The development of SHAP began in early 2006, after it primarily on the analysis of single microbial genomes (Almeida et al., 2003; Meyer et al., 2004; Hemmerich et al., 2010) and the derived annotation- specific server ISGA (Hemmerich et al., 2010) have become available. Streamlining the capacity of Ergatis, ISGA addresses many of the issues that motivated the creation of SHAP. However, the ambitious nature of the system makes it complex and demanding to deploy as a monolithic application. As the name implies, SHAP provides a relatively simple means of annotating high-throughput DNA sequencing datasets while, at the same time, allowing for customization and expansion.

2 DESIGN AND IMPLEMENTATION

SHAP is implemented entirely in Java, utilizing Spring (http://www.springsource.org) as an application framework; Hibernate (http://www.hibernate.org) for relational storage and search integration and Apache Lucene (http://lucene.apache.org) for full-text indexing and search capabilities (Fig. 1). The design of SHAP pays attention to the rapid and continued development of metagenomics by avoiding implementation of complex or fine-scale features. A layered separation of concerns allowed the isolation of domains with the greatest potential for change. Driven by developments in methodology and platform selection, if left unconsidered this change would represent a significant cost in ongoing codebase maintenance. The selection and behaviour of the underlying analysis tools (BLASTALL, HMMER, etc.) was identified as one such area and therefore their execution detail and subsequent result parsing was isolated within a single layer.

By way of Inversion of Control, much of SHAP’s instantiation logic of collaborating objects is XML configuration detail, providing flexibility in differing post-deployment environments without the need for code changes. By introducing an ORM (Hibernate), the persistence layer is only loosely coupled to the underlying relational database provider. Full-text search facilities (Hibernate Search/Apache Lucene) simplifies user-driven retrieval of results for downstream analysis. High-throughput analysis is achieved by supporting both conventional multi-threaded local execution and Grid computing by way of DRMAA.

SHAP’s minimalist persistent domain model begins with a Project which contains one or more Samples. Following this template, biological data is organized hierarchically via cascading one-to-many associations; where instances of Project and Sample are user defined; instances of Sequence are defined by imported sequencing data; and Feature and Annotation instances are the derived result of computational analyses. The database schema emitted by the Hibernate mapped object model parallels a subset of BioSQL (http://www.biosql.org), with the notable exception that, for the sake of simplicity, SHAP makes only light use of ontological terms and adopts no official ontology (Eilbeck et al., 2005).

A computational analysis is represented by a Job that is parcelled into computationally independent Tasks. Job and Task store historical runtime metadata as well as execution state, permitting work to be stopped, restarted and reattempted. Jobs are divided into two classes: detection, which determine Features from Sequences, and Annotation, which determine Annotations from Features. Within each class, the workload is embarrassingly parallel, allowing for concurrent execution of all outstanding work at the task level.

SimpleDetector and SimpleAnnotator are primary objects supporting analysis within SHAP, designed for simplicity and reusability with underlying analysis tools by imposing a basic contract. The contract stipulates that an analyzer acts on a specific target object type (Sequence for SimpleDetector, Feature for SimpleAnnotator) and produces a specific result object type...
The persistence domain and analysis throughput, not become overly that could deal with large volumes of data, scale both in the larger Ergatis framework (73 227 physical LOC) for workflow comparative reduction could be greater still if ISGA’s dependence on 2000), there remains a 38% reduction in codebase size. This the relatively higher level nature of PERL to Java (6:2.5) (Prechelt, respectively. Ignoring a possible further normalization to account for Perl, whereas SHAP comprises 16 395 and 10 683 lines of Java, that ISGA comprises 26 867 physical and 17 305 logical lines of 2007) applied to the core codebases of SHAP and ISGA finds the relatively higher level nature of PERL to Java (6:2.5) (Prechelt, 2000), there remains a 38% reduction in codebase size. This comparative reduction could be greater still if ISGA’s dependence on the larger Ergatis framework (73 227 physical LOC) for workflow support was included in the metric. However, the SHAP and Ergatis frameworks should not be compared as simplicity of purpose brings forth a feature disparity particularly in the subtly of process.

From the outset of development, the goal was to build a system that could deal with large volumes of data, scale both in the persistence domain and analysis throughput, not become overly complex or difficult to adapt in a developing field of research and allow migration between platforms. Through a layered design, the areas of greatest change are isolated from the rest of system and thereby insulated from their changing habits. SHAP’S Web interface provides simple and expedient access to the annotation results, presenting users with a familiar means of access.

SHAP has been in active use within the group, having supported recent research activities (Ng et al., 2010; Lauro et al., 2011; Yau et al., 2011). Deployed on our in-house 64 core cluster, the running system has accumulated over 13 years of computational time in 97 days, analysing 2.6 million DNA contigs (2.1 billion bases), producing 4.1 million ORF predictions (578 million amino acids) and recording 7.7 million significant annotation results.

Future development may involve converging SHAP to OSGi dynamic modules to further enhance platform independence and ease of deployment.

**Funding:** Australian Research Council.

**Conflict of Interest:** none declared.

![Image](image-url)

**Fig. 1.** A simplified representation of the application layers within SHAP. The Data Service façade provides abstracted transactional access to persistent storage, retrieving information conventionally by reference or full-text search. The Batch Service façade offers a similar abstraction for job processing, either by local execution or queue submission via DRMAA.

**REFERENCES**


