPNNPADVKFFSIQERVWIIRRVQASTGSSIEQKVFKKSQFREA 282
YD306_YEAST  QTSFG-------------------------------YAPLISLKSCHDLKFLDLGLVSET 261
                                               :   .

ENV10_YEAST  W-----------------------------------------------CLLLCPIYA 126
YM8M_YEAST   WNQAPIVGWDKPYIIVLLIISVIFLVAFFVYESKYAEVPLLPRAMTKNRHMIMILLAVFL 344
SRF1_YEAST   RPPLTSLLLDN---------------------------------------QYLILGLRIFT 276
YCT1_YEAST   MKDYITWLFGLFFLLQQLANNLPYQQNLLFEG--------------MGGVDALGSTLVSV 328
YD306_YEAST  VKLKELFS-----------------------------------------AIKNFTKLTHL 280


ENV10_YEAST  GYKLYGLKNMFMPGAQQTQADNR------------------------------------ 149
YM8M_YEAST   GWGSFGIWTFYYVSFQLNLRHYSPVWTGGTYFVFVIFGSMAAFFVAFSIKRLGPALLLCF 404
SRF1_YEAST   GILSCISLALAIKIFQNSRSNNTISESKIGQQPSTIMAICVNAVAIAYIIYIAHDEFAGK 336
YCT1_YEAST   AGAGFAVVCAFIATLMLAKWKNISALTAIFWTLPALVGSIAAAALPWDNKIGILANICMA 388
YD306_YEAST  SFPRSSIDCQGFQDIQWPQNLRYLKLSGGITNEFVIDTKWPTTITTLEFSYCPQITELSI 340
             .

ENV10_YEAST  -------------------------------------------SKNANEGQSKSKRQMK- 165
YM8M_YEAST   SLMAFDAGSIMFSVLPVEQSYWKLNFAMQAILCFGMDLSFPASSIILSDGLPMQYQGMAG 464
SRF1_YEAST   PVGLRNPLSKLKLILLDLLFIIFSSANLALAFNTRFDKEWVCTSIRRSNGSTYGYPKIPR 396
YCT1_YEAST   GQIFGIPFIIALSWASSSASGYTKKLTRSSVSLFAMGIANIISPQIWREKDSPRFLPAWI 448
YD306_YEAST  YSLLSQIGDNLKHLFFHYPMPSLAENSLDHVFTYCANLISLQLMVDYCSKWCFSEFMLSK 400
                                               .

ENV10_YEAST  ----------------------RERRGE---TDSKIKYKYR---------------------- 181
YM8M_YEAST   SLVNTVINYSASLCLGMGGTVEHQINK---SGNDLLKGYRAAVYLGVGLASLGVVISVTY 521
SRF1_YEAST   ----------------------ICRKQE---ALSAFLFVALFMWVITFSISIVRVVEKVSS 432
YCT1_YEAST   VQIVLSFSLAPAILLLIHFILKRRNNQRLKNYDENLQNYLDRIQLIESENPSSIEEGKVV 508
```

```
YD306_YEAST  LVEYDRPLKTLYLECSGSLGLASKIHP-DDLTIAILESRLPCLKNICVSPKLGWNMKSDE 459
                              :

ENV10_YEAST  ------------------------
YM8M_YEAST   MLENLWNRHRKSEDRSLEA---- 540
SRF1_YEAST   ITNRN------------------- 437
YCT1_YEAST   THENNLAVFDLTDLENETFIYPL 531
YD306_YEAST  VADLVVSLEDQDGSLYLNY---- 478
```

A.2.7. Simple multi-alignment format

Almost all multi-aligners can generate this format. Simple multi-alignment format is very similar to ClustalW's (see Appendix A.2.6) but it lacks the sequence start coordinate (see Format A.9). T-Coffee (Notredame et al., 2000) and MUSCLE (Edgar, 2004) are able to produce this output, although it is not MUSCLE's default output (see Appendix A.2.8).

Format A.9: Simple multi-alignment created by T-Coffee.

```
ENV10_YEAST MAGKAGRKQAS---------SN-----------------------------------------------
YD306_YEAST MANKSRPKKIKAPYRKYVAGEGFSSTRNDN---KAKEFTITIPEDAELIETPQGSYYYD----------
SRF1_YEAST MGDSN---------SSQEAYSDTTSTNASRI---ADQNQLNLNVDLEKNQTVRKSGSLEALQNAKIHVP
YM8M_YEAST MFSIF---------KKKTSVQGTDSEIDEKITVKAKDKVVVSTEDEEVTTIVSSTKSTQ----------
YCT1_YEAST MSKVD---------VKIGADS-----------ISSSDEILVPSRLADVTLAFMEENDAA----------
          *             .

ENV10_YEAST -----A-----------------------------------------------------------------
YD306_YEAST -ETNDTIVKLTRLSNEKK----------------------DKKGRKQSP---------------------
SRF1_YEAST KHSDGSPLD------YPKLN------TYTFVPTTVPPYVL-----------------------------
YM8M_YEAST -VTNDSPWQ------DPTYFSS-FGKELMFIATCMLAQLLNQAGQTHALCIMNVLSKSFNSEANNQAWL
YCT1_YEAST -VPEITPEQ------EKKLKRKLFLTIFTFVSAINLLLYMDKATLSYDSILGFFEDTGLTQNTYNTVNT
              :

ENV10_YEAST ----------------------KIIQGL-------YKQVS------------------------------
YD306_YEAST --------SSSSTSSSKGEKNGKVIESEE------ARMHSVSVKMVLPWEIQHRIIHYLDIPEKEEKLN
SRF1_YEAST EAQFDKLRLQDK-----GTVDGN-----VTDDKNLPKEFKWG----QFASTIGCHSAYTRD----QNYN
YM8M_YEAST MASFPL-AAGSF-----ILISGR-----LGDIYGLKKMLIVGYVIVIVWSIISGLSKYSNS----DAFF
YCT1_YEAST LFYVGF-AIGQFPGQ-YLAQKLP-----LGKFLG---------GLLATWTILIFLSCTAYN----FSGV


ENV10_YEAST ----------------------LFLGMAIV----------------------RLF---------------
YD306_YEAST KTANGKKTTTGINMNYLLVCRNWYAMCLP----------------------KLYY-APALTSKNFNGFV
SRF1_YEAST P---SHKSYDGYSLSSS-------------------------TSSKNAALREILGDMCSEWGGE--ERL
YM8M_YEAST I---TSRAFQGVGIAFIL----------PNIMGLVGHVYKVGSFRKNIV-ISFIG-ACAPTGGMFGGLF
YCT1_YEAST V---ALRFFLGLTESVVI----------PILITTMGMFFDA-SERAAAQ-PFFFA-ACMGSPIPTGFIA
                                                :

ENV10_YEAST ----ISRKVTIGQWIKLVALNVP------------------------MFVALYIIVLSGKPKYDGNRV--
YD306_YEAST DTIIINKKKNLGHYVF--ELNLSTILQSGRNSFVSKLLRRCCSNLTKFIAP-------QTSFGYAPLIS
SRF1_YEAST EGVLHS-------------------------------------------------------------
YM8M_YEAST GGLIVTEDPN-----------------------------------------------------------
YCT1_YEAST YGVLHITNPS-----------------------------------------------------------


ENV10_YEAST --------------VKQGIDLNDNTNLISY-----FFDLIYLSL-FG-NIGIIAFRTFK----------
YD306_YEAST LKSCHDLKFLDLGLVSETVKLKELFSAIKN-----FTKLTHLSFPRS-SIDCQGFQDIQWPQNLR----
SRF1_YEAST ------------------------------------------EI-GANLEFNTTEERKEWLQYIEK
YM8M_YEAST ------------------QWPWVFYA--FGIATFLSLLMAWYSIPNNVPTNIHGLSMDW---T------
YCT1_YEAST ------------------ISLWKIFTIIIGGLTFIMTVVVILWFPNN-PADVKFFSIQERVWII-----
                                                .

ENV10_YEAST ----------------------------------------FWWCLLLCPIYAGYKLYG----------
YD306_YEAST ---------YLKLSGGITNEFVIDTKWP------TTITTLEFSYCPQ----ITELSIYSLLSQ------
SRF1_YEAST VKDFYYGDNKKNPESPESVHNKVYKSDWVNELNKEREKWR----R----------LKQRKLQQWRPPLT
YM8M_YEAST --------GSALAIIGLILFNFVWNQAPIVGWDKPYIIVL----LII--------SVIFLVAF------
YCT1_YEAST --------RRVQASTGSSIEQKVFKKSQFREAMKDYITWL----F----------GLFFLLQQ------


ENV10_YEAST ------------------------------------------------------------------------
YD306_YEAST ------------------------------------------------------------------IG
SRF1_YEAST SLLLDNQYLILGLRIFTGILSCISLALAIKIFQNSRSNNTISESKIGQQPSTIMAICVNAVAIAYIIYI
YM8M_YEAST ------------------------------------------------------------------FV
YCT1_YEAST ------------------------------------------------------------------LA
```

```
ENV10_YEAST --L----KNMFMPG------------------------------------------------------
YD306_YEAST DNL----KHLFFHY------------------------------------------------------
SRF1_YEAST AHDEFAGKPVGLRN-------------------------------------------------------
YM8M_YEAST YESKYAEVPLLPRAMTKNRHMIMILLAVFLGWGSFGIWTFYYVSFQLNLRHYSPVWTG--------GTY
YCT1_YEAST NNLPYQQN-LLFEGMG-----------------------------------GVDALGSTLVSVAGAG

                :
ENV10_YEAST ----------------------AQ-QTQADN---------------------------------------
YD306_YEAST ----------------------PM-PSLAENSLDHVFT-YCANLISLQLMVDYC----------------
SRF1_YEAST ----------------------PL-SKLKLILLDLLFIIFSSANLALAFNTRFD----------------
YM8M_YEAST FVFVIFGSMAAFFVAF---SIKRLGPALLLCFSLMAFDAGSIMFSVLPVEQSYWKLNFAMQAILCFGM-
YCT1_YEAST F-----AVVCAFIATLMLAKWKNI-SALTAIFWTLPALVGSIAAAALPWDNKIGILANICMAGQIFGIP


ENV10_YEAST -------------RSKNANEGQ----------------------SK------------------------
YD306_YEAST ----SKWCFSEFMLSKLVEYDR----------------------PLKTLYLECSGSLGLASKIHPDDLTI
SRF1_YEAST ----KEWVCTSIRRSNGSTYGY-------PK-----------------------------------IP
YM8M_YEAST ---DLSFPASSIILSDGLPMQYQGMAGSLVNTVINYSASL-CLG-------------MGGTV-----EH
YCT1_YEAST FIIALSWASSS--------ASGY-------TKKLTRSSVSLFAMG-------------IANII-----SP


ENV10_YEAST ------------SKRQM---------K---RERR-------------GETDSKIKY-------------
YD306_YEAST AILESRL---PCLKNICVSPKLGWNMK---SDEVAD-LVVSL-----EDQDGSLYL-------------
SRF1_YEAST RICRKQEALSAFLFVALFMWVITFSIS---IVRVVE-KVSSITNRN-----------------------
YM8M_YEAST QINKSG--N-DLLKGYRAAVYLGVGLASLGVVISVTYMLENLWNRHRKSEDRSLEA-------------
YCT1_YEAST QIWREKDSP-RFLPAWIVQIVLSFSLAP-AILLLIHFILKRRNNQRLKNYDENLQNYLDRIQLIESENP


ENV10_YEAST -----KYR-------------------------
YD306_YEAST ------NY-------------------------
SRF1_YEAST ---------------------------------
YM8M_YEAST ---------------------------------
YCT1_YEAST SSIEEGKVVTHENNLAVFDLTDLENETFIYPL
```

A.2.8. FASTA alignment format

It is possible to describe a multiple alignment in FASTA format (see Format A.10). The default output created by MUSCLE (Edgar, 2004) follows this format.

Format A.10: MUSCLE output format.

```
>YD306_YEAST
MANKSRPKKIKAPYRKYVAGEGFSSTRNDNKAKEFTITIPEDAELIETPQGSYYYDETND
TIVKLTRLSNEKKDKKGRKQSPSSSSTSSSKGEKNGKVIESEEARMHSVSVKMVLPWEIQ
HRIIHYLDIPEKEE-----KLNKTANGKKTTTGINMNYLLVCRNWYAMCLPKLY---YAP
ALTSKNFNGFVDTIIINKKKNLGHYVFELNLSTILQSGRNSFVSKLLRRCCSN--LTKFI
APQTSFGYAPLISLKSCHDLKFLDLGLVSETVKLKELFSAIKNFTKLTHLSFPRSSIDCQ
GFQDIQ---WPQNLRYLKLSGGITNEFVIDTKWPTTITTLEFSYCPQITELSIYSL----
----LSQIGDNLKHLFFHYPMP--SLAENSLDHVFTYCANLISLQLMVDYCSKWCFSEFM
LSKLVEYDRPLKTLYLECSGSLGLASKIHPDDLTIAI---------------LESRLPCL
KNICVSPKLGWNMKSDEVA-----------------------------------------
--------------------------------------------------DLVVSLEDQDG
SLYLNY----------
>SRF1_YEAST
------------MGDSNSSQEAYSDTTSTNA----SRI--ADQNQLNLN----VDLEKNQ
TVRKSGSLEA-------LQNAKIHVPKHSDGSPLDYPKLNTYTFVPTTVPPYVLEAQFDK
LRLQ-DKGTVDGNV------TDDKNLPKEFKWGQFA---------STIGCHSAYTRDQNY
NPSHKSYDGYSLSSSTSSKNAALREILG------------DMCSEWGGEERLEGVLHSEI
GANLEFNTTEERKEWLQYI----------E---------------KVKDFYYGDNKKNPE
SPESVHNKVYKSD-WVNELNKEREKWRRLKQR---KLQQWRPPLTSLLLDNQYLILGLRI
FTGILSCISLALAIKIFQNSRSNNTISESKIGQQPSTIMAICVNAVAIAYIIYIAHDEF-
AGKPVGLRNPLSKLKLILLDLLFIIFSSANLALAFNTR--------------FDKEWVCT
SIRRSNGSTYGYPKIPRICRKQEALSAFLFVALFMW------------------VITFSI
SIVRVV-----------------------------------------EKVSSITNRN-
----------------
>ENV10_YEAST
-----------------MAGKAGRKQASSNA----KII----------------------
--------------------------------------------------QGLYKQ
VSL--FLGMAIVRL----------FISRKVTIGQWI--KLVA--LNVPMFVALYIIVLSG
KPKYDGNRVVKQGIDLNDNTNLISYFFDL-----------------------------
----------------IYLSLFGNIGIIAF---------------RTFKFW---------
---------W-------------------------------------------------
-----------------------------------CLLLC--PIYAGYKLYGLKNMFM
PGAQQTQADNRS-----------------------------------------------
```

```
-KNANEGQSKSKRQMKRE-----------------------------------------
---------------RRGET-----------------------------DSKIKYKYR--
----------------
>YM8M_YEAST
MFSIFKKKTSVQGTDSEIDEKI---TVKAKD----KVVVSTEDEEVTTI----VSSTKST
QVTNDSPWQDPTYFSS-FGKELMFIATCMLAQLLNQAGQTHALCIMNVLSKSFNSEANNQ
AWLMASFPLAAGSFILISGRLGDIYGLKKMLIVGYV--IVIV--WSIISGLSKYSNSDAF
FITSRAFQGVGIAFILPNIMGLVGHVYKVG------SFRKNIVISFIGACAPTGGMFGGL
FGGLIVTEDPNQWPWVFYA--FGIATFLSL----------------LMAWYSIPNNVPTNI
HGLSMD---W-TG-SALAIIGLILFNFVWNQA---PIVGWDKPYIIVLLIISVIFL----
-----------VAFFVYESKYAEVPLLPRAMTKNRHMIMILL--AVFLGWGSFGIWTFYY
VSFQLNLRHYSPVWTGGTYFVFVIFGSMAAFFVAFSIKRLGPALLLCFSLMAFDAGSIM-
FSVLPVEQSYWKLNFAMQAILCFGMDLSFPASSIILSDGLPMQYQGMAGSLVNTVINYSA
SLCLGMGGTVEHQINKSGNDLLKGY---------RAAVYLGVGLASL--GVVISVTYMLE
NLWNRHRKSEDRSLEA
>YCT1_YEAST
--------------MSKVDVKIGADSISSSD----EILVPSRLADVTLA----FMEENDA
AVPEITPEQEKKLKRKLFLTIFTFVSAINLLLYMDKATLSYDSILGFFEDTGLTQNTYNT
VNTLFYVGFAIGQF---PGQ----YLAQKLPLGKFLGGLLAT--WTILIFLSCTAYNFSG
VVALRFFLGLTESVVIPILITTMGMFFDAS------E-RAAAQPFFFAACMGSPIPTGFI
AYGVLHITNPSISLWKIFTIIIGGLTFIMT----------------VVVILWFPNN-PADV
KFFSIQERVWIIR-RVQASTGSSIEQKVFKKS---QFREAMKDYITWLF--GLFFL----
-----------LQQLANNLPYQQ-NLLFEGMGGVDALGSTLV--SV-AGAG-FAVVCAFI
ATLMLAKWKNISALTAIFWTLPALVGSIAAAALPWD-NKIGILANICMAGQIFGIPFIIA
LSWASSSASGYTKKLTRSSVSLFAMGIANIISPQIWREKDSPRF--LPAWIVQIVLSFSL
APAILL--LIHFILKRRNNQRLKNYDENLQNYLDRIQLIESENPSSIEEGKVVTHENNLA
VFDLTDLENETFIYPL
```

A.2.9. PHYLIP multiple alignment format

The PHYLIP multiple alignment format (see Format A.11) is the one used by the PHYLIP package (Felsenstein, 2009). This format is similar to the other alignment formats already seen, but the sequence identifiers are just stated once, and its first line shows the number of sequences been aligned and the total length of the alignment, including gaps.

Format A.11: PHYLIP output format.

```
   5  563
ENV10_YEAS ---------- ---------- ---------- ---------- ----------
YM8M_YEAST ----MFSIFK KKTSVQGTDS EIDEKITVKA KDKVVVSTED EEVTTIVSST
SRF1_YEAST MGDSNSSQEA YSDTTSTNAS RIADQNQLNL NVDLEKNQTV RKSGSLEALQ
YCT1_YEAST ---------- ----MSKVDV KIGADSISSS DEILVPSRLA DVTLAFMEEN
YD306_YEAS --------MA NKSRPKKIKA PYRKYVAGEG FSSTRNDNKA KEFTITIPED

        ---------- ---------- ---------- ----MAGKAG RKQASSN---
        KSTQVTNDSP WQDPTYFSSF GKELMFIATC MLAQLLNQAG QTHALCI---
        NAKIHVPKHS DGSPLDYPKL NTYTFVPTTV PPYVLEAQFD KLRLQDK---
        DAAVPEITPE QEKKLKRKLF LTIFTFVSAI NLLLYMDKAT LSYDSILGFF
        AELIETPQGS YYYDETNDTI VKLTRLSNEK KDKKGRKQSP SSSSTSSSKG

        ---AKIIQGL YKQVS----- LFLGMAIVR- LFISRKVTIG ------QWIK
        ---MNVLSKS FNSEANNQAW LMASFPLAAG SFILISGRLG ------DIYG
        ---GTVDGNV TDDKNLPKEF KWGQFASTIG CHSAYTRDQN Y-NPSHKSYD
        E-DTGLTQNT YNTVN-TLFY VGFAIGQFPG QYLAQKLPLG --KFLGGLLA
        EKNGKVIESE EARMHSVSVK MVLPWEIQHR IIHYLDIPEK EEKLNKTANG

        LVALNVP-MF VALYIIVLSG KPKYDGN--- -------RVV KQGIDLNDNT
        LKKMLIVGYV IVIVWSIISG LSKYSNSDAF FITSRAFQGV GIAFILPNIM
        GYSLSSSTSS KNAALREILG DMCSEWGGEE RLEGVLHSEI GANLEFNTTE
        TWTILIFLSC TAYNFSGVVA LRFFLGLTES VVIPILITTM GMFFDASERA
        KKTTTGINMN YLLVCRNWYA MCLPKLYYAP ALTSKNFNGF VDTIIINKKK

        NLISYFFDL- ---------- ----------I YLSLFG---- ----------
        GLVGHVYKVG SFRKNIVISF IGACAPTGGM FGGLFGGLIV TEDPNQWPWV
        ERKEWLQYIE KVKDFYYGDN KKNPESPESV HNKVYKSDWV N---------
        AAQPFFFAAC MGSPIPTGFI AYGVLHITNP SISLWKIFTI IIGGLTFIMT
        NLGHYVFELN LSTILQSGRN SFVSKLLRRC CSNLTKFIAP QTSFG-----

        ---------- ---------- ---------- -------NIG IIAFRTFKFW
        FYAFGIATFL SLLMAWYSIP NNVPTNIHGL SMDWTGSALA IIGLILFNFV
        ---------- ---------- ---------- ELNKEREKWR RLKQRKLQQW
        VVVILWFPNN PADVKFFSIQ ERVWIIRRVQ ASTGSSIEQK VFKKSQFREA
```

```
---------- ---------- ------YAPL ISLKSCHDLK FLDLGLVSET

W--------- ---------- ---------- ---------- ----------
WNQAPIVGWD KPYIIVLLII SVIFLVAFFV YESKYAEVPL LPRAMTKNRH
RPPLTSLLLD N--------- ---------- ---------- ---------Q
MKDYITWLFG LFFLLQQLAN NLPYQQNLLF EG-------- ------MGGV
VKLKELFS-- ---------- ---------- ---------- ---------A

-CLLLCPIYA GYKLYGLKNM FMPGAQQTQA DNR------- ----------
MIMILLAVFL GWGSFGIWTF YYVSFQLNLR HYSPVWTGGT YFVFVIFGSM
YLILGLRIFT GILSCISLAL AIKIFQNSRS NNTISESKIG QQPSTIMAIC
DALGSTLVSV AGAGFAVVCA FIATLMLAKW KNISALTAIF WTLPALVGSI
IKNFTKLTHL SFPRSSIDCQ GFQDIQWPQN LRYLKLSGGI TNEFVIDTKW

---------- ---------- ---------- ---------- ----------
AAFFVAFSIK RLGPALLLCF SLMAFDAGSI MFSVLPVEQS YWKLNFAMQA
VNAVAIAYII YIAHDEFAGK PVGLRNPLSK LKLILLDLLF IIFSSANLAL
AAAALPWDNK IGILANICMA GQIFGIPFII ALSWASSSAS GYTKKLTRSS
PTTITTLEFS YCPQITELSI YSLLSQIGDN LKHLFFHYPM PSLAENSLDH

---------- ---SKNANEG QSKSKRQMK- ---------- ----------
ILCFGMDLSF PASSIILSDG LPMQYQGMAG SLVNTVINYS ASLCLGMGGT
AFNTRFDKEW VCTSIRRSNG STYGYPKIPR ---------- ----------
VSLFAMGIAN IISPQIWREK DSPRFLPAWI VQIVLSFSLA PAILLLIHFI
VFTYCANLIS LQLMVDYCSK WCFSEFMLSK LVEYDRPLKT LYLECSGSLG

-RERRG---E TDSKIKYKYR ---------- ---------- ----------
VEHQIN---K SGNDLLKGYR AAVYLGVGLA SLGVVISVTY MLENLWNRHR
-ICRKQ---E ALSAFLFVAL FMWVITFSIS IVRVVEKVSS ITNRN-----
LKRRNNQRLK NYDENLQNYL DRIQLIESEN PSSIEEGKVV THENNLAVFD
LASKIHP-DD LTIAILESRL PCLKNICVSP KLGWNMKSDE VADLVVSLED

---------- ---
KSEDRSLEA- ---
---------- ---
LTDLENETFI YPL
QDGSLYLNY- ---
```

A.2.10. Newick tree format

The Newick tree format was created by the English mathematician Arthur Cayley in 1857. This computer-readable format is widely used in phylogenetics to represent evolutionary tools. It consists of nested parentheses, node names, and node distances (see Format A.12). It is easy to visualize (see Figure A.3).

Format A.12: Newick tree format.

```
(
(
ENV10_YEAST:0.43550,
SRF1_YEAST:0.47610)
:0.01328,
(
YD306_YEAST:0.48315,
YM8M_YEAST:0.44991)
:0.00441,
YCT1_YEAST:0.46869);
```

A.2.11. Orthostrapper output format

Orthostrapper (Storm and Sonnhammer, 2002) output (see Format A.13) is composed of two columns. The first is related to the identifier of the target sequences compared to the query. The second shows the query identifier followed by the values that indicate the orthology relationship. In the case of just comparing one query with a set of targets, 1 is assigned to the most likely orthologs and 0 for the rest.

Figure A.3: Newick tree visualization for the example given on Format A.12.

Format A.13: Orthostrapper output format.

```
                  DCXR_HUMAN
sp|Q91XV4|DCXR_MESAU        1
sp|Q1JP75|DCXR_BOVIN        1
sp|Q91X52|DCXR_MOUSE        1
sp|Q920P0|DCXR_RAT          1
sp|Q920N9|DCXR_CAVPO        1
sp|Q29529|CBR2_PIG          0
sp|P08074|CBR2_MOUSE        0
sp|Q8JIS3|DER_CHICK         0
sp|Q21929|DCXR_CAEEL        0
```

A.3. Other entry and file formats

A.3.1. GFF

The General Feature Format (GFF) (Welcome Trust Sanger Institute, 2012; Ensembl Genome Bowser, 2015) is a tab-separated format widely used to represent genomic features. It consists of 9 columns:

1. **Seqid:** the name or accession number of the landmark associated to the feature.
2. **Source:** free text that describes the algorithm or method used to generate the feature.
3. **Type:** feature's type, for example *Gene*, *Similarity*, or *Annotation*.
4. **Start:** start position of the feature.
5. **End:** end position of the feature.
6. **Score:** a floating point value. In case of MASSA, the score is the *e-value* associated to the feature, if any.

7. **Strand:** strand of the feature (i.e., "+" or "-").

8. **Phase:** integer from 0 to 2 that indicates the number of bases that have to be removed from the beginning of the feature in order to access the reading frame. Thus, if the next codon starts at the first base, the phase is 0; and so on.

9. **Attributes:** A list of tag-value pairs related to the feature, separated by semicolons. Some of these tags are already defined, such as *Target*, that indicates the target sequence of an alignment. This attribute is described as: *Target TargetType:AccessionNumber TargetStart TargetEnd*. In MASSA, *TargetType* can be *Protein*, *Nucleotide*, or *Domain*.

The GFF format (see Formar A.7) has changed over the years and there are some versions that differ from the original, like GFF3 (Stein, Lincoln, 2013). Despite their differences, both formats hold the same information. MASSA employs GFF and uses the GMOD (Generic Model Organism Database (GMOD) Project, 2013) script *bp_load_gff.pl* to load its information to a MySQL database. This script can also create the database, when it does not exist, and upload FASTA files (see Appendix A.2.1) to it.

MASSA produces two GFF files when each of its subsystem finishes its work. MASSAPipe creates an intermediate file that contains information of all candidates and the query sequences. This file is uploaded to the RDB and its informartion is used by MASSAInference to infer the annotation. The second one is the system output where the annotation is described. This file is delivered to the user and can have, besides the *Target*, the following *tags* as *attributes*:

- **Description:** target description transferred to the query as annotation. See Section 5.3, Table 3.2.
- **Bits:** bit score of the alignment. See Section 5.3, Table 3.11.
- **FradId:** percent of identity of the alignment. See Section 5.3, Table 3.12.
- **GO:** GO terms associated with the annotation. See Section 5.3, Table 3.18.
- **Source_score:** source score of the target sequence. This value varies from 0 to 1 according to the level of reliability of the database. Manual curated databases, like SwissProt, get higher scores than automatic ones. The system configuration file (see A.3.3.1) has pre-defined values for specific databases. See Section 5.3, Table 4.
- **Orthology_score:** orthology score given by Orthostrapper (Storm and Sonnhammer, 2002). See Section 5.3, Table 4.
- **Domain_score:** value that indicates the similarity of the domains/families found in the query related to the ones existing in the target sequence. See Section 5.3, Table 4.
- **Specificity_score:** integer that represents if the domain is specific (see 2.5.1.3) (1) or not (0). See Section 5.3, Table 4.

- **Conserved_score:** value that indicates how conserved in the query sequence the important sites and regions described in the candidate are. See Section 5.3, Table 4.

A.3.2. GO output

GO information is stored in a MySQL database. The format presented here is the one provided by the GO's website, and it is just presented for illustrative purposes. In MASSA, the GO information is obtained through queries to a local version of the GO database. Format A.14 shows the information available on the database.

Format A.14: GO output format example[7]

```
Accession
   GO:0007165
Name
   signal transduction
Ontology
   biological_process
Synonyms
   alt. id: GO:0023033
   signaling pathway
   signalling pathway
   signaling cascade
   signalling cascade
Definition
   The cellular process in which a signal is conveyed to trigger a change in the activity or state of a cell.
         Signal transduction begins with reception of a signal (e.g. a ligand binding to a receptor or receptor
         activation by a stimulus such as light), or for signal transduction in the absence of ligand,
         signal-withdrawal or the activity of a constitutively active receptor. Signal transduction ends with
         regulation of a downstream cellular process, e.g. regulation of transcription or regulation of a metabolic
         process. Signal transduction covers signaling from receptors located on the surface of the cell and
         signaling via molecules located within the cell. For signaling between cells, signal transduction is
         restricted to events at and within the receiving cell. Source: GOC:go_curators, GOC:mtg_signaling_feb11
Comment
   Note that signal transduction is defined broadly to include a ligand interacting with a receptor, downstream
         signaling steps and a response being triggered. A change in form of the signal in every step is not
         necessary. Note that in many cases the end of this process is regulation of the initiation of
         transcription. Note that specific transcription factors may be annotated to this term, but core/general
         transcription machinery such as RNA polymerase should not.
History
   See term history for GO:0007165 at QuickGO
Subset
   goslim_metagenomics
   gosubset_prok
   goslim_plant
   goslim_aspergillus
   goslim_candida
   goslim_chembl
   goslim_generic
```

A.3.3. MASSA Files

A.3.3.1. General Configuration File

The GCF is the file where MASSA configuration is defined. It comprises all the information related to the system defined by self-explanatory variables (see Format A.19).

Format A.15: MASSA configuration file used to annotated the data from CAFA (see Section 4.5.1).

[7] http://amigo.geneontology.org/amigo/term/GO:0007165. [Online: accessed 5-September-2015]

```
############
# GENERAL #
############

[DATABASE]
DB_USER = dxavier
DB_PASSWD = jex5135
DB_HOST = localhost
DB_PREFIX = massa
DB_PORT = 3306

[GENERAL]
DEFAULT_CONFIG_FILE = /path/MASSA.properties
OUT_PATH = /home/dxavier/Desktop/MASSA/Ejemplo
OUT_PATH_TMP = /home/dxavier/Desktop/MASSA/Ejemplo/tmp
DOWNLOAD_TMP = /home/dxavier/Desktop/MASSA/Ejemplo/download
DBUPCONFIG = /home/dxavier/Desktop/MASSA/DBUPCONFIG.properties
GFF_OUT_DIR = gff
ANNOT_OUT_DIR = annot
TRANSLATED_OUT_DIR = translated
LOG_OUT_DIR = log
MASSA_PIPE_PREFFIX = MP
MASSA_INFERENCE_PREFFIX = MI
MASSA_PIPE_NAME = MASSA_PIPE
MASSA_INFERENCE_NAME = MASSA_INFERENCE
#MADATORY FIELDS NAME
FASTA_INPUT_FIELD = fasta.sequenceInput
FASTA_SOURCE_FIELD = fasta.source
FASTA_METHOD_FIELD = fasta.method
FASTA_DATABASE_FIELD = fasta.database
FASTA_TAXON_FIELD = fasta.taxon
FASTA_ORGANISM_FIELD = fasta.organism
FASTA_PROJECT_FIELD = fastq.project
CONTAINER_FIELD = fasta.container
MASSA_SUBSYSTEM_FIELD_NAME = fasta.subsystem
INFA_FIELD_NAME = INFA
SEQN_FIELD_NAME = SEQN
ORDER_FIELD_NAME = order
DATABASE_LIMIT_MIN_FIELD = limitMin
DATABASE_NROW_FIELD = numRows
ANNOTATION_OUT_PATH_FIELD = outPath



;Number of agents launched to process the queries
INFA_NUMBER = 1
;Get GO for each annotation
GET_GO = true
#GO DATABASE CONFIGURATIONS
GO_DBNAME = go_latest_lite
HOMEGO_DBNAME = id_to_go1
#go_local
#id_to_go1


###############
# MASSA PIPE #
###############

[SCRIPTS_PATH]
BP_LOAD = bp_load_gff.pl
NCBI_UPDATE_SCRIPT = /home/dxavier/Desktop/MASSA/scripts/update_blastdb.pl
UNIPROT_UPDATE_SCRIPT = /home/dxavier/Desktop/MASSA/scripts/update_uniprot_blastdb.pl
CDD_FILES_UPDATE_SCRIPT = /home/dxavier/Desktop/MASSA/scripts/update_cdd_files.pl
BLAST2GFF=/home/dxavier/Desktop/MASSA/scripts/BLAST2gff.pl
#EXECBLAST=/home/dxavier/Desktop/MASSA/scripts/exec_BLAST.pl
FASTA2GFF=/home/dxavier/Desktop/MASSA/scripts/FASTA2gff.pl
EXEC_FASTATRANSLATE=/home/dxavier/Desktop/MASSA/scripts/exec_fastatranslate.pl
EXEC_RPSBLAST=/home/dxavier/Desktop/MASSA/scripts/exec_RPSBLAST.pl
EXEC_INTERPROSCAN=/home/dxavier/Desktop/MASSA/scripts/exec_interproscan.pl
INTERPROSCAN2GFF=/home/dxavier/Desktop/MASSA/scripts/InterProScan2gff.pl
EXEC_ORTHOLOGY=/home/dxavier/Desktop/MASSA/scripts/execOrthostrapper.pl
ORTHOLOGY_SCRIPT=/opt/orthostrapper/Orthostrapper.jar

[SUFFIX]
RPS_BLAST_SUFFIX = .rpsblast
BLASTN_SUFFIX = .blastn
BLASTX_SUFFIX = .blastx
FASTATRANSLATED_SUFFIX = .fastatranslated

[DEFAULT VALUES]
```

```
FASTA_DEFAULT_SOURCE = myAssembly
;MIRA, 454, Velvet, etc
FASTA_DEFAULT_METHOD = mySequencing
;Contig, EST, READ, Singleton, etc
MAX_BLACKBOARD_ROWS = 100
NUM_PROCESSORS = 2
MAX_EVALUE = 1.0E-05
DB_UPDATE_NUM_DAY = 1000
CANDIDATES_NUM = 5

####################
# MASSA INFERENCE #
####################

[EXTRA_INFO_PATHS]
#FILES PATHS
#file that contains the relation domain/family|superfamily
#[CDD_PATHS]
#list of NCBI-curated and imported domain models that are members of CDD superfamilies, along with the superfamily
      accession (cl*) to which each domain model belongs
FAMILY_LINKS = /opt/cdd_info/family_superfamily_links
#information about conserved family features (such as binding and catalytic sites) as recorded for NCBI-curated CD
      models (scope C: NCBI-curated domain models)
CDDANNOT = /opt/cdd_info/cddannot.dat
#domain-specific score thresholds used by CD-Search tool to determine whether hits to NCBI-curated domain models
      are specific or non-specific
BITSCORE_FILE = /opt/cdd_info/bitscore_specific.txt
CDD_VERSION = /opt/cdd_info/cdd.versions



[GO_FILES]
SMART2GO = /opt/GO_terms/smart2go
PFAM2GO = /opt/GO_terms/pfam2go
TIGRFAMS2GO = /opt/GO_terms/tigrfams2go
PRODOM2GO = /opt/GO_terms/prodom2go
PRINTS2GO = /opt/GO_terms/prints2go
PIRSF2GO = /opt/GO_terms/pirsf2go
INTERPRO2GO = /opt/GO_terms/interpro2go
COG2GO = /opt/GO_terms/cog2go
PROSITE2GO = /opt/GO_terms/prosite2go
GENE2ACCESSION = /opt/ncbi/gene2accession
GENE2GO = /opt/ncbi/gene2go
UNIPROT2GO = /opt/GO_terms/gp_association.goa_uniprot
PDB2GO = /opt/GO_terms/gene_association.goa_pdb
#ID MAPPING (UniProt <-> RefSeq <-> Ensembl)
IDMAPPING = /opt/GO_terms/idmapping.dat


[SCRIPT_PATH]
#command to the shell script execution
SHELL = sh
```

A.3.3.2. Configuration File

The CF describes the information related to the pipeline to be executed. It comprises the name of the tool, its purpose, and the location of its SDB if applicable with some thresholds (e.g., maximum *e-value* and number of candidates), number of threads. Besides, it contains some optional information (i.e., name and taxonomy identifier) of the specie to be annotated.

Format A.16: MASSA configuration file exemple.

```
############
# PIPELINE #
############

#FASTA
fasta.organism = Dictyostelium discoideum
fasta.taxon = 44689

#BLAST_1
blast_1.DB = /path_to_nr/nr
```

```
blast_1.program = blastx
blast_1.gffTargetDef=Protein
blast_1.numProcessors = 4
blast_1.maxEvalue = 1.0E-20
blast_1.candidatesNum = 5

#RPSBLAST
rpsBlast_1.DB = /path_to_cdd/Cdd
rpsBlast_1.program = rpsblast
rpsBlast_1.gffTargetDef=Domain
rpsBlast_1.maxEvalue = 1.0E-05
rpsBlast_1.candidatesNum = 20

#InterProScan
interproscan_1.program = interproscan
interproscan_1.gffTargetDef=Domain

#BLAST_4
blast_4.DB = /path_to_uniprot/uniprot
blast_4.program = blastx
blast_4.gffTargetDef=Protein
blast_4.maxEvalue = 1.0E-05
blast_4.candidatesNum = 10

#Orthology
orthology_1.program = orthostrapper
```

A.3.3.3. Task Log File

The TLF is a tabular file that comprises the information existent in the TB (see Section 4.3.1) for a given *project*. It has five columns. The first one is related to the name of the *project*. The second presents the *container* where the *project* is being executed. The third describes the *task* or *substask* that is running. Every time an IA starts/ends a *task*, the column shows the word "Task", and when it is removed from the *container*, it shows "Kill'. The fourth one is the status of the *task* or *subtask*. It can be "Started", "Finished", or "Failed". The fifth one is the date and time when the *task* or *subtask* started or finished. Since MASSAPipe and MASSAInference are two independent systems they generate different TLFs. Format A.17 and A.18 exemplify TLF for MASSAPipe and MASSAInference respectively.

Format A.17: MASSAPipe Task Log File example.

```
ECOLI   Container-1_fasta       Task          Started     08/01/2014 16:14:15
ECOLI   Container-1_interproscan_1    Task          Started     08/01/2014 16:14:15
ECOLI   Container-1_fasta       FASTA2gff.pl       Started     08/01/2014 16:14:15
ECOLI   Container-1_blast_1        Task          Started     08/01/2014 16:14:15
ECOLI   Container-1_blast_4       Task          Started     08/01/2014 16:14:15
ECOLI   Container-1_rpsBlast_1       Task          Started     08/01/2014 16:14:15
ECOLI   Container-1_interproscan_1    exec_interproscan.pl     Started     08/01/2014 16:14:15
ECOLI   Container-1_blast_4       blastall         Started     08/01/2014 16:14:15
ECOLI   Container-1_rpsBlast_1       exec_fastatranslate.pl   Started      08/01/2014 16:14:15
ECOLI   Container-1_fasta       FASTA2gff.pl        Finished    08/01/2014 16:14:18
ECOLI   Container-1_fasta       Task          Finished    08/01/2014 16:14:18
ECOLI   Container-1_fasta       Kill          Started     08/01/2014 16:14:18
ECOLI   Container-1_fasta       Kill          Finished   08/01/2014 16:14:18
ECOLI   Container-1_rpsBlast_1       exec_fastatranslate.pl   Finished    08/01/2014 16:14:18
ECOLI   Container-1_rpsBlast_1       exec_RPSBLAST.pl      Started     08/01/2014 16:14:18
ECOLI   Container-1_blast_4       blastall         Finished  08/01/2014 16:24:35
ECOLI   Container-1_blast_4       BLAST2gff.pl      Started     08/01/2014 16:24:35
ECOLI   Container-1_rpsBlast_1       exec_RPSBLAST.pl      Finished  08/01/2014 16:32:17
ECOLI   Container-1_rpsBlast_1       BLAST2gff.pl       Started     08/01/2014 16:32:17
ECOLI   Container-1_rpsBlast_1       BLAST2gff.pl       Finished  08/01/2014 16:45:08
ECOLI   Container-1_rpsBlast_1       Task          Finished  08/01/2014 16:45:08
ECOLI   Container-1_rpsBlast_1       Kill          Started     08/01/2014 16:45:08
ECOLI   Container-1_rpsBlast_1       Kill          Finished  08/01/2014 16:45:08
ECOLI   Container-1_blast_4       BLAST2gff.pl     Finished  08/01/2014 17:20:24
ECOLI   Container-1_blast_4       Task          Finished  08/01/2014 17:20:24
ECOLI   Container-1_blast_4       Kill          Started     08/01/2014 17:20:24
```

```
ECOLI  Container-1_blast_4       Kill          Finished  08/01/2014 17:20:24
ECOLI  Container-1_interproscan_1     exec_interproscan.pl     Finished    08/01/2014 17:23:37
ECOLI  Container-1_interproscan_1     InterProScan2gff.pl     Started     08/01/2014 17:23:37
ECOLI  Container-1_interproscan_1     InterProScan2gff.pl     Finished  08/01/2014 17:23:39
ECOLI  Container-1_interproscan_1     Task          Finished    08/01/2014 17:23:39
ECOLI  Container-1_interproscan_1      Kill          Started    08/01/2014 17:23:39
ECOLI  Container-1_interproscan_1      Kill          Finished  08/01/2014 17:23:39
ECOLI  Container-1_blast_1        blastall         Finished  08/01/2014 20:50:05
ECOLI  Container-1_blast_1       BLAST2gff.pl       Started     08/01/2014 20:50:05
ECOLI  Container-1_blast_1       BLAST2gff.pl       Finished  08/01/2014 22:15:22
ECOLI  Container-1_blast_1       Task          Finished  08/01/2014 22:15:22
ECOLI  Container-1_blast_1        Kill          Started     08/01/2014 22:15:22
ECOLI  Container-1_blast_1       Kill          Finished    08/01/2014 22:15:22
```

Format A.18: MASSAInference Task Log File example.

```
ECOLI  Container-1_Inference_0  Task        Started     08/04/2014 08:16:57
ECOLI  Container-1_Inference_0  Inference      Started     08/04/2014 08:16:57
ECOLI  Container-1_Inference_0  Inference      Finished  08/04/2014 08:42:04
ECOLI  Container-1_Inference_0  Task        Finished  08/04/2014 08:42:04
ECOLI  Container-1_Inference_0  Kill        Started     08/04/2014 08:42:04
ECOLI  Container-1_Inference_0  Kill        Finished  08/04/2014 08:42:04
```

A.3.3.4. Potential Candidates List File

The PCLF presents a sorted list of the likely annotation candidates together with their scores (see Section 5.3, Table 4) and attributes (see Section 5.3, Table 3) took into consideration at the inference process. It can contain the information of one or more queries. The query information starts with the symbol ">" and the query identifier. The lines that follows this entry are related to the targets found. The are organized as:

target identifier: *total score* (SOURCE: *source score* ORTHOLOGY: emphorthology score SPECIFICITY: *specificity score* DOMAIN: *domain score* CONSERVED: *conserved site score*), evalue: *e-value*, bitscore: *bit score*, identity: *identity*, annotation: *candidate description*, GO: *GO terms*

Format A.19: PCLF for the entrey AGP_ECOLI (Glucose-1-phosphatase from *Escherichia coli*).

```
>AGP_ECOLI
sp|O33921|AGP_SALTY: 5.0 (SOURCE: 1.0 ORTHOLOGY: 1.0 SPECIFICITY: 0.0 DOMAIN: 1.0 CONSERVED: 1.0), evalue: 0.0,
      bitscore: 694.0, identity: 0.840909090909091, annotation: Glucose-1-phosphatase, GO:GO:0003993 GO:0008877
      GO:0016787 GO:0042597
gi|446967016|ref|WP_001044272.1|: 5.0 (SOURCE: 0.0 ORTHOLOGY: 1.0 SPECIFICITY: 1.0 DOMAIN: 1.0 CONSERVED: 1.0),
      evalue: 0.0, bitscore: 801.0, identity: 0.961259079903148, annotation: glucose-1-phosphatase/inositol
      phosphatase, GO:GO:0003993
gi|446967018|ref|WP_001044274.1|: 5.0 (SOURCE: 0.0 ORTHOLOGY: 1.0 SPECIFICITY: 1.0 DOMAIN: 1.0 CONSERVED: 1.0),
      evalue: 0.0, bitscore: 800.0, identity: 0.958837772397094, annotation: glucose-1-phosphatase/inositol
      phosphatase, GO:GO:0003993
gi|446967107|ref|WP_001044363.1|: 4.0 (SOURCE: 0.0 ORTHOLOGY: 0.0 SPECIFICITY: 1.0 DOMAIN: 1.0 CONSERVED: 1.0),
      evalue: 0.0, bitscore: 799.0, identity: 0.958837772397094, annotation: glucose-1-phosphatase/inositol
      phosphatase, GO:GO:0003993
gi|446967034|ref|WP_001044290.1|: 4.0 (SOURCE: 0.0 ORTHOLOGY: 0.0 SPECIFICITY: 1.0 DOMAIN: 1.0 CONSERVED: 1.0),
      evalue: 0.0, bitscore: 799.0, identity: 0.956416464891041, annotation: glucose-1-phosphatase/inositol
      phosphatase, GO:GO:0003993
gi|446674395|ref|WP_000751741.1|: 4.0 (SOURCE: 0.0 ORTHOLOGY: 0.0 SPECIFICITY: 1.0 DOMAIN: 1.0 CONSERVED: 1.0),
      evalue: 0.0, bitscore: 795.0, identity: 0.951573849878935, annotation: glucose-1-phosphatase/inositol
      phosphatase, GO:GO:0003993
sp|Q52309|AGP_PRORE: 4.0 (SOURCE: 1.0 ORTHOLOGY: 0.0 SPECIFICITY: 0.0 DOMAIN: 1.0 CONSERVED: 1.0), evalue:
      1.0E-148, bitscore: 435.0, identity: 0.535248041775457, annotation: Glucose-1-phosphatase, GO:GO:0003993
      GO:0008877 GO:0016787 GO:0042597
```

Appendix B

Methodologies

It is beyond a doubt that all our knowledge
begins with experience.

Immanuel Kant

B.1. CommonKADS

The development of KBSs requires the understanding and structuring of the knowledge in a way it can be incorporated into the system. Knowledge Engineering has methodologies to facilitate knowledge analysis and modeling, helping to build well-structured architectures for systems that are easier to use and maintain (Schreiber et al., 2000).

CommonKADS (Schreiber et al., 2000) is a standard approach for knowledge analysis and KBS development. This flexible and powerful methodology can be applied in any context, independently of its complexity, and it can also be reused for similar problems. Through it, it is possible to identify the most suitable strategy to propose a solution to a problem, and to establish the methodological bases to tackle it in a general way. Moreover, CommonKADS (Schreiber et al., 2000) improves communication, and standardization, and promotes technology support and the availability of reusable components.

The CommonKADS (Schreiber et al., 2000) process (see Figure 5.1.2) is organized around three groups of models:

- The *Context* analysis aims to understand why a RBS could be a potential solution or help for a problem. In order to answer this question, the organizational environment is studied in detailed, looking for crucial success factors for a knowledge approach. Therefore, an analysis of the major features of the organization (i.e., *Organization Model*) is carried out, together with an analysis of the global task layout (i.e., *Task Model*), and the agents in charge to execute the task (i.e., *Agent Model*).

- The *Concept* focuses on the nature and structure of the knowledge involved, providing a conceptual description of problem-solving functions and data. It comprises

analyses regarding the type of knowledge and its structures used to perform a task (i.e., *Knowledge Model*) and the communication between agents (i.e., *Communication Model*).

- The *Artifact* focuses on the technical aspects of the implementation. The *Design Model* describes how to implement the knowledge in the system.

All these models can give valuable insights for the knowledge modeling and system development. Nevertheless, their construction depends on the goals of the project and the experience of the developers running it (Schreiber et al., 2000).

Based on the previous experience with the studied domain and on the important role the knowledge plays in the annotation problem, this work gives a special attention to the *Knowledge Model*, focusing on the analysis and structure of the expertise required to infer precise annotations.

B.1.1. Knowledge model

The *Knowledge Model* is organized in three knowledge categories, hierarchically organized as depicted in Section 5.1, Figure 3. The next subsections describe briefly each category.

B.1.1.1. Domain knowledge

The *Domain Knowledge* comprises the knowledge regarding the domain and the types of information the application deals with. Its first basic element, the *Domain Schema*, describes the data schematically through the association of attributes and values to instances of the domain.

The second element, the KB, comprehends the instances of knowledge types defined in the *Domain Schema*.

In the case of this work, the KB contains: the rules formulated to infer the best annotation; the databases used to obtain the candidates; and all the files that provide extra information.

B.1.1.2. Inference knowledge

The *Inference Knowledge* describes how the *Domain Knowledge* is going to be used. This knowledge consists of *inferences*, *knowledge roles*, and *transfer functions*. The first ones use the knowledge in the KB to infer new information. *Inferences* are like black boxes that represent machine reasoning actions (e.g., verify, order, generate, predict, and evaluate) as function of its input and output. These inputs and outputs are classified based on the roles they play in the reasoning process. A *knowledge role* is said to be *dynamic* when it varies according to the invocation of the *inference*, and *static* when it keeps stable during a certain part of the process. The *transfer function* is in charge of communicating the inference with the world, transferring information between them.

B.1.1.3. Task knowledge

The *Task Knowledge* plays a very important role in the system design, since it describes the goals to be accomplished through the knowledge and the strategies to be carried out to achieve them. This knowledge can be decomposed recursively into simpler subtasks and at its lower level, it is associated with *inferences* and *transfer functions* as depicted in Section 5.1, Figure 7.

The principal elements of this category are the *task* and the *task method*. The first one describes a complex reasoning goal in terms of inputs and outputs, answering the question "what can be done?". The second focus on how the *task* can be carried out, decomposing it in a set of subfunctions and a control structure where the order of execution of these functions is specified.

CommonKADS (Schreiber et al., 2000) has a collection of templates that can be used to model a task. These templates can be applied in different domains without the need of major changes, and also be reused regardless the application. They are grouped by the system the task operates on, being *analytic* when representing a system that has not been fully characterized, and *synthetic* when the system does not exist and the goal is to construct its first description.

This work describes a system that is not completely known, implying the usage of an *analytical* template. This category comprises tasks such as classification, diagnosis, assessment, monitoring, and prediction. Since the functional annotation can be viewed as a problem which aims to find the protein "class" that best suits the sequence, it fits perfectly the *classification* task. This template, as well as all others, is composed of:

- *General characterization*: it defines the features of the task, presenting its *goals*, inputs, and outputs. It also specifies in terms of *class*, *attributes*, and *features* the *terminology* used to describe the object to be classified.
- *Default method*: this is related to the first decision to be carried out. It can be data-driven when it uses the initial features to create a set of possible solutions, or solution-driven, when the process starts with the complete solution set and it is reduced based on the information obtained. In this work, the default method is data-driven, since the features of the query sequence are compared to a database resulting in a subset of candidates.
- *Variation method:* it replaces the *default method* when this cannot be used to come up with a solution. In this work, there is no variation method.
- *Typical schema of the domain:* it is the schema of how the knowledge is structured and flows.

B.2. Activity Theory

The AT Leontiev (1978) is a philosophical and analytical framework from Social Sciences to analyze societies or individuals and their behavior and actions. This ap-

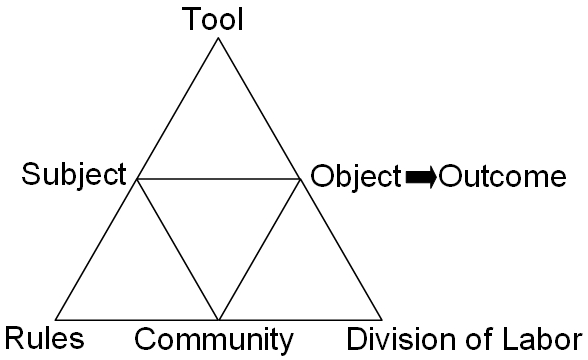


Figure B.1: Diagram of the activity system.

proach has been extensively applied in researches in Psychology (El'konin, D. B, 1977; Engeström, 1987; Ilyenkov, 1982; Leontiev, 1978; Luria, 1976; Zinchenko, 1985), Neuropsychology (Luria, 1979), Computer Supported Cooperative Work (CSCW) (Bødker and Grønbæk, 1996), Ergonomy and Human-Computer Interaction (Bednyi and Meister, 1997; Kaptelinin et al., 1999), and so on.

This theory adopts a multi-perspective approach where several dimensions appear simultaneously and interleaved. It focuses on the individual cognition and acting, but considering it to a large extent as the result and cause of its socio-historical and artifactual context. People interact with their context changing it, but also been changed in this interaction. These complex contextualized interactions are the *activities*. AT considers them the minimal meaningful unit of analysis of human action.

An *activity system* is the context that encompasses an activity and the network of elements interacting in it (see Fig. B.1). These elements are commonly referred as *artifacts*. Most of them can be both concrete (e.g., a gene sequence or a computer) and/or mental (e.g., a plan or experiences).

The activity system considers both the individual and social levels. The individual one focuses on the process a *subject* carries out to satisfy some of her/his needs. These needs are the *objectives*. The objectives are satisfied by the *outcomes* produced transforming *objects* using *tools*. The social level focuses on the *community*. This is the set of subjects related to the same object, either directly or indirectly (Kuutti, 1996). The relationships of the community with the subject are organized by *rules*, which determine how the later should behave in the former. The *division of labor* establishes the organization of the community in the activity process. Both *rules* and *division of labor* include similar types of artifacts, such as norms, social conventions, culture knowledge, or social structures.

Tools, rules, and division of labor are said to mediate relationships in the transformation process. They shape the forms of interaction. For instance, tools empower the subject with the community experience they crystallize, but at the same time constrain the transformation according to their features.

Activity systems can be interconnected through shared artifacts. For instance, the

outcome of a system (e.g., a database or a trained annotator) can become the input for another (e.g., in annotation, the database is a tool and the annotator a subject). In these cases, the activity focusing the analysis is called the *central activity*, and the others producing artifacts for it or using its outcomes are the *neighbor activities*.

AT Leontiev (1978) also considers the dynamics of social systems through *contradictions*. These are tensions among the artifacts in networks of activity systems, which promote evolution. There are four types of contradictions according to the artifacts they affect and their relationships (Engeström, 1987):

- *Primary contradictions* are tensions internal to an artifact or between artifacts playing the same role in an activity system. They are typically the result of trying to satisfy contradictory needs. An example is the conflict for annotation tools between offering accurate results or processing fast.
- *Secondary contradictions* correspond to inadequacies between artifacts playing different roles in an activity system. Potential causes of them are the integration of new artifacts in an established system or changes in the expected capabilities of the existing ones. For instance, the need of trained curator subjects to get high-quality resources is in tension with rules of limited funding.
- *Tertiary contradictions* are conflicts that emerge as a consequence of the evolution of activities. The old and new forms of the activity compete, for instance, to attract resources. An example of this was the movement from the initial sequencing activities to the recent ones with high throughput.
- *Quaternary contradictions* appear between a central activity system and its interconnected neighbors. An example of this could be a training activity that is not producing curators with the expected skills for the annotation activity.

AT literature offer a catalog of contradictions and potential solutions or mitigating actions for them. These come from the study of actual social situations, which can be extrapolated to other settings sharing their features.

B.3. INGENIAS

INGENIAS (Pavón et al., 2005) ia a well-defined methodology for the development of MASs. One of its key advantages in the context of annotation systems is that its process assumes the need of addressing evolution. Thus, it is able to drive system progress depending on agent technology, new tools and data sources, or experts' processes.

For MAS specification, INGENIAS proposes an organization based on five viewpoints: *organization*, *agent*, *goals/task*, *interactions*, and *environment*. These viewpoints are described using a specific modeling language. This in turn has been specified using a metamodel described with the GOPRR (Graph, Object, Property, Role, Relationship) metamodeling language (Lyytinen and Rossi, 1996) language. Next subsections explain briefly each viewpoint.

B.3.1. Organization viewpoint

The *organization* illustrates the framework where *agents*, *resources*, *tasks*, and *goals* appear. It can be defined based on three aspects: structure, functionality, and relationships. The first one decomposes the MAS in *groups* and *workflows*. The second one uses purpose (i.e., *goals* and *task* to define the functionalities. The third can be established at the level of *organizations*, *groups*, *agents*, or *roles*.

Groups can contain *agents*, *roles*, *resources*, and *applications*, while *workflows* describe the association between *tasks* and present general information regarding their execution. *Organizations* can group the other elements.

B.3.2. Agent viewpoint

The *agent* viewpoint takes into account the *goals* the *agent* pursues , the *tasks* it executes, and the *roles* it plays. This defines the *agent* behavior based on:

- Mental state: any information (e.g., *goals*, *believes*, *facts*, and *components*) that allows the *agent* to make a decision.
- Mental state manager: manages the *mental state*, acting (e.g., create, destroy, and modify) upon its elements and relationships.
- Mental state processor: decides which *tasks* to execute, determining how the *mental state* evolves.

B.3.3. Task/Goals viewpoint

The Task/Goals Viewpoint takes into consideration the decomposition of *goals* and *tasks*. It also determines the required elements and the expected outputs for that execution.

B.3.4. Interaction viewpoint

The *Interaction* viewpoint describes the exchange of information among agents or agents and users. An *interaction* comprises:

- Actors in the interaction: who starts the interaction (i.e., *initiator*) and who collaborates with it (i.e., *collaborator*). There is only one *initiator* and at least one *collaborator*.
- Interaction specification: specifies how the *interaction* is built. It mainly refers to the protocols that guide the *interaction*, but it can include mental attributes and actions triggered by messages.
- Context of the interaction: comprises the *goals* of the *interaction* and the *mental state* the participants have during all the stages of the process.

- Nature on the interaction: is related to the attitude the participant have (e.g., negotiate, cooperate, perform)

B.3.5. Environment viewpoint

The *environment* viewpoint defines the entities the MAS interacts with. These entities can be *resources* (e.g., CPU, file descriptors, or memory), other *agents*, or *applications*.

Bibliography

A room without books
is like a body without a soul.

Cicero

Akerkar, R. and Sajja, P. *Knowledge-Based Systems*. Jones & Bartlett Learning, 2010.

Alferes, J. J. and Pereira, L. M. *Reasoning with Logic Programming*, vol. 1111 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1996.

Alonso, A. and Guijarro, B. *Ingeniería del Conocimiento. Aspectos Metodológicos.*. Pearson Prentice Hall, 1st edition, 2004.

Alterovitz, G. and Ramoni, M., editors. *Knowledge-Based Bioinformatics: From analysis to interpretation*. Wiley, 2010.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, vol. 25(17), pages 3389–3402, 1997.

Andrade, M. A., Brown, N. P., Leroy, C., Hoersch, S., de Daruvar, A., Reich, C., Franchini, A., Tamames, J., Valencia, A., Ouzounis, C. and Sander, C. Automated genome sequence analysis and annotation. *Bioinformatics*, vol. 15(5), pages 391–412, 1999.

Aniba, M. R., Siguenza, S., Friedrich, A., Plewniak, F., Poch, O., Marchler-Bauer, A. and Thompson, J. D. Knowledge-based expert systems and a proof-of-concept case study for multiple sequence alignment construction and analysis. *Briefings in Bioinformatics*, vol. 10(1), pages 11–23, 2009.

Arias, E., Eden, H., Fischer, G., Gorman, A. and Scharff, E. Transcending the individual human mind – creating shared understanding through collaborative design. *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 7(1), pages 84–113, 2000. Special issue on Human-Computer Interaction in the New Millennium, Part 1.

ASHBURNER, M., BALL, C. A., BLAKE, J. A., BOTSTEIN, D., BUTLER, H., CHERRY, J. M., DAVIS, A. P., DOLINSKI, K., DWIGHT, S. S., EPPIG, J. T. ET AL. Gene Ontology: tool for the unification of biology. *Nature Genetics*, vol. 25(1), pages 25–29, 2000.

ATTWOOD, T. K., CRONING, M. D. R., FLOWER, D. R., LEWIS, A. P., MABEY, J. E., SCORDIS, P., SELLEY, J. N. and WRIGHT, W. PRINTS-S: the database formerly known as PRINTS. *Nucleic Acids Research*, vol. 28(1), pages 225–227, 2000.

AWAD, E. M. and GHAZIRI, H. M. *Knowledge Management*. Pearson Education India, 2004.

AZÉ, J., GENTILS, L., TOFFANO-NIOCHE, C., LOUX, V., GIBRAT, J.-F., BESSIÈRES, P., ROUVEIROL, C., POUPON, A. and FROIDEVAUX, C. Towards a semi-automatic functional annotation tool based on decision-tree techniques. *BMC Proceedings*, vol. 2(Suppl 4), page S3, 2008.

BAIROCH, A. The enzyme database in 2000. *Nucleic Acids Research*, vol. 28(1), pages 304–305, 2000.

BEDNYI, G. Z. and MEISTER, D. *The Russian Theory of Activity: Current Application to Design and Learning*. Lawrence Erlbaum Associates, 1997.

BELLIFEMINE, F., POGGI, A. and RIMASSA, G. JADE: a FIPA2000 compliant agent development environment. In *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS 2001)*, pages 216–217. ACM, 2001. ISBN 1-58113-326-X.

BENSON, D. A., CAVANAUGH, M., CLARK, K., KARSCH-MIZRACHI, I., LIPMAN, D. J., OSTELL, J. and SAYERS, E. W. GenBank. *Nucleic Acids Research*, vol. 41(D1), pages D36–D42, 2013.

BENSON, D. A., KARSCH-MIZRACHI, I., LIPMAN, D. J., OSTELL, J. and SAYERS, E. W. GenBank. *Nucleic Acids Research*, vol. 39(Database-Issue), pages 32–37, 2011.

BERGER, B. and LEIGHTON, T. Protein folding in the hydrophobic-hydrophilic (HP) is NP-complete. In *2nd Annual International Conference on Computational Molecular Biology*, RECOMB '98, pages 30–39. ACM, New York, NY, USA, 1998. ISBN 0-89791-976-9.

BERNSTEIN, F. C., KOETZLE, T. F., WILLIAMS, G. J., MEYER, E. F., BRICE, M. D., RODGERS, J. R., KENNARD, O., SHIMANOUCHI, T. and TASUMI, M. The Protein Data Bank. A computer-based archival file for macromolecular structures. *European Journal of Biochemistry*, vol. 80(2), pages 319–324, 1977a.

BERNSTEIN, F. C., KOETZLE, T. F., WILLIAMS, G. J., MEYER, E. F., BRICE, M. D., RODGERS, J. R., KENNARD, O., SHIMANOUCHI, T. and TASUMI, M. The Protein Data Bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, vol. 112(3), pages 535–542, 1977b.

BODENREIDER, O. and STEVENS, R. Bio-ontologies: current trends and future directions. *Briefings in Bioinformatics*, vol. 7(3), pages 256–274, 2006.

BØDKER, S. and GRØNBÆK, K. Users and Designers in Mutual Activity: An Analysis of Cooperative Activities in System Design. In *Cognition and Communication at Work* (edited by Y. Engeström and D. Middleton), pages 130–158. Cambridge, England: Cambridge University Press, 1996.

BOGUSKI, M. Bioinformatics – a new era. In *Trends Guide to Bioinformatics*, vol. 19, pages 1–3. Elsevier Science, 1998.

BRACHMAN, R. J. and LEVESQUE, H. J. *Knowledge representation and reasoning*. Morgan Kaufmann, 2004.

BRYSON, K., LUCK, M., JOY, M. and JONE, D. T. Agent Interaction for Bioinformatics Data Management. *Applied Artificial Intelligence*, vol. 15, pages 917–947, 2001.

BUCHAN, D. W. A., MINNECI, F., NUGENT, T. C. O., BRYSON, K. and JONES, D. T. Scalable web services for the PSIPRED Protein Analysis Workbench. *Nucleic Acids Research*, vol. 41(W1), pages W349–W357, 2013.

BURGE, C. and KARLIN, S. Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology*, vol. 268(1), pages 78 – 94, 1997. ISSN 0022-2836.

BURTON, A., SHADBOLT, N., RUGG, G. and HEDGECOCK, A. The efficacy of knowledge elicitation techniques: a comparison across domains and levels of expertise. *Knowledge Acquisition*, vol. 2(2), pages 167–178, 1990.

CADAG, E., LOUIE, B., MYLER, P. J. and TARCZY-HORNOCH, P. Biomediator data integration and inference for functional annotation of anonymous sequences. In *Pacific Symposium on Biocomputing* (edited by R. B. Altman, A. K. Dunker, L. Hunter, T. Murray and T. E. Klein), pages 343–354. World Scientific Publishing Co., 2007. ISBN 981-270-417-5.

CAMON, E., MAGRANE, M., BARRELL, D., LEE, V., DIMMER, E., MASLEN, J., BINNS, D., HARTE, N., LOPEZ, R. and APWEILER, R. The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Research*, vol. 32(suppl 1), pages D262–D266, 2004.

CHEN, T.-W., GAN, R.-C. R., WU, T. H., HUANG, P.-J., LEE, C.-Y., CHEN, Y.-Y. M., CHEN, C.-C. and TANG, P. FastAnnotator - an efficient transcript annotation web tool. *BMC Genomics*, vol. 13(Suppl 7), page S9, 2012.

CHOREV, M. and CARMEL, L. The function of introns. *Frontiers in genetics*, vol. 3, 2012.

CLAUDEL-RENARD, C., CHEVALET, C., FARAUT, T. and KAHN, D. Enzyme-specific profiles for genome annotation: PRIAM. *Nucleic Acids Research*, vol. 31(22), pages 6633–6639, 2003.

CLAVERIE, J. M. and NOTREDAME, C. *Bioinformatics for Dummies*. Wiley, 2nd edition, 2006.

COCK, P. J. A., ANTAO, T., CHANG, J. T., CHAPMAN, B. A., COX, C. J., DALKE, A., FRIEDBERG, I., HAMELRYCK, T., KAUFF, F., WILCZYNSKI, B. and DE HOON, M. J. L. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, vol. 25(11), pages 1422–1423, 2009.

CONESA, A., GÖTZ, S., GARCÃA-GÓMEZ, J. M., TEROL, J., TALÓN, M. and ROBLES, M. Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research. *Bioinformatics*, vol. 21(18), pages 3674–3676, 2005.

CONSORTIUM, T. U. UniProt: a hub for protein information. *Nucleic Acids Research*, vol. 43(D1), pages D204–D212, 2015.

COOPER, G. M. *The Cell - A Molecular Approach 2nd Edition*. Sunderland (MA): Sinauer Associates, 2000.

COZZETTO, D., BUCHAN, D. W. A., BRYSON, K. and JONES, D. T. Protein function prediction by massive integration of evolutionary analyses and multiple data sources. *BMC Bioinformatics*, vol. 14(S-3), page S1, 2013.

CRESCENZI, P., GOLDMAN, D., PAPADIMITRIOU, C., PICCOLBONI, A. and YANNAKAKIS, M. On the complexity of protein folding. *Journal of Computational Biology*, vol. 5(3), pages 423–465, 1998a.

CRESCENZI, P., GOLDMAN, D., PAPADIMITRIOU, C. H., PICCOLBONI, A. and YANNAKAKIS, M. On the complexity of protein folding. In *2nd Annual International Conference on Computational Molecular Biology*, RECOMB '98, pages 61–62. ACM, New York, NY, USA, 1998b.

CRG - COUNCIL FOR RESPONSIBLE GENETICS. Genetic privacy. `http://www.councilforresponsiblegenetics.org/geneticprivacy`, 2015. [Online: accessed 5-September-2015].

CRUZ, J. PredictSPTM. Unpublished data.

CURWEN, V., EYRAS, E., ANDREWS, T. D., CLARKE, L., MONGIN, E., SEARLE, S. M. and CLAMP, M. The Ensembl automatic gene annotation system. *Genome Res*, vol. 14(5), pages 942–950, 2004.

DAINAT, J., PAGANINI, J., PONTAROTTI, P. and GOURET, P. GLADX: An Automated Approach to Analyze the Lineage-Specific Loss and Pseudogenization of Genes. *PLoS ONE*, vol. 7(6), page e38792, 2012.

DECKER, K., KHAN, S., SCHMIDT, C., SITU, G., MAKKENA, R. and MICHAUD, D. BioMAS: a Multi-Agent System for Genomic Annotation. *International Journal of Cooperative Information Systems*, vol. 11(03n04), pages 265–292, 2002.

DECKER, K., PANNU, A., SYCARA, K. and WILLIAMSON, M. Designing behaviors for information agents. In *Proceedings of the First International Conference on Autonomous Agents*, AGENTS '97, pages 404–413. ACM, New York, NY, USA, 1997. ISBN 0-89791-877-0.

DECKER, K. S. and LESSER, V. R. Quantitative Modeling of Complex Computational Task Environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224. 1993.

DECKER, K. S. and SYCARA, K. Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, vol. 9(3), pages 239–260, 1997. ISSN 09259902.

DOMSELAAR, G. H. V., STOTHARD, P., SHRIVASTAVA, S., CRUZ, J. A., GUO, A., DONG, X., LU, P., SZAFRON, D., GREINER, R. and WISHART, D. S. BASys: a web server for automated bacterial genome annotation. *Nucleic Acids Research*, vol. 33(2), pages W455–W459, 2005.

DURBIN, R., HAUSSLER, D., STEIN, L., LEWIS, S. and KROG, A. GFF (general feature format) specifications document. `http://www.sanger.ac.uk/resources/software/gff/spec.html`, 2011. [Online: accessed 5-September-2015].

EDDY, S. R. Profile hidden markov models. *Bioinformatics*, vol. 14, pages 755–763, 1998. `http://hmmer.org`.

EDDY, S. R., WHEELER, T. J. and THE HMMER DEVELOPMENT TEAM. HMMER User's Guide: Biological sequence analysis using profile hidden Markov models. `ftp://selab.janelia.org/pub/software/hmmer3/3.1b2/Userguide.pdf`, 2015. [Online: accessed 5-September-2015].

EDGAR, R. C. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, vol. 32(5), pages 1792–1797, 2004.

EDWARDS, D., STAJICH, J. E. and HANSEN, D. *Bioinformatics tools and applications*. Springer, 2009. ISBN 9780387927381.

EL'KONIN, D. B. Toward the problem of stages in the mental development of the child. In *Soviet developmental psychology* (edited by M. Cole), pages 538–563. White Plains, N.Y.: Sharpe, 1977.

ELSIK, C. G., WORLEY, K. C., ZHANG, L., MILSHINA, N. V., JIANG, H., REESE, J. T., CHILDS, K. L., VENKATRAMAN, A., DICKENS, C. M., WEINSTOCK, G. M. and GIBBS, R. A. Community annotation: procedures, protocols, and supporting tools. *Genome Research*, vol. 16(11), pages 1329–1333, 2006.

ENGESTRÖM, Y. *Learning by expanding: An activity-theoretical approach to developmental research*. Orienta-Konsultit, Helsinki, Finland, 1987.

ENSEMBL GENOME BOWSER. GTF/GTF File Format. http://useast.ensembl.org/info/website/upload/gff.html, 2015. [Online: accessed 5-September-2015.].

FALDA, M., TOPPO, S., PESCAROLO, A., LAVEZZO, E., DI CAMILLO, B., FACCHINETTI, A., CILIA, E., VELASCO, R. and FONTANA, P. Argot2: a large scale function prediction tool relying on semantic similarity of weighted gene ontology terms. *BMC bioinformatics*, vol. 13 Suppl 4, 2012.

FARLEY, J. Java distributed computing. O'Reilly, 1998.

FELSENSTEIN, J. Phylip (phylogeny inference package) version 3.7a. *Distributed by the author*, 2009. Department of Genome Sciences, University of Washington, Seattle.

FINN, R., MISTRY, J., TATE, J., COGGILL, P., HEGER, A., POLLINGTON, J., GAVIN, O., GUNESEKARAN, P., CERIC, G., FORSLUND, K., HOLM, L., SONNHAMMER, E., EDDY, S. and BATEMAN, A. The Pfam protein families database. *Nucleic Acids Res*, vol. 38(Database issue), pages D211–222, 2010. http://pfam.sanger.ac.uk.

FINN, R. D., BATEMAN, A., CLEMENTS, J., COGGILL, P., EBERHARDT, R. Y., EDDY, S. R., HEGER, A., HETHERINGTON, K., HOLM, L., MISTRY, J., SONNHAMMER, E. L. L., TATE, J. and PUNTA, M. Pfam: the protein families database. *Nucleic Acids Research*, vol. 42(D1), pages D222–D230, 2014.

FLICEK, P., AMODE, M. R., BARRELL, D., BEAL, K., BILLIS, K., BRENT, S., CARVALHO-SILVA, D., CLAPHAM, P., COATES, G., FITZGERALD, S., GIL, L., GIRÓN, C. G., GORDON, L., HOURLIER, T., HUNT, S., JOHNSON, N., JUETTEMANN, T., KÄHÄRI, A. K., KEENAN, S., KULESHA, E., MARTIN, F. J., MAUREL, T., MCLAREN, W. M., MURPHY, D. N., NAG, R., OVERDUIN, B., PIGNATELLI, M., PRITCHARD, B., PRITCHARD, E., RIAT, H. S., RUFFIER, M., SHEPPARD, D., TAYLOR, K., THORMANN, A., TREVANION, S. J., VULLO, A., WILDER, S. P., WILSON, M., ZADISSA, A., AKEN, B. L., BIRNEY, E., CUNNINGHAM, F., HARROW, J., HERRERO, J., HUBBARD, T. J. P., KINSELLA, R., MUFFATO, M., PARKER, A., SPUDICH, G., YATES, A., ZERBINO, D. R. and SEARLE, S. M. J. Ensembl 2014. *Nucleic Acids Research*, vol. 42(D1), pages D749–D755, 2014.

FLICEK, P., AMODE, M. R., BARRELL, D., BEAL, K., BRENT, S., CHEN, Y., CLAPHAM, P., COATES, G., FAIRLEY, S., FITZGERALD, S., GORDON, L., HENDRIX, M., HOURLIER, T., JOHNSON, N., KÄHÄRI, A., KEEFE, D., KEENAN, S., KINSELLA, R., KOKOCINSKI, F., KULESHA, E., LARSSON, P., LONGDEN, I., MCLAREN, W., OVERDUIN, B., PRITCHARD, B., RIAT, H. S., RIOS, D., RITCHIE, G. R. S., RUFFIER, M., SCHUSTER, M., SOBRAL, D., SPUDICH, G., TANG, Y. A., TREVANION, S., VANDROVCOVA, J., VILELLA, A. J., WHITE, S., WILDER, S. P., ZADISSA, A., ZAMORA, J., AKEN, B. L., BIRNEY, E., CUNNINGHAM, F., DUNHAM, I., DURBIN, R., FERNÁNDEZ-SUAREZ, X. M., HERRERO, J., HUBBARD, T. J. P., PARKER, A.,

PROCTOR, G., VOGEL, J. and SEARLE, S. M. J. Ensembl 2011. *Nucleic Acids Research*, vol. 39(suppl 1), pages D800–D806, 2011. ISSN 1362-4962.

FRIEDBERG, I. Automated protein function prediction - the genomic challenge. *Briefings in Bioinformatics*, vol. 7(3), pages 225–242, 2006.

GAMMA, E., HELM, R., JOHNSON, R. E. and VLISSIDES, J. M. Design Patterns: Abstraction and Reuse of Object-Oriented Design. In *Proceedings of the 7th European Conference on Object-Oriented Programming*, ECOOP '93. Springer-Verlag, London, UK, UK, 1993.

GARCÍA-MAGARIÑO, I., GÓMEZ-SANZ, J. J. and PÉREZ-AGÜERA, J. R. A Complete-Computerised Delphi Process with a Multi-Agent System. In *Programming Multi-Agent Systems* (edited by K. V. Hindriks, A. Pokahr and S. Sardina), vol. 5442 of *Lecture Notes in Computer Science*, pages 120–135. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03277-6.

GENERIC MODEL ORGANISM DATABASE (GMOD) PROJECT. GBrowse - GMOD. `http://gmod.org/wiki/GBrowse`, 2013. [Online: accessed 5-September-2015].

GIARRATANO, J. C. and RILEY, G. *Expert Systems: Principles and Programming*. Computer Science Series. PWS Publishing Company, 1998. ISBN 9780534950538.

GOECKS, J., NEKRUTENKO, A., TAYLOR, J. and TEAM, T. G. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, vol. 11(8), page R86, 2010. ISSN 1465-6906.

GÖTZ, S., ARNOLD, R., SEBASTIÃ¡N-LEÓN, P., MARTÍN-RODRÍGUEZ, S., TISCHLER, P., JEHL, M.-A., DOPAZO, J., RATTEI, T. and CONESA, A. B2G-FAR, a species-centered GO annotation repository. *Bioinformatics*, vol. 27(7), pages 919–924, 2011.

GOURET, P., VITIELLO, V., BALANDRAUD, N., GILLES, A., PONTAROTTI, P. and DANCHIN, E. G. FIGENIX: Intelligent automation of genomic annotation: expertise integration in a new software platform. *BMC Bioinformatics*, vol. 6, page 198, 2005.

GRAHAM, J., DECKER, K. and MERSIC, M. DECAF - A Flexible Multi Agent System Architecture. *Autonomous Agents and Multi-Agent Systems*, vol. 7(1-2), pages 7–27, 2003. ISSN 1387-2532.

GUSTIN, B. H. Charisma, recognition, and the motivation of scientists. *American Journal of Sociology*, vol. 78(5), pages 1119–1134, 1973.

HAFT, D. H., SELENGUT, J. D., RICHTER, R. A., HARKINS, D., BASU, M. K. and BECK, E. TIGRFAMs and Genome Properties in 2013. *Nucleic Acids Research*, vol. 41(D1), pages D387–D395, 2013.

HAFT, D. H., SELENGUT, J. D. and WHITE, O. The TIGRFAMs database of protein families. *Nucleic Acids Research*, vol. 31(1), pages 371–373, 2003.

HAYES-ROTH, F. Rule-based systems. *Communications of the ACM*, vol. 28(9), pages 921–932, 1985.

HOLLAND, R. C. G., DOWN, T. A., POCOCK, M., PRLIC, A., HUEN, D., JAMES, K., FOISY, S., DRAGER, A., YATES, A., HEUER, M. and SCHREIBER, M. J. BioJava: an open-source framework for bioinformatics. *Bioinformatics*, vol. 24(18), pages 2096–2097, 2008.

HUNTER, S., JONES, P., MITCHELL, A., APWEILER, R., ATTWOOD, T. K., BATEMAN, A., BERNARD, T., BINNS, D., BORK, P., BURGE, S., DE CASTRO, E., COGGILL, P., CORBETT, M., DAS, U., DAUGHERTY, L., DUQUENNE, L., FINN, R. D., FRASER, M., GOUGH, J., HAFT, D., HULO, N., KAHN, D., KELLY, E., LETUNIC, I., LONSDALE, D., LOPEZ, R., MADERA, M., MASLEN, J., MCANULLA, C., MCDOWALL, J., MCMENAMIN, C., MI, H., MUTOWO-MUELLENET, P., MULDER, N., NATALE, D., ORENGO, C., PESSEAT, S., PUNTA, M., QUINN, A. F., RIVOIRE, C., SANGRADOR-VEGAS, A., SELENGUT, J. D., SIGRIST, C. J. A., SCHEREMETJEW, M., TATE, J., THIMMAJANARTHANAN, M., THOMAS, P. D., WU, C. H., YEATS, C. and YONG, S.-Y. InterPro in 2011: new developments in the family and domain prediction database. *Nucleic Acids Research*, vol. 40(D1), pages D306–D312, 2012.

IGLESIAS, C. A., GARIJO, M., GONZÁLEZ, J. C. and VELASCO, J. R. MAS-CommonKADS: A Comprehensive Agent-Oriented Methodology. In *Proceedings of the 11th Int. Conference on Mathematical and Computer Modelling and Scientific Computing*. 1997. (An extended version of this paper has been published in the journal Mathematical Modelling and Scientific Computing, Vol.8, 1997).

ILYENKOV, E. V. *The dialectics of the abstract and the concrete in Marx's Capital*. Moscow: Progress, 1982.

INTERPRO. Interproscan5outputformats. `https://code.google.com/p/interproscan/wiki/InterProScan5OutputFormats`, 2013. [Online: accessed 5-September-2015].

JONES, C. E., BROWN, A. L. and BAUMANN, U. Estimating the annotation error rate of curated go database sequence annotations. *BMC Bioinformatics*, vol. 8(1), page 170, 2007.

JONES, D. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. *J Mol Biol*, vol. 287, pages 797–815, 1999.

JONES, D., TAYLOR, W. and THORNTON, J. A new approach to protein fold recognition. *Nature*, vol. 358, pages 86–9, 1992.

JONES, D., TAYLOR, W. and THORNTON, J. A model recognition approach to the prediction of all-helical membrane protein structure and topology. *Biochemistry*, vol. 33, pages 3038–3049, 1994.

JONES, P., BINNS, D., CHANG, H.-Y., FRASER, M., LI, W., MCANULLA, C., MCWILLIAM, H., MASLEN, J., MITCHELL, A., NUKA, G., PESSEAT, S., QUINN, A. F., SANGRADOR-VEGAS, A., SCHEREMETJEW, M., YONG, S.-Y., LOPEZ, R. and HUNTER, S. InterProScan 5: genome-scale protein function classification. *Bioinformatics*, vol. 30(9), pages 1236–1240, 2014.

KAHN, D., REZVOY, C. and VIVIEN, F. Parallel Large Scale Inference of Protein Domain Families. In *ICPADS'08, the 14th IEEE International Conference on Parallel and Distributed Systems*, pages 72–79. IEEE Computer Society Press, 2008.

KANEHISA, M. and GOTO, S. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, vol. 28(1), pages 27–30, 2000.

KAPTELININ, V., NARDI, B. A. and MACAULAY, C. The Activity Checklist: A tool for representing the "space" of context. *Interactions*, vol. 6(4), pages 27–39, 1999.

KARLIN, S. and ALTSCHUL, S. F. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 87(6), pages 2264–2268, 1990.

KEEDWELL, E. and NARAYANAN, A. *Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems*. Wiley, 2005.

KLIMKE, W., AGARWALA, R., BADRETDIN, A., CHETVERNIN, S., CIUFO, S., FEDOROV, B., KIRYUTIN, B., OÑEILL, K., RESCH, W., RESENCHUK, S., SCHAFER, S., TOLSTOY, I. and TATUSOVA, T. The National Center for Biotechnology Information's Protein Clusters Database. *Nucleic Acids Research*, vol. 37(1), pages D216–D223, 2009.

KOSKI, L. B., GRAY, M. W., LANG, B. F. and BURGER, G. AutoFACT: An Automatic Functional Annotation and Classification Tool. *BMC Bioinformatics*, vol. 6, page 151, 2005.

KOSKINEN, P., TÖRÖNEN, P., NOKSO-KOIVISTO, J. and HOLM, L. PANNZER: high-throughput functional annotation of uncharacterized proteins in an error-prone environment. *Bioinformatics*, vol. 31(10), pages 1544–1552, 2015.

KROGH, A., BROWN, M., MIAN, I., SJÖLANDER, K. and HAUSSLER, D. Hidden Markov Models in Computational Biology: Applications to Protein Modeling . *Journal of Molecular Biology*, vol. 235(5), pages 1501 – 1531, 1994. ISSN 0022-2836.

KROGH, A., LARSSON, B., VON HEIJNE, G. and SONNHAMMER, E. L. L. Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *Journal of Molecular Biology*, vol. 305(3), pages 567–580, 2001. ISSN 00222836.

KUMAR, P., HENIKOFF, S. and NG, P. C. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nature Protocols*, vol. 4(7), pages 1073–1081, 2009. ISSN 1750-2799.

KUUTTI, K. Activity theory as a potential framework for human-computer interaction research. In *Context and Consciousness: Activity Theory and Human-Computer Interaction* (edited by B. A. Nardi), pages 17–44. MIT Press, Cambridge, MA, USA, 1996.

LARKIN, M., BLACKSHIELDS, G., BROWN, N., CHENNA, R., MCGETTIGAN, P., MCWILLIAM, H., VALENTIN, F., WALLACE, I., WILM, A., LOPEZ, R., THOMPSON, J., GIBSON, T. and HIGGINS, D. Clustal W and Clustal X version 2.0. *Bioinformatics*, vol. 23(21), pages 2947–2948, 2007.

LEE, D., REDFERN, O. and ORENGO, C. Predicting protein function from sequence and structure. *Nature Reviews Molecular Cell Biology*, vol. 8(12), pages 995–1005, 2007.

LEHNINGER, A. L., NELSON, D. L. and COX, M. M. *Lehninger Principles of Biochemistry*. W. H. Freeman & Company, 5th edition, 2008.

LEONTIEV, A. N. *Activity, Consciousness, and Personality*. Prentice-Hall, Hillsdale, NJ, USA, 1978.

LETUNIC, I., DOERKS, T. and BORK, P. SMART 7: recent updates to the protein domain annotation resource. *Nucleic Acids Research*, vol. 40(D1), pages D302–D305, 2012a.

LETUNIC, I., DOERKS, T. and BORK, P. SMART 7: recent updates to the protein domain annotation resource. *Nucleic Acids Research*, vol. 40(D1), pages D302–D305, 2012b.

LI, H. and HOMER, N. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, vol. 11(5), pages 473–483, 2010.

LOBLEY, A. E. *Human protein function prediction: application of machine learning for integration of heterogeneous data sources*. Doctoral Thesis, University College London, 2010.

LUCK, M. and MERELLI, E. Agents in Bioinformatics. *The Knowledge Engineering Review*, vol. 20(2), pages 117–125, 2005.

LUPAS, A., VAN DYKE, M. and STOCK, J. Predicting coiled coils from protein sequences. *Science*, vol. 252(5009), pages 1162–1164, 1991.

LURIA, A. R. *Cognitive development: Its cultural and social foundations*. Cambridge, Mass.: Harvard University Press, 1976.

LURIA, A. R. *The Making of Mind: A Personal Account of Soviet Psychology*. Harvard University Press, 1979.

LYYTINEN, K. S. and ROSSI, M. METAEDIT+: A fully configurable multiuser and multi-tool CASE and CAME Environment. In *Proceedings of the 8th International Conference on Advances in Information System Engineering* (edited by J. Mylopoulos and Y. Vassiliou), 1000, pages 1–21. Engineering. Vol. 1080 (pp. 1-21)., 1996.

MAGRANE, M. and CONSORTIUM, U. UniProt Knowledgebase: a hub of integrated protein data. *Database*, vol. 2011, 2011.

MAGRANE, M. and UNIPROT CONSORTIUM. UniProt knowledgebase: a hub of integrated protein data. *Database*, vol. 2011, page bar009, 2011. ISSN 1758-0463.

MARCHLER-BAUER, A. and BRYANT, S. H. CD-Search: protein domain annotations on the fly. *Nucleic Acids Research*, vol. 32(suppl 2), pages W327–W331, 2004.

MARCHLER-BAUER, A., LU, S., ANDERSON, J. B., CHITSAZ, F., DERBYSHIRE, M. K., DEWEESE-SCOTT, C., FONG, J. H., GEER, L. Y., GEER, R. C., GONZALES, N. R., GWADZ, M., HURWITZ, D. I., JACKSON, J. D., KE, Z., LANCZYCKI, C. J., LU, F., MARCHLER, G. H., MULLOKANDOV, M., OMELCHENKO, M. V., ROBERTSON, C. L., SONG, J. S., THANKI, N., YAMASHITA, R. A., ZHANG, D., ZHANG, N., ZHENG, C. and BRYANT, S. H. CDD: conserved domains and protein three-dimensional structure. *Nucleic Acids Research*, vol. 41(D1), pages D348–D352, 2013.

MARCHLER-BAUER, A., LU, S., ANDERSON, J. B., CHITSAZ, F., DERBYSHIRE, M. K., DEWEESE-SCOTT, C., FONG, J. H., GEER, L. Y., GEER, R. C., GONZALES, N. R., GWADZ, M., HURWITZ, D. I., JACKSON, J. D., KE, Z., LANCZYCKI, C. J., LU, F., MARCHLER, G. H., MULLOKANDOV, M., OMELCHENKO, M. V., ROBERTSON, C. L., SONG, J. S., THANKI, N., YAMASHITA, R. A., ZHANG, D., ZHANG, N., ZHENG, C. and BRYANT, S. H. CDD: a Conserved Domain Database for the functional annotation of proteins. *Nucleic Acids Research*, vol. 39(1), pages D225–D229, 2011.

MARCHLER-BAUER, A., PANCHENKO, A. R., SHOEMAKER, B. A., THIESSEN, P. A., GEER, L. Y. and BRYANT, S. H. CDD: a database of conserved domain alignments with links to domain three-dimensional structure. *Nucleic Acids Research*, vol. 30(1), pages 281–283, 2002.

MAYFIELD, J., LABROU, Y. and FININ, T. Evaluation of KQML as an agent communication language. In *Intelligent Agents: Theories, Architectures, and Languages*, vol. 1037 *of LNAI*, pages 347–360. 1996.

MAZUMDER, R., NATALE, D. A., JULIO, J. A. E., YEH, L.-S. and WU, C. H. Community annotation in biology. *Biology Direct*, vol. 5, page 12, 2009.

MCCALLUM, A. and NIGAM, K. A comparison of event models for Naive Bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. 1998.

MERELLI, E., ARMANO, G., CANNATA, N., CORRADINI, F., DÍNVERNO, M., DOMS, A., LORD, P., MARTIN, A., MILANESI, L., MÖLLER, S., SCHROEDER, M. and LUCK, M. Agents in Bioinformatics, Computational and Systems Biology. *Briefings in Bioinformatics*, vol. 8(1), pages 45–59, 2007.

MI, H., MURUGANUJAN, A. and THOMAS, P. D. PANTHER in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees. *Nucleic Acids Research*, vol. 41(Database-Issue), pages 377–386, 2013.

MIAO, Y., HOLST, S., HOLMER, T., FLESCHUTZ, J. M. and ZENTEL, P. An activity-oriented approach to visually structured knowledge representation for problem-based learning in virtual learning environments. In *Designing Cooperative Systems – The Use of Theories and Models, Proceedings of the 4th International Conference on the Design of Cooperative Systems (COOP 2000)* (edited by R. Dieng, A. Giboin, L. Karsenty and G. De Michelis), pages 303–320. IOS Press, 2000.

MINNECI, F., PIOVESAN, D., COZZETTO, D. and JONES1, D. T. FFPred 2.0: Improved Homology-Independent Prediction of Gene Ontology Terms for Eukaryotic Protein Sequences. *PLoS One*, vol. 8(5), 2013.

MÖLLER, S., LESER, U., FLEISCHMANN, W. and APWEILER, R. EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. *Bioinformatics*, vol. 15(3), pages 219–227, 1999.

MÖLLER, S., SCHROEDER, M. and APWEILER, R. Consistent integration of non-reliable heterogeneous information resources applied to the annotation of transmembrane proteins. *Computers & Chemistry*, vol. 26(1), pages 41–49, 2001.

NAKAI, K. and HORTON, P. PSORT: a program for detecting sorting signals in proteins and predicting their. *Trends in Biochemical Sciences*, 1999.

NCBI. Web BLAST page options. `http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml`, 2007. [Online; accessed 5-September-2015].

NCBI - NATIOANAL CENTER FOR BIOTECHNOLOGY INFORMATION. `http://www.ncbi.nlm.nih.gov`, 2013. [Online: accessed 5-September-2015].

NCBI - NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION. Web BLAST page options. `http://www.ncbi.nlm.nih.gov/BLAST/blastcgihelp.shtml`, 2007. [Online: accessed 5-September-2015].

NCBI - NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION. NCBI Conserved Domain Database (CDD) Help. `http://www.ncbi.nlm.nih.gov/Structure/cdd/cdd_help.shtml`, 2013. [Online: accessed 5-September-2015].

NCBI - NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION. Blast Frequently Asked Questions. `http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&`

PAGE_TYPE=BlastDocs&DOC_TYPE=FAQ, 2015. [Online: accessed 5-September-2015].

NCBI RESOURCE COORDINATORS. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, vol. 41(D1), pages D8–D20, 2013.

NOTREDAME, C., HIGGINS, D. and HERINGA, J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, vol. 302(1), pages 205–217, 2000.

PAJARES, G. and SANTOS, M. *Inteligencia Artíficial e Ingeniería del Conocimiento*. Ra-Ma, Librería y Editorial Microinformática, 2005.

PAVÓN, J., GÓMEZ-SANZ, J. J. and FUENTES, R. The INGENIAS methodology and tools. In *Agent-Oriented Methodologies* (edited by B. Henderson-Sellers and P. Giorgini), pages 236–276. Idea Group Publishing, 2005.

PEARSON, W. R. and LIPMAN, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 85(8), pages 2444–2448, 1988. ISSN 0027-8424. http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml.

PETERSEN, T. N., BRUNAK, S., VON HEIJNE, G. and NIELSEN, H. SignalP 4.0: discriminating signal peptides from transmembrane regions. *Nature Methods*, vol. 8(10), pages 785–786, 2011.

PEVSNER, J. *Bioinformatics and functional genomics*. Wiley-Blackwell, 2009. ISBN 0470085851 (cloth).

PONTIUS, J. U., WAGNER, L. and SCHULER, G. D. The BLAST Sequence Analysis Tool. In *The NCBI Handbook* (edited by Bethesda (MD): National Center for Biotechnology Information (NCBI)). National Center for Biotechnology Information, 2003.

PORTELA, C., VASCONCELOS, A., SILVA, A., SINIMBÚ, A., SILVA, E., RONNY, M., LIRA, W. and OLIVEIRA, S. A comparative analysis between bpmn and spem modeling standards in the software processes contextn. *Journal of Software Engineering and Applications*, vol. 5(5), page 19428, 2012.

POTTER, S. C., CLARKE, L., CURWEN, V., KEENAN, S., MONGIN, E., SEARLE, S. M. J., STABENAU, A., STOREY, R. and CLAMP, M. The Ensembl analysis pipeline. *Genome Research*, vol. 14(5), pages 934–941, 2004.

POWELL, S., FORSLUND, K., SZKLARCZYK, D., TRACHANA, K., ROTH, A., HUERTA-CEPAS, J., GABALDÓN, T., RATTEI, T., CREEVEY, C., KUHN, M., JENSEN, L. J., VON MERING, C. and BORK, P. eggNOG v4.0: nested orthology inference across 3686 organisms. *Nucleic Acids Research*, vol. 42(D1), pages D231–D239, 2014.

PRUITT, K. D., BROWN, G. R., HIATT, S. M., THIBAUD-NISSEN, F., ASTASHYN, A., ERMOLAEVA, O., FARRELL, C. M., HART, J., LANDRUM, M. J., MCGARVEY, K. M., MURPHY, M. R., O'LEARY, N. A., PUJAR, S., RAJPUT, B., RANGWALA, S. H., RIDDICK, L. D., SHKEDA, A., SUN, H., TAMEZ, P., TULLY, R. E., WALLIN, C., WEBB, D., WEBER, J., WU, W., DICUCCIO, M., KITTS, P., MAGLOTT, D. R., MURPHY, T. D. and OSTELL, J. M. RefSeq: an update on mammalian reference sequences. *Nucleic Acids Res.*, vol. 42, pages D756–763, 2014.

PUNTA, M., COGGILL, P. C., EBERHARDT, R. Y., MISTRY, J., TATE, J., BOURSNELL, C., PANG, N., FORSLUND, K., CERIC, G., CLEMENTS, J., HEGER, A., HOLM, L., SONNHAMMER, E. L. L., EDDY, S. R., BATEMAN, A. and FINN, R. D. The Pfam protein families database. *Nucleic Acids Research*, vol. 40(D1), pages D290–D301, 2012.

QUEVILLON, E., SILVENTOINEN, V., PILLAI, S., HARTE, N., MULDER, N., APWEILER, R. and LOPEZ, R. InterProScan: protein domains identifier. *Nucleic Acids Research*, vol. 33(2), pages W116–W120, 2005.

R CORE TEAM. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

RADIVOJAC, P., CLARK, W. T., ORON, T. R., SCHNOES, A. M., WITTKOP, T., SOKOLOV, A., GRAIM, K., FUNK, C., VERSPOOR, K., BEN-HUR, A., PANDEY, G., YUNES, J. M., TALWALKAR, A. S., REPO, S., SOUZA, M. L., PIOVESAN, D., CASADIO, R., WANG, Z., CHENG, J., FANG, H., GOUGH, J., KOSKINEN, P., TORONEN, P., NOKSO-KOIVISTO, J., HOLM, L., COZZETTO, D., BUCHAN, D. W. A., BRYSON, K., JONES, D. T., LIMAYE, B., INAMDAR, H., DATTA, A., MANJARI, S. K., JOSHI, R., CHITALE, M., KIHARA, D., LISEWSKI, A. M., ERDIN, S., VENNER, E., LICHTARGE, O., RENTZSCH, R., YANG, H., ROMERO, A. E., BHAT, P., PACCANARO, A., HAMP, T., KASZNER, R., SEEMAYER, S., VICEDO, E., SCHAEFER, C., ACHTEN, D., AUER, F., BOEHM, A., BRAUN, T., HECHT, M., HERON, M., HONIGSCHMID, P., HOPF, T. A., KAUFMANN, S., KIENING, M., KROMPASS, D., LANDERER, C., MAHLICH, Y., ROOS, M., BJORNE, J., SALAKOSKI, T., WONG, A., SHATKAY, H., GATZMANN, F., SOMMER, I., WASS, M. N., STERNBERG, M. J. E., SKUNCA, N., SUPEK, F., BOSNJAK, M., PANOV, P., DZEROSKI, S., SMUC, T., KOURMPETIS, Y. A. I., VAN DIJK, A. D. J., BRAAK, C. J. F. T., ZHOU, Y., GONG, Q., DONG, X., TIAN, W., FALDA, M., FONTANA, P., LAVEZZO, E., DI CAMILLO, B., TOPPO, S., LAN, L., DJURIC, N., GUO, Y., VUCETIC, S., BAIROCH, A., LINIAL, M., BABBITT, P. C., BRENNER, S. E., ORENGO, C., ROST, B., MOONEY, S. D. and FRIEDBERG, I. A large-scale evaluation of computational protein function prediction. *Nat Meth*, vol. 10(3), pages 221–227, 2013.

REEVES, G. A., TALAVERA, D. and THORNTON, J. M. Genome and proteome annotation: organization, interpretation and integration. *Journal of the Royal Society Interface*, vol. 6(31), pages 129–147, 2009.

REINHARD, W., SCHWEITZER, J., VÖLKSEN, G. and WEBER, M. CSCW tools: Concepts and architectures. *Computer*, vol. 27(5), pages 28–36, 1994.

REINHARDT, A. and HUBBARD, T. Using neural networks for prediction of the subcellular location of proteins. *Nucleic Acids Research*, vol. 26(9), pages 2230–2236, 1998.

REMM, M., STORM, C. E. V. and SONNHAMMER, E. L. L. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of Molecular Biology*, vol. 314(5), pages 1041–1052, 2001.

RICE, P., LONGDEN, I. and BLEASBY, A. EMBOSS: the European Molecular Biology Open Software Suite. *Trends in Genetics: TIG*, vol. 16(6), pages 276–277, 2000.

ROST, B. and SANDER, C. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proc. Natl. Acad. Sci. USA*, vol. 90, pages 7558–7562, 1993.

RUST, A. G., MONGIN, E. and BIRNEY, E. Genome annotation techniques: new approaches and challenges. *Drug Discovery Today*, vol. 7(11 Suppl), pages S70–S6, 2002. ISSN 1359-6446.

SALI, A., GLAESER, R., EARNEST, T. and BAUMEISTER, W. From words to literature in structural proteomics. *Nature*, vol. 422(6928), pages 216–225, 2003.

SANDIA NATIONAL LABORATORIES. Jess, the Rule Engine for the Java Platform. `http://www.jessrules.com/`, 2009.

SAYERS, E. A General Introduction to the E-utilities. In *Entrez Programming Utilities Help [Internet]* (edited by Bethesda (MD): National Center for Biotechnology Information (NCBI)). National Center for Biotechnology Information, 2010. [Online: accessed 5-September-2015].

SAYERS, E. W., BARRETT, T., BENSON, D. A., BOLTON, E., BRYANT, S. H., CANESE, K., CHETVERNIN, V., CHURCH, D. M., DICUCCIO, M., FEDERHEN, S., FEOLO, M., FINGERMAN, I. M., GEER, L. Y., HELMBERG, W., KAPUSTIN, Y., KRASNOV, S., LANDSMAN, D., LIPMAN, D. J., LU, Z., MADDEN, T. L., MADEJ, T., MAGLOTT, D. R., MARCHLER-BAUER, A., MILLER, V., KARSCH-MIZRACHI, I., OSTELL, J., PANCHENKO, A. R., PHAN, L., PRUITT, K. D., SCHULER, G. D., SEQUEIRA, E., SHERRY, S. T., SHUMWAY, M., SIROTKIN, K., SLOTTA, D. J., SOUVOROV, A., STARCHENKO, G., TATUSOVA, T. A., WAGNER, L., WANG, Y., WILBUR, W. J., YASCHENKO, E. and YE, J. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, vol. 40(Database-Issue), pages 13–25, 2012.

SCHNOES, A. M., BROWN, S. D., DODEVSKI, I. and BABBITT, P. C. Annotation error in public databases: misannotation of molecular function in enzyme superfamilies. *PLoS Computational Biology*, vol. 5(12), page e1000605, 2009.

SCHREIBER, G., AKKERMANS, H., ANJEWIERDEN, A., DE HOOG, R., SHADBOLT, N., VAN DE VELDE, W. and WIELINGA, B. *Knowledge Engineering and Management – The CommonKADS Methodology*. MIT Press, Cambridge, MA, 2000. ISBN 978-0-262-19300-9.

SCHULER, G. D. Pieces of the puzzle: expressed sequence tags and the catalog of human genes. *J Mol Med*, vol. 75(10), pages 694–698, 1997.

SHEETLIN, S., PARK, Y. and SPOUGE, J. L. The gumbel pre-factor k for gapped local alignment can be estimated from simulations of global alignment. *Nucleic Acids Research*, vol. 33(15), pages 4987–4994, 2005.

SIGRIST, C. J. A., CASTRO, E. D., CERUTTI, L., CUCHE, B. A., HULO, N., BRIDGE, A., BOUGUELERET, L. and XENARIOS, I. New and continuing developments at PROSITE. *Nucleic Acids Research*, vol. 41(Database-Issue), pages 344–347, 2013.

SMIT, A., HUBLEY, R. and GREEN, P. RepeatMasker Open-4.0. `http://www.repeatmasker.org`, 2013-2015.

SONNHAMMER, E. L. and HOLLICH, V. Scoredist: A simple and robust protein sequence distance estimator. *BMC Bioinformatics*, vol. 6(108), 2005.

SONNHAMMER, E. L. and ÖSTLUND, G. InParanoid 8: orthology analysis between 273 proteomes, mostly eukaryotic. *Nucleic Acids Research*, vol. 43(D1), pages D234–D239, 2015.

STAJICH, J. E., BLOCK, D., BOULEZ, K., BRENNER, S. E., CHERVITZ, S. A., DAGDIGIAN, C., FUELLEN, G., GILBERT, J. G. R., KORF, I., LAPP, H., LEHVASLAIHO, H., MATSALLA, C., MUNGALL, C. J., OSBORNE, B. I., POCOCK, M. R., SCHATTNER, P., SENGER, M., STEIN, L. D., STUPKA, E. D., WILKINSON, M. and BIRNEY, E. The BioPerl Toolkit: Perl modules for the life sciences. *Genome Research*, vol. 12(10), pages 1611–1618, 2002.

STEIN, LINCOLN. Generic Feature Format Version 3 (GFF3). `http://www.sequenceontology.org/gff3.shtml`, 2013. [Online: accessed 5-September-2015.].

STORM, C. E. V. and SONNHAMMER, E. L. L. Automated ortholog inference from phylogenetic trees and calculation of orthology reliability. *Bioinformatics*, vol. 18(1), pages 92–99, 2002.

STUDER, R., BENJAMINS, V. R. and FENSEL, D. Knowledge engineering: principles and methods. *Data & Knowledge Engineering*, vol. 25(1–2), pages 161–197, 1998.

SUNDARARAJ, S., GUO, A., HABIBI-NAZHAD, B., ROUANI, M., STOTHARD, P., ELLISON, M. and WISHART, D. S. The CyberCell Database (CCDB): a comprehensive, self-updating, relational database to coordinate and facilitate in silico modeling of *Escherichia coli*. *Nucleic Acids Research*, vol. 32(suppl 1), pages D293–D295, 2004.

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., Wu, C. H. and the UniProt Consortium. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, vol. 31(6), pages 926–932, 2015.

Sycara, K., Decker, K., Pannu, A. S., Williamson, M. and Zeng, D. Distributed intelligent agents. *IEEE Expert*, vol. 11(6), pages 36–46, 1996.

The Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, vol. 25(1), pages 25–29, 2000.

The JBoss Drools Team. Drools expert user guide. `http://docs.jboss.org/drools/release/5.4.0.Final/drools-expert-docs/pdf/drools-expert-docs.pdf`, 2012. [Online: accessed 5-September-2015].

The UniProt Consortium. Activities at the Universal Protein Resource (UniProt). *Nucleic Acids Research*, vol. 42(D1), pages D191–D198, 2014. ISSN 1362-4962.

UniProt Consortium. Reorganizing the protein space at the universal protein resource (UniProt). *Nucleic Acids Research*, vol. 40(Database issue), pages D71–D75, 2012. ISSN 1362-4962.

Velasco, R., Zharkikh, A., Affourtit, J., Dhingra, A., Cestaro, A., Kalyanaraman, A., Fontana, P., Bhatnagar, S. K., Troggio, M., Pruss, D., Salvi, S., Pindo, M., Baldi, P., Castelletti, S., Cavaiuolo, M., Coppola, G., Costa, F., Cova, V., Dal Ri, A., Goremykin, V., Komjanc, M., Longhi, S., Magnago, P., Malacarne, G., Malnoy, M., Micheletti, D., Moretto, M., Perazzolli, M., Si-Ammour, A., Vezzulli, S., Zini, E., Eldredge, G., Fitzgerald, L. M., Gutin, N., Lanchbury, J., Macalma, T., Mitchell, J. T., Reid, J., Wardell, B., Kodira, C., Chen, Z., Desany, B., Niazi, F., Palmer, M., Koepke, T., Jiwan, D., Schaeffer, S., Krishnan, V., Wu, C., Chu, V. T., King, S. T., Vick, J., Tao, Q., Mraz, A., Stormo, A., Stormo, K., Bogden, R., Ederle, D., Stella, A., Vecchietti, A., Kater, M. M., Masiero, S., Lasserre, P., Lespinasse, Y., Allan, A. C., Bus, V., Chagne, D., Crowhurst, R. N., Gleave, A. P., Lavezzo, E., Fawcett, J. A., Proost, S., Rouze, P., Sterck, L., Toppo, S., Lazzari, B., Hellens, R. P., Durel, C.-E., Gutin, A., Bumgarner, R. E., Gardiner, S. E., Skolnick, M., Egholm, M., Van de Peer, Y., Salamini, F. and Viola, R. The genome of the domesticated apple (*Malus* times *domestica* Borkh). *Nature Genetics*, vol. 42(10), pages 833–839, 2010. ISSN 1061-4036.

Velasco, R., Zharkikh, A., Troggio, M., Cartwright, D. A., Cestaro, A., Pruss, D., Pindo, M., FitzGerald, L. M., Vezzulli, S., Reid, J., Malacarne, G., Iliev, D., Coppola, G., Wardell, B., Micheletti, D., Macalma, T., Facci, M., Mitchell, J. T., Perazzolli, M., Eldredge, G., Gatto, P., Oyzerski, R., Moretto, M., Gutin, N., Stefanini, M., Chen, Y., Segala, C., Davenport, C., DemattÃ¨, L., Mraz, A., Battilana, J., Stormo, K., Costa, F.,

TAO, Q., SI-AMMOUR, A., HARKINS, T., LACKEY, A., PERBOST, C., TAILLON, B., STELLA, A., SOLOVYEV, V., FAWCETT, J. A., STERCK, L., VANDEPOELE, K., GRANDO, S. M., TOPPO, S., MOSER, C., LANCHBURY, J., BOGDEN, R., SKOLNICK, M., SGARAMELLA, V., BHATNAGAR, S. K., FONTANA, P., GUTIN, A., VAN DE PEER, Y., SALAMINI, F. and VIOLA, R. A high quality draft consensus sequence of the genome of a heterozygous grapevine variety. *PLoS ONE*, vol. 2(12), page e1326, 2007.

VOSSEN, G. The CORBA specification for cooperation in heterogeneous information systems. In *Proceedings of the First International Workshop on Cooperative Information Agents* (edited by P. Kandzia and M. Klusch), vol. 1202 *of LNAI*, pages 101–115. 1997.

WAGNER, T. A., GARVEY, A. J. and LESSER, V. R. Complex Goal Criteria and its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. Providence, Rhode Island, 1997.

WALDROP, M. Big data: wikiomics. *Nature*, vol. 455(7209), pages 22–25, 2008.

WAN, H. and WOOTTON, J. C. A global compositional complexity measure for biological sequences: AT-rich and GC-rich genomes encode less complex proteins. *Computers & Chemistry*, vol. 24(1), pages 71–94, 2000. ISSN 0097-8485.

WEISS, G., editor. *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*. The MIT press, 1999.

WELCOME TRUST SANGER INSTITUTE. GFF (General Feature Format) Specifications Document. `http://www.sanger.ac.uk/resources/software/gff/spec.html`, 2012. [Online: accessed 5-September-2015].

WHEELER, D. L., CHURCH, D. M., EDGAR, R., FEDERHEN, S., HELMBERG, W., MADDEN, T. L., PONTIUS, J. U., SCHULER, G. D., SCHRIML, L. M., SEQUEIRA, E., SUZEK, T. O., TATUSOVA, T. A. and WAGNER, L. Database resources of the National Center for Biotechnology Information: update. *Nucleic Acids Research*, vol. 32 (Database Issue), pages D35–D40, 2004.

WIKIPEDIA. Alternative splicing — Wikipedia, the free encyclopedia. 2015. [Online: accessed 5-September-2015].

WOLSTENCROFT, K., HAINES, R., FELLOWS, D., WILLIAMS, A., WITHERS, D., OWEN, S., SOILAND-REYES, S., DUNLOP, I., NENADIC, A., FISHER, P., BHAGAT, J., BELHAJJAME, K., BACALL, F., HARDISTY, A., DE LA HIDALGA, A. N., BALCAZAR VARGAS, M. P., SUFI, S. and GOBLE, C. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research*, vol. 41(W1), pages W557–W561, 2013.

WU, C. H., YEH, L.-S. L., HUANG, H., ARMINSKI, L., CASTRO-ALVEAR, J., CHEN, Y., HU, Z., KOURTESIS, P., LEDLEY, R. S., SUZEK, B. E., VINAYAKA, C. R., ZHANG,

J. and BARKER, W. C. The Protein Information Resource. *Nucleic Acids Research*, vol. 31(1), pages 345–347, 2003.

XAVIER, D., CRESPO, B. and FUENTES-FERNÁNDEZ, R. A rule-based expert system for inferring functional annotation. *Applied Soft Computing*, vol. 35, pages 373 – 385, 2015. ISSN 1568-4946.

XAVIER, D., CRESPO, B., FUENTES-FERNÁNDEZ, R. and GÓMEZ-SANZ, J. J. MASSA: Multi-Agent System to Support Functional Annotation. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection* (edited by Y. Demazeau, F. Zambonelli, J. M. Corchado and J. Bajo), vol. 8473 of *Lecture Notes in Computer Science*, pages 291–302. Springer-Verlag, Berlin-Heidelberg, Germany, 2014.

XAVIER, D. and FUENTES-FERNÁNDEZ, R. An analysis of tool requirements for collaborative annotation. In *Proceedings of the 2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (edited by G. Fortino, W. Shen, J.-P. Barthès, J. Luo, W. Li, S. Ochoa, M.-H. Abel, A. Guerrieri and M. Ramos), pages 212–217. IEEE, Calabria, Italy, 2015.

XAVIER, D., MORÁN, F., FUENTES-FERNÁNDEZ, R. and PAJARES, G. Modelling knowledge strategy for solving the DNA sequence annotation problem through CommonKADS methodology. *Expert Systems with Applications*, vol. 40(10), pages 3943–3952, 2013.

ZACHARIAS, V. Development and Verification of Rule Based Systems – A Survey of Developers. In *Rule Representation, Interchange and Reasoning on the Web – International Symposium, RuleML 2008* (edited by N. Bassiliades, G. Governatori and A. Paschke), vol. 5321 of *Lecture Notes in Computer Science*, pages 6–16. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-88807-9.

ZAIN, J. M., MOHD, W. M. W. and QAWASMEH, E. E., editors. *Software Engineering and Computer Systems: Second International Conference. Proceding, Part I*. Springer, 2011.

ZDOBNOV, E. M. and APWEILER, R. InterProScan - an integration platform for the signature-recognition methods in InterPro. *Bioinformatics*, vol. 17(9), pages 847–848, 2001.

ZINCHENKO, V. P. Vygotsky's ideas about units for the analysis of mind. In *Culture, communication, and cognition: Vygotskian perspectives* (edited by V. Wertsch), pages 94–118. Cambridge: Cambridge University Press, 1985.

List of acronyms

AA	Amino Acid
AI	Artificial Intelligence
API	Application Programming Interface
AT	Activity Theory
BAL	BioAgent Language
BP	Biological Processes
CA	Controller Agent
CAFA	Critical Assessment of protein Function Annotation
CC	Cellular Components
CDD	Conserved Domains Database
CDS	Coding DNA Sequence
CF	Configuration File
COG	Clusters of Orthologous Groups
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CS	Conserved site
CSCW	Computer Supported Cooperative Work
CSV	Comma-Separated Values
DDBJ	DNA Data Bank of Japan
DNA	Deoxyribonucleic Acid
E-Utilities	Entrez Programming Utilities
e-value	expectation value
EAP	Ensembl Analysis Pipeline
EBI	European Bioinformatics Institute
EInfoF	External Information File
EInfoUA	External Information Updater Agent
EMBL	European Molecular Biology Laboratory
ENA	European Nucleotide Archive
EST	Expressed Sequence Tag
FSLF	Family Superfamily Links File
FSM	Finite State Machine
GB	GenBank
GCF	General CF

GFF	Generic Feature Format
GFF3	Generic Feature Format version 3
GO	Gene Ontology
GOPRR	Graph, Object, Property, Role, Relationship
HMM	Hidden Markov Model
IA	Interface Agent
IEA	Information Extraction Agents
InfA	Inference Agent
IntA	Intelligent Agent
JESS	Java Expert System Shell
JSON	JavaScript Object Notation
KB	Knowledge Base
KBS	Knowledge-Based System
KE	Knowledge Elicitation
KEGG	Kyoto Encyclopedia of Genes and Genomes
KQML	Knowledge Query and Manipulation Language
LA	Launcher Agent
LKBMA	Local Knowledge Base Management Agent
MAS	Multi-Agent System
MASSA	Multi-Agent System to Support functional Annotation
MF	Molecular Functions
mRNA	messenger RNA
MUSCLE	MUltiple Sequence Comparison by Log-Expectation
NA	Not Applicable
NCBI	National Center for Biotechnology Information
NIH	National Institutes of Health
NP	Nondeterministic Polynomial
NR	Database of non-redundant protein sequences
NT	Database of nucleotide sequences
PANTHER	Protein ANalysis THrough Evolutionary Relationships
PCLF	Potential Candidates List File
PDB	Protein Data Bank
PHYLIP	PHYLogeny Inference Package
PIR	Protein Information Resource
PRF	Protein Research Foundation
PRK	PRotein K(c)lusters
ProDom	Protein Domain
RBES	Rule-Based Expert System
RDB	Result Database
RMI	Remote Method Invocation
RNA	Ribonucleic Acid
rRNA	ribosomal RNA
SD	Service Description

SDB	Search Database
SDBUA	Search Database Updater Agent
SVG	Scalable Vector Graphics
TA	Tool Agent
TB	Task Blackboard
TBS	Threshold Bit Score
TBSF	TBS File
TF	Trace File
TLF	Task Log File
tRNA	transfer RNA
TSV	Tab-Separated Values
WFSX	Well-Founded Semantics with eXplicit negation
WM	Working memory
XML	EXtensible Markup Language
YP	Yellow Pages

Alice: Would you tell me, please, which way I ought to go from here?
The Cheshire Cat: That depends a good deal on where you want to get to.
Alice: I don't much care where.
The Cheshire Cat: Then it doesn't much matter which way you go.
Alice: ...So long as I get somewhere.
The Cheshire Cat: Oh, you're sure to do that, if only you walk long enough.

Alice in Wonderland
Lewis Carroll

Why it's simply impassible!
Alice: Why, don't you mean impossible?
Door: No, I do mean impassible. (chuckles) Nothing's impossible!

Alice's Adventures in Wonderland & Through the Looking-Glass
Lewis Carroll