BioSmalltalk: A pure object system and library for bioinformatics
Hernán F. Morales* and Guillermo Giovambattista

Instituto de Genética Veterinaria (IGEVET), CONICET La Plata – Facultad de Ciencias Veterinarias, Universidad Nacional de La Plata, Argentina, Calle 60 y 118 s/n, La Plata B1900AVW, CC 296, Argentina.

ABSTRACT
Summary: We have developed BioSmalltalk, a new environment system for pure object-oriented bioinformatics programming. Adaptive end-user programming systems tend to become more important in the discovering of biological knowledge, as the emergence of open-source programming toolkits for bioinformatics have demonstrated in the last years. Our software is intended to bridge the gap between bioscientists and rapid software prototyping while preserving possibility of scaling to whole systems biology applications. BioSmalltalk performs better in terms of execution time and memory usage than Biopython and BioPerl for some classical situations. BioSmalltalk is cross-platform and freely available (MIT license) through the Google Project Hosting at http://code.google.com/p/biosmalltalk.

1 INTRODUCTION

We present a novel Free/Open Source Software (FOSS) platform for the development of bioinformatics software and applications. BioSmalltalk attempts to reconcile the current de facto scripting modalities of textual programming languages with the features of Smalltalk (Goldberg and Robson, 1983), a pure object dynamic programming environment.

BioSmalltalk provides similar functionality to other FOSS toolkits for bioinformatics like BioPerl (Stajich et al., 2002), Biopython (Cock et al., 2009) and BioJava (Holland et al., 2008), based in industry-leading general-purpose textual programming languages. Precedent of bioinformatics tools exists in Smalltalk but none of them tried to provide a bioinformatics API. MolTalk (Diemand and Scheib, 2004) was developed in StepTalk, a scripting environment, for doing Structural Bioinformatics. Also a cross-platform GUI for protein sequence analysis was done in Smalltalk (Wishart et al., 1997).

Object-orientation (OO) is a term first coined by one of the Smalltalk inventors, Alan Kay (1969). It was initially conceived as a programming paradigm based in the recognition of real-world communicating objects in computer simulations (Fichman and Kemerer, 1992). Kolling, (1999). OO features were integrated accordingly to platform limitations, in virtually all major programming languages, and toolkits, including the Bio* projects. The Bio* toolkits usage of OO is commonly hybrid or emulated through modules (Stajich et al., 2002. Cock et al., 2009), mixing objects with primitive data types and hampering the use of reflective functionalities (Maes, 1977). BioSmalltalk benefits from decreased source code verbosity, and its execution in a self-contained snapshot system which promotes run-time adaptability, critical for systems where shutdown cycles cannot be tolerated (Hirschfeld and Lämmel, 2005).

2 FEATURES

2.1 Bioinformatics

BioSmalltalk provides objects to manipulate biological sequences and data from databases like the Entrez system (Schuler et al., 1996), wrappers for command-line tools like ClustalW (Thompson, 1994) and HMMER (Finn, 2011), sequence visualization and format conversion.

We based implementation on existing FOSS bioinformatics platforms, specifically BioPerl and Biopython, to prevent educational obsolescence preserving the familiar object model interfaces to experienced bioinformaticians.

```
outFileStream := BioObject newFullFileNamed: 'out.txt';
msa := BioAlignment new;
(BioNCBIBlastReader newFromXML: 'BS617-Alignment.xml')
    selectedNodes: #('Hit_id' 'Hsp_align-len' 'Hsp_hseq');
hitNodesDo: [ :hitNode |
    hspNodesDo: [ :hspNode |
       hspNode selectHspAlign = 240 ifTrue: [
        msa addSequence: (BioSequence
           newName: hitNode selectAccessionNumber
              sequence: hspNode selectHspSeq)]].
msa asFasta outputTo: outFileStream.
outFileStream close.
```

Fig. 1. A downloaded NCBI BLAST XML result is filtered with alignments matching 240 nucleotide bases. Alignment object is built from sequences and exported as FASTA in a file.

BioSmalltalk contains tokenizers, parsers and formatters for common sequence identifiers, FASTA, BLAST and Entrez XML, PHYLIP (Felsenstein, 1989), Arlequin (Excoffier, 2005), and others. Most parsers use PetitParser (Renggli et al., 2010) a dynamically reconfigurable parser library. Additional features can be found in the project documentation. We did a microbenchmark to compare the performance of our library using the script in Fig. 1. We have executed the scripts 5 times immediately after booting
without unnecessary processes\(^1\). Results shown BioSmalltalk has a faster execution time compared with the corresponding BioPerl and Biopython versions. Our approach enabled to remove unnecessary iterations and thus reducing also lines of code (LOC). Comparison details and scripts are included in the supplementary material.

<table>
<thead>
<tr>
<th>Environment</th>
<th>LOC</th>
<th>Avg. exec. time (msec)</th>
<th>Peak memory usage (Mbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BioPerl</td>
<td>23</td>
<td>11.496</td>
<td>88.035</td>
</tr>
<tr>
<td>Biopython</td>
<td>18</td>
<td>9.595</td>
<td>47.443</td>
</tr>
<tr>
<td>BioSmalltalk</td>
<td>11</td>
<td>8.073</td>
<td>45.717</td>
</tr>
</tbody>
</table>

### 2.2 Software Engineering

We wrote a cross-platform engine subsystem for enabling multiple interchangeable implementations of packages which may fail, become unsupported, or too slow. Interchangeable serializers, web client & servers, accessing OS functions, were included in our initial release. We have applied design patterns through the library, for example, the Façade pattern (Gamma et al. 1994) is used to wrap the complex hierarchy of specific parsers. The developer guide provides further information of new engines, testing process, and other subsystems.

Maintainability was recognized as an unfilled gap in bioinformatics software development (Umarji and Seaman, 2008). An advantage of BioSmalltalk is relying on a development style promoting highly factored reusable code (Boehm, 1986) using browsers and inspectors in a targeted navigation fashion (Bergel et al., 2007, Bunge, 2009), applying automated code refactorings directly through menu options (Opdyke, 1992). This style replaces taking care of boilerplate code, static or primitive type coercion casting, checking class or function scopes, maintaining directory trees, configuration files or compiler flags.

The software was tested on Windows, Linux and Mac OS X platforms under Squeak and Pharo Smalltalk (Black et al. 2009).

### 3 RESULTS

We delivered an interactive programming system using a fully reflective language for bioinformatics development. We believe that our platform is suitable for a bioinformatics evolution to human-centered long-running software. Of interest for future research is building an user-base and solid automated build process. We are open to collaboration in any of the areas in which BioSmalltalk project can evolve.

### ACKNOWLEDGEMENTS

Funding: This work is supported by The National Scientific Research Council in Argentina (CONICET).

Conflict of Interest: None declared.

### REFERENCES


Thompson,J.D. et al. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting


\(^1\) Tests were performed on GNU/Linux Debian kernel 2.6.32-358.2.1.el6.x86_64 using an Intel(R) Xeon(R) CPU E5620 at 2.40GHz, 8 GB DDR3 RAM