**2. Specific Aims (one page maximum; separate PDF attachment)**

NextGen sequencing technologies are fast approaching the ‘$1,000 genome’ target (1): a $5,000 genome will be available in May 2009 by Comparative Genomics while other NextGen industry players are rapidly increasing run yield and reducing cost per Mbase. A new paradigm is emerging of the correlated and rapid analysis of individual genomic variation, methylation, histone-binding, expression analysis and other genome-wide factors that may begin to unlock the secrets of the cell (2) and create new avenues for clinical diagnostics. Bioinformatics infrastructure – hardware, software and personnel – is the bottleneck in the development of this new paradigm (2, 3). Distributed processing on high performance computing clusters is necessary in order to cope with the large data volumes and reduce the cost of processing. Furthermore, costly investments in skilled personnel are also required to develop, evaluate and run bioinformatics algorithms, and to integrate diverse biological data sources. Most biomedical research and diagnostics labs are unable to provide even the minimum of these hardware and personnel requirements. With regard to software, workflow tools are essential to allow non-technical staff to automate and run well-defined but complex analysis processes. These tools must also be flexible enough to support exploratory analysis through interaction with the data using a wide range of different software applications and data processing steps. Ideally, workflow tools should also be web-enabled for flexibility of access by users and integration with other bioinformatics web resources. They should also provide visualization functionality capable of handling large volumes of NextGen data and integrating heterogeneous external genome feature data sets. Given the budget considerations mentioned above, the ideal workflow tool should also be open source and freely available to the academic community. At present, there are no tools available that span the whole process from NextGen sequence generation to analysis and visualization.

To help address these opportunities, we propose the rapid deployment of a software system and analysis tools for managing NextGen sequencing projects, from short read generation to bioinformatics analysis to data visualization. The system will meet the following challenges: 1) facilitating the analysis of large-scale sequencing studies, 2) enabling transcriptome and genetic network analyses, and 3) determination of the relationship of sequence variation and phenotypes to disease. These challenges will be addressed through the following specific aims:

**Specific Aim 1: Implement an optimized assembly workflow based on the evaluation of current NextGen assemblers/aligners.**

Evaluate objectively

Enable pipeline customization by end user

**Specific Aim 2: Implement SNP annotation, expression analysis and network analysis workflows**

We propose developing a SNP annotation pipeline with defined quality control/assurance algorithms for SNP verification. The pipeline will be integrated with current expression analysis packages and will develop new expression analysis algorithms. As part of the reporting and visualization of results, data filters will be designed based on user requirements to extract result subsets and provide genome-level views of the results integrated with external genomic features. Results will also be exportable to downstream analysis applications (Cytoscape, Genespring, R, etc.).

**Specific Aim 3: Development of workflow and visualization tool tailored to the requirements of Next Gen analyses**

We propose developing a workflow tool that supports optimized and customizable workflows

Establish software design plan based on requirements in Aims 1 and 2.

Implement user-centric development process to ensure usability

Execute thorough post-release user testing at each project milestone

**5. Research Design and Methods (12 pages maximum; separate PDF attachment)**

**Challenge Area:** *06: Enabling Technologies***Challenge Topic:** *06-HG-101\*   
New computational and statistical methods for the analysis of large data sets from next-generation sequencing technologies.*

**The Challenge and Potential Impact:** *What is the research opportunity, scientific knowledge gap or technology that will be addressed? How broad is the potential impact in science and/or health? Which community (ies) will be affected? What is (are) the size(s) of the community(ies)? Will the potential impact be major?*

The following sections describe the particular challenges of NextGen technologies, applications and bioinformatics in more detail and discuss the anticipated impact of solutions provided by this study.

**Specific Aim 1: Implement an optimized assembly workflow based on the evaluation of current NextGen assemblers/aligners.**

The impact of achieving Aim 1 will be to accelerate the development of NextGen sequencing assembly tools by providing an improved methodology for sequence assembly/alignment and the tools do so on a routine basis. Objective performance comparisons of assemblers/aligners involving preset criteria and using a wide range of data sets will provide a basis for comparisons between assemblers.

Optimized pipelines based on these evaluations will allow users to improve their work efficiency and the quality of their results. At the other end of the spectrum, customizable pipelines will help meet the bioinformatics challenges faced by researchers at the cutting edge life science exploration. Dynamic pipeline configuration coupled with high performance computing will enable researchers and other end users to rapidly develop and adapt different approaches to solving particular problems.

**NextGen Technologies**

The current mainstream NextGen platforms produce millions of short (50bp – 400bp) sequence reads. Each of the three main platforms, namely, Illumina/Solexa (4), Roche/454 (5) and ABI/SOLiD (6) have their own inherent problems, including significant sequencing error rates and systematic errors. Large sequencing organizations such as genome centers, academic core facilities and commercial contract-sequencing enterprises across the globe have already adopted this NextGen technology (Figure 1) and smaller labs and molecular diagnostics facilities participating in growing numbers. A common refrain among adaptors of this technology is that the downstream bioinformatics analysis are often poorly understood and underestimated.

This study aims to reduce these hidden costs by providing a free tool for accomplishing the entirety of common tasks in NextGen sequence analysis. Extensive training for users will not be required as only a basic familiarity with web page navigation and drag and drop user interfaces is assumed.

Prior to sequencing, NextGen sample preparation varies considerably but usually involves multiple steps taking 2–4 days to complete,depending on the platform. ‘Barcodes’ – unique identifier sequences added to reads – can also be used to analyze multiple samples within the same separate flow-cell lanesor compartments. These barcodes are ligated to individual samples which are then pooled and sequenced and later separated out based on their barcode.Barcode-based multiplexing and other incremental innovations in process streamlining, automation and chemistryrefinements will continue to reduce costs and sequencing errors. The recent rollout of paired-end reads (a.k.a. mate-paired reads) – short reads that flank a region of known length in the sample sequences - by all of the major platforms has provided a major advance in de novo assembly and the correction of reference alignment errors by eliminating alignments that do not match the size of gap between the paired reads (7). There is also a strong need for flexible and effective targeted capture methods for isolating reduced genomic subsets – such as genomic regions or exons of candidate genes – implicated in disease prior to NextGen sequencing of multiple individuals. Different approaches have already shown proof-of-concept, such as 10,000-fold enrichment by hybridizing biotinylated BACs (bacterial artificial chromosomes) with targeted segments of genomic DNA (8), microarray-based enrichment of several kilobase-sized human genomic regions (9) and multiplex PCR amplification of 170 exons (10). Two commercial capture methods are currently available from Nimblegen and Agilent.



Figure 2. NextGen Sequencing Technology Roadmap

**3G Technologies**

Third generation sequencing technologies are being developed to sequence single DNA molecules faster and cheaper with streamlined samplepreparation. Real-time sequencing by synthesis is being developedby VisiGen (<http://www.visigenbio.com>) and Pacific Biosciences(<http://www.pacificbiosciences.com>). Pacific Biosciences is due to launch commercially in 2010 and has a mean DNA synthesisrate of approximately 4 bases per second, with a maximum read length of 4,000 bp.Also in development is sequencing based on sensing the bases of DNA molecules passed through nanopores (~5 nmin diameter). Different methods are being tested to create nanopores, including inorganicmembranes (solid-state nanopores), genetically engineered protein channelsby Oxford Nanopore Technologies(<http://www.nanoporetech.com>), polymer-based nanofluidicchannels, and a combinationof nanopores with sequencing by hybridization by NABsys (<http://www.nabsys.com>). As these technologies develop, they will present new bioinformatics problems to be solved and greater data infrastructure demands.

**NextGen Applications**

A growing variety of molecular methods has enabled the investigation of a broad range of biological phenomena by high-throughput DNA sequencing, including genetic variation, RNA expression, protein-DNA interactions and chromosome conformation (7). Chromatin immuno-precipitation (ChIP) is used to investigate protein-DNA interactions, which play a key part in regulating gene expression and controlling the availability of DNA for transcription and replication. In the technique, DNA chemically cross-linked to associated proteins is fragmented and transcription factor-specific antibodies are used to immunoprecipitate selected protein-DNA complexes. The DNA is then processed by NextGen sequencing (ChIP-Seq). Expression profiling (a.k.a. RNA-seq or transcriptome analysis) is another popular NextGen application that has been demonstrated as robust and sensitive in comparison to five microarray platforms (11). Furthermore, microarrays cannot detect antisense transcription, which was found in 51% of all genes. Transcriptomes for mouse brain, liver and skeletal muscle were mapped by NextGen deep sequencing (12), providing a digital measure of the presence and prevalence of transcripts from known and previously unknown genes. RNA standards were used to quantify transcript prevalence and to test the linear range of transcript detection, which spanned five orders of magnitude.

In yeast, NextGen expression analysis has demonstrated a larger, more complex transcriptome than had been expected (13). An estimated 74.5% of the non-repetitive yeast genome was shown to be transcribed, as were many overlapping genes, alternative initiation codons and upstream open reading frames of yeast genes were demonstrated using short reads to generate a high-resolution map of the genome. Similarly, the first high-resolution map of human genome structural variation revealed complex and large-scale structural variation in the form of insertions, deletions and inversions from a few thousand to millions of base pairs in length (14). Somatically acquired genomic rearrangements have been implicated in cancer development. Paired-end read pairs that did not align correctly with respect to each other on the reference human genome, were used to characterize 306 germline structural variants and 103 somatic rearrangements to the base-pair level of resolution (15). The results demonstrate the feasibility of using NextGen sequencing for the systematic, genome-wide characterization of rearrangements in human cancer genomes. At the level of large-scale genomic variation, copy number variations (CNV) remain difficult to measure although CNVs of 100 kilobases and greater contribute substantially to genomic variation between normal humans (16, 17). Microarray-based approaches for detecting CNVs depend on microarray signal intensity differences to predict regions of variation and cannot detect inversions. Before the advent of NextGen CNV technique, only a small fraction of copy-number variant (CNV) base pairs had been determined at the sequence level (18). NextGen CNV mapping allows the discovery cancer-causing genes in genomic regions that show recurrent copy-number alterations (gains and losses) in tumor genomes (19).

Other applications include discovering non-coding RNAs (e.g., miRNAs), sequencing the nuclear genomes of extinct species and metagenomics, such as the characterization of changes in biodiversity due to climate changes. For metagenomics, the growing number of sequenced genomes enables us to interpret partial sequences obtained by direct sampling of specific environmental niches to determine which kinds of species are present. The rapid, inexpensive, and massive data production enabled by NextGen platforms has caused a recent explosion in metagenomic studies. The NIH Human Microbiome Project is one of several international efforts using metagenomic analysis to study human health and has developing the new technological and bioinformatics tools as one of its four stated goals (http://www.genome.gov/25521743). New bioinformatics tools for assembling metagenomics data (http://nihroadmap.nih.gov/hmp/fundedresearch.asp) are required to assembling and finding genes and genomic variation in heterogeneous metagenomic datasets, where currently available software performs poorly.

Alongside the profound impact of NextGen applications in basic research, high throughput sequencing is now being adopted by clinical diagnostics laboratories for applications requiring deep sequence coverage and high-sensitivity such as rare HIV drug resistant variant detection (20). As the focus in human genetics has shifted to complex, multi-gene diseases, there is an increasing need for comprehensive diagnostic evaluations of multiple genes, enhanced by sequence enrichment/capture methods. NextGen analysis of placental mRNA - counting the numberof reads that map to each chromosome – has been used to confirm trisomy 21pregnancies, with additional supporting evidence obtained fortrisomy 18 and 13 pregnancies (REF). Other novel applications include the sequencing of ancient DNA samples and large-scale metagenomic analysis of environmentally derived samples.

**NextGen Bioinformatics**

The anticipated growth of NextGen sequencing among clinical diagnostics labs must be accompanied by streamlined sample preparation methods and improved robustness through characterization of accuracy in validation studies [6]. Biomedical research labs also require methods for mitigating systemic bias in next generation data analysis. Particularly in the early phases of the development of NextGen technology while many competing algorithms vie for supremacy, scientific publications will require comparisons of results using several different sequence analysis algorithms. At present there are insufficient studies comparing the efficacy and applicability of the various tools. An objective, performance-based comparison of NextGen bioinformatics tools is an important step towards lowering the bioinformatics hurdle and allowing researchers to answer more penetrating questions more convincingly and in less time. Bioinformatics tools are available for reference alignment, de novo assembly, variant-discovery and alignment viewing. Among the reference aligners are Eland (GAPipeline v0.30, Illumina), Mira (21), Genomics Workbench (CLC Bio), Seqman NGen (DNAStar), NextGene (Soft Genetics), MAQ (22) (23) and Shrimp http://compbio.cs.toronto.edu/shrimp). De novo assemblers include Edina (24), EULER-SR (25), SHARCGS (26), SSAKE (27),Velvet (28), and SOAPdenovo (http://soap.genomics.org.cn). Some NextGen statistical data-analysis tools are also available, such as JMP Genomics (<http://www.jmp.com/software/genomics>).

Despite the growing number of software packages available for NextGen data, obtaining an accurately assembled sequence contig is still a very challenging problem. The currently available assembly/alignment programs vary widely in terms of data volume capacity (e.g., bacterial versus human data sets), number of reads aligned/assembled, error rates and bias, all of which may lead to suboptimal assemblies. Little is known about the comparative performance of the available tools because the scarce available performance statistics are based mainly on different non-human data sets results (e.g., phage, bacteria, yeast). So comparison between assemblers is difficult even before considering the particular performance with human data.

A performance-based comparison of these tools is an important precondition for mitigating systemic bias in next generation data analysis. One anticipated outcome of this will be hastening the transition to a mature technology, with fewer bioinformatics applications used for a wide number of applications. In some cases, extensive comparisons may be needed to determine that certain bioinformatics approaches are suitable for a particular task. This study proposes to facilitate multi-tool comparisons by providing a workflow system to enable high-throughput assembly with multiple algorithms in parallel.

Data visualization and interpretation become paramount as the bioinformatics challenge shifts from mastering the basic tools to gaining biological insights from huge amounts of data. Three commercial software packages by DNAStar, SoftGeneticsand CLC Bio contain data viewers that allow the user to see read alignments, coveragedepth, genome annotations, and variant analysis. However, they as yet lack the capability for viewing data sets as large as a whole human chromosome and show poor performance even on sub-chromosome data sets. The three major genome viewers – UCSC Genome Browser (REF), Ensembl genome browser (REF) and GBrowse (REF) – are based on the traditional client-server model where user page clicks result in a reloaded image file sent from the server. Java-based applications such as Apollo (REF) lack a concerted approach to data sharing although newer applications IGV (REF) do allow for limited filtering of the displayed features (see Figure 2).

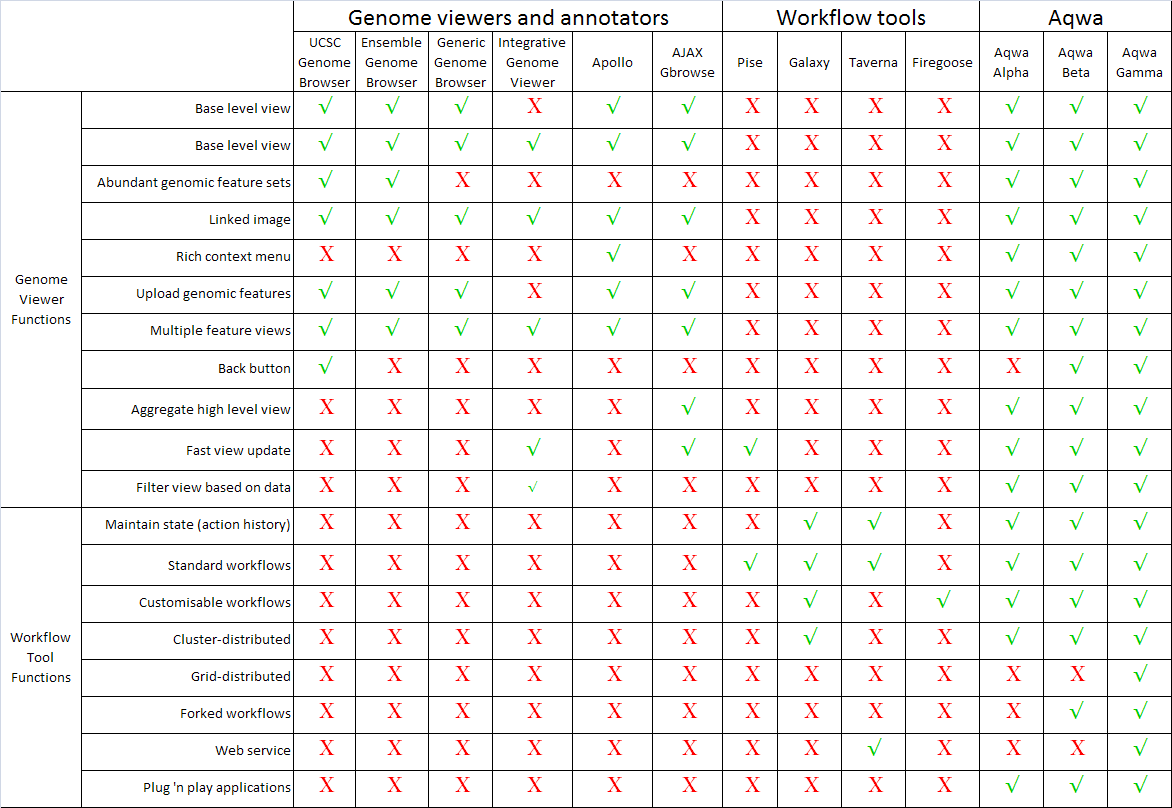


Figure 2. A comparison of genome viewer and bioinformatics workflow tool functionalities.

BioCoRE, Pegasys, Wildfire,

Taverna, and BCJ allowed execution on a local cluster.

Only Pegasys, Wildfire, Taverna, and the BCJ offer workflow

capabilities.

At the other end of the spectrum from NextGen sequence assembly, the few studies on common bioinformatics tasks (Stephens, 2001) and usability of bioinformatics tools (Colchini, 2007) identified a lack of workflow-based tools for bioscience researchers. The majority of bioinformatics workflow tools only partially meet the need to be able to routinely design and run complex and robust workflows using multiple applications and data sets. Bioinformatics web sites are mostly simple GUIs (Graphical User Interfaces) with mutually incompatible input/output formats. Anyone seeking

With notable exceptions (Firegoose paper 2007), there has been limited progress in connecting different sites with the client as the intermediary. Another approach is to use a central site as a service directory lookup (e.g., BioMOBY) with the attendant limitations due to the lack of a commonly-accepted ontology for biological data despite several attempts to create one (e.g., Tambis – Baker 1999, …).

Biosciences are becoming increasingly quantitative and it is hard to see people in this area doing much without getting data collected and being analyzed into software.

Creating a bioinformatics nation

<http://www.nature.com/nature/journal/v417/n6885/full/417119a.html>

Distiction between analysis workflows and production workflows

Workflow tools enable users to automate well-defined, repetitive processes and to explore data with ad-hoc analyses. They generate large and complex data processing systems from distinct programs and data sources by leveraging the abundant biological data applications and the growing number of structured data and computational services accessible using common web protocols. We propose a different approach to that of existing workflow tools: providing an end-to-end integrated tool for running highly customized workflows and complex data analyses, followed by real-time genomic visualization of huge data sets.

**A Survey of Data Provenance in e-Science**

Yogesh L. Simmhan Beth Plale Dennis Gannon

Computer Science Department

Indiana University, Bloomington, IN 47405

{ysimmhan, plale, gannon}@cs.indiana.edu

**ABSTRACT**

Data management is growing in complexity as largescale

applications take advantage of the loosely coupled

resources brought together by grid middleware and by

abundant storage capacity. Metadata describing the data

products used in and generated by these applications is

essential to disambiguate the data and enable reuse. Data

provenance, one kind of metadata, pertains to the

derivation history of a data product starting from its

original sources.

Tilson 2007 Workflows ppt

Taverna, Complex workflows

* Service Technologies
  + BioMoby
  + Seqhound
  + Soaplab
  + WSDL
  + Custom
  + Etc.
* Taverna can support all of these ( and more)
  + **Develop useful Data-flow workflows**

Shannon 2006 Gaggle

Guided by the classic

software engineering strategy of *separation of concerns* and a policy of *semantic flexibility*, it integrates

existing popular programs and web resources into a user-friendly, easily-extended environment

Tambis Manchester

There are three main types of information source: databases, online services, files (Tambis ppt).

Researchers in biological sciences often gather and compare sequencing and other heterogeneous information. This process takes significant manual effort and is a constant barrier to progress.

There are over 200 biological information sources world-wide

Web resources may be underused if users feel too much time is used selecting among appropriate sources, downloading and uploading files,

**Pegasys: software for executing and integrating analyses of**

**biological sequences**

Sohrab P Shah, David YM He, Jessica N Sawkins, Jeffrey C Druce,

Gerald Quon, Drew Lett, Grace XY Zheng, Tao Xu and BF Francis Ouellette\*

Address: UBC Bioinformatics Centre, University of British Columbia, Vancouver, British Columbia, Canada

Email: Sohrab P Shah - sohrab@bioinformatics.ubc.ca; David YM He - david@bioinformatics.ubc.ca;

Jessica N Sawkins - jessica@bioinformatics.ubc.ca; Jeffrey C Druce - jdruce@bioinformatics.ubc.ca; Gerald Quon - gtquon@uwaterloo.ca;

Drew Lett - drewlett@bioinformatics.ubc.ca; Grace XY Zheng - gxz@interchange.ubc.ca; Tao Xu - taoxu@bioinformatics.ubc.ca; BF

Francis Ouellette\* - francis@bioinformatics.ubc.ca

\* Corresponding author

**Abstract**

**Background:** We present Pegasys – a flexible, modular and customizable software system that

facilitates the execution and data integration from heterogeneous biological sequence analysis

tools.

**Results:** The Pegasys system includes numerous tools for pair-wise and multiple sequence

alignment, *ab initio* gene prediction, RNA gene detection, masking repetitive sequences in genomic

DNA as well as filters for database formatting and processing raw output from various analysis

tools. We introduce a novel data structure for creating workflows of sequence analyses and a

unified data model to store its results. The software allows users to dynamically create analysis

workflows at run-time by manipulating a graphical user interface. All non-serial dependent analyses

are executed in parallel on a compute cluster for efficiency of data generation. The uniform data

model and backend relational database management system of Pegasys allow for results of

heterogeneous programs included in the workflow to be integrated and exported into General

Feature Format for further analyses in GFF-dependent tools, or GAME XML for import into the

Apollo genome editor. The modularity of the design allows for new tools to be added to the system

with little programmer overhead. The database application programming interface allows

programmatic access to the data stored in the backend through SQL queries.

**Conclusions:** The Pegasys system enables biologists and bioinformaticians to create and manage

sequence analysis workflows. The software is released under the Open Source GNU General

Public License. All source code and documentation is available for download at http://

bioinformatics.ubc.ca/pegasys/.

# GPipe workflow constructor for Pise

# Workflows in bioinformatics: meta-analysis and prototype implementation of a workflow generator

**Alexander Garcia Castro**1,2 , **Samuel Thoraval**2,3 , **Leyla J Garcia**4 and **Mark A Ragan**1,2



BMC Bioinformatics 2005, **6:**87doi:10.1186/1471-2105-6-87

### Abstract

#### Background

Computational methods for problem solving need to interleave information access and algorithm execution in a problem-specific workflow. The structures of these workflows are defined by a scaffold of syntactic, semantic and algebraic objects capable of representing them. Despite the proliferation of GUIs (Graphic User Interfaces) in bioinformatics, only some of them provide workflow capabilities; surprisingly, no meta-analysis of workflow operators and components in bioinformatics has been reported.

#### Results

We present a set of syntactic components and algebraic operators capable of representing analytical workflows in bioinformatics. Iteration, recursion, the use of conditional statements, and management of suspend/resume tasks have traditionally been implemented on an ad hoc basis and hard-coded; by having these operators properly defined it is possible to use and parameterize them as generic re-usable components. To illustrate how these operations can be orchestrated, we **present GPIPE, a prototype graphic pipeline generator for PISE that allows the definition of a pipeline, parameterization of its component methods, and storage of metadata in XML formats. This implementation goes beyond the macro capacities currently in PISE. As the entire analysis protocol is defined in XML, a complete bioinformatic experiment (linked sets of methods, parameters and results) can be reproduced or shared among users.** Availability: <http://if-web1.imb.uq.edu.au/Pise/5.a/gpipe.html> [webcite](http://www.webcitation.org/query.php?url=http://if-web1.imb.uq.edu.au/Pise/5.a/gpipe.html&refdoi=10.1186/1471-2105-6-87) (interactive), <ftp://ftp.pasteur.fr/pub/GenSoft/unix/misc/Pise/> [webcite](http://www.webcitation.org/query.php?url=ftp://ftp.pasteur.fr/pub/GenSoft/unix/misc/Pise/&refdoi=10.1186/1471-2105-6-87) (download).

#### Conclusion

**From our meta-analysis we have identified syntactic structures and algebraic operators common to many workflows in bioinformatics. The workflow components and algebraic operators can be assimilated into re-usable software components. GPIPE, a prototype implementation of this framework, provides a GUI builder to facilitate the generation of workflows and integration of heterogeneous analytical tools.**

# Genome wiki

The idea of a genome wiki was articulated by Salzberg SL.  [Genome re-annotation: a wiki solution?](http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=17274839)  Genome Biol. 2007;8(1):102.

Here, it's meant in the sense of a web-based [genome browser](http://biowiki.org/GenomeBrowser) to which annotation tracks can be uploaded, persistently, so that they can be shared with other users.

This page presents a single point of view on what a "genome wiki" should be, acknowledging that the term is ambiguous and the concept somewhat flexible.

## Ideal properties

A wiki would allow the community of experts to work out the best name for each gene, to indicate uncertainty where appropriate and to discuss alternative annotations. - [Salzberg](http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve&db=pubmed&dopt=Abstract&list_uids=17274839)

An ideal genome wiki should have the following

1. **core wiki functionality**: a [wiki-wiki](http://en.wikipedia.org/wiki/Wiki), [web 2.0](http://en.wikipedia.org/wiki/Web_2.0) sort of feel, empowering community collaboration
   * powerful search
   * ability to upload, share, discuss, tag & edit track annotations
   * management of user accounts
   * social-networking tools: who else is working on, or near, this genome?
2. **bioinformatic granularity**: the UI should offer per-feature operations (e.g. on individual genes or exons) as well as per-track or per-genome operations:
   * simple, fast, fluid, responsive, real-time [genome browser](http://biowiki.org/GenomeBrowser) interface
   * editing of features from within the genome browser; track & feature merge operations
   * browsing of feature & track revision histories
   * appropriate consortium-oriented access controls (privacy, sharing, approval)
3. **robust database** properties
   * scalable
   * **ACID** [(Wikipedia)](http://en.wikipedia.org/wiki/ACID)
   * **distributed database** [(Wikipedia)](http://en.wikipedia.org/wiki/distributed%20database)
4. **portability**
   * open source
   * client works in any web browser
   * server is agnostic to hardware platform
   * well-documented (for users and developers)
5. **compatibility**
   * compatible w/standard web apps & protocols (e.g. PageRank, RSS, OpenID, Wikipedia, deep search, semantic web...)
   * compatible w/standard bioinformatics formats (GFF, BED, WIG, MIAME, etc.)
   * close integration with established databases (UCSC, Ensembl, Wikipedia) and terminologies (GO, InterPro, etc.)

## Pragmatic approximations

The simple JBrowse [TWikiPlugin](http://biowiki.org/view/JBrowse/TWikiPlugin) makes a game effort at being a wiki with genome-browsable attachments, but takes only baby steps toward bioinformatic granularity (an attachment is essentially a track, and you cannot edit or operate on individual features, except using an external text editor) or database infrastructure (it's filesystem-based, though it does use some cool dataset-indexing structures in the jbrowse part).

[Genboree](http://genboree.org) scores well on most points; much more evolved than the simple jbrowse plugin in terms of transactional & user infrastructure, though arguably less fluid than jbrowse's minimalistic interface, and perhaps less portable (in terms of being unusual or difficult to install on a new system, since it appears designed to run mostly from one central site; not so much a disadvantage as a different operations model, c.f. blogger vs wordpress).

# Genome re-annotation: a wiki solution?

**Steven L Salzberg**

**2007**

A relatively new model of sharing expertise through the Internet might offer a solution. This model is the 'wiki': a shared resource that anyone can edit. This open-editing framework for websites and data was first introduced in 1995, and it was initially viewed with skepticism by many in the Internet community, who argued that wiki-based websites would be filled with unreliable, inaccurate information. But the success of the online encyclopedia Wikipedia [[14](http://genomebiology.com/2007/8/1/102#B14)] has demonstrated that, despite the skeptics, a wiki site can be accurate, up-to-date and incredibly useful. Genome annotation has many of the same features of an encyclopedia: the information required to produce it is broad-based and the expertise is scattered around the scientific community in a very wide range of laboratories, most of whom are not connected to genome projects. I therefore propose that a 'genome wiki' might provide just the solution we need for genome annotation. A wiki would allow the community of experts to work out the best name for each gene, to indicate uncertainty where appropriate and to discuss alternative annotations. Although wikis will not (and should not) supplant well-curated model-organism databases, for the majority of species they might represent our best chance for creating accurate, up-to-date genome annotation.

Whether or not a genome wiki emerges, we will probably need an archival repository of annotation for many years to come. The international database consortium represented by GenBank, EMBL and DDBJ has served that purpose remarkably well for a long time and will continue to do so. Despite this success, the genomics community needs an accurate, continually updated source of genome annotation for every species, and we can hope that a solution to this problem will emerge in the near future.

Aqwa may make use of existing or novel syntactic structures and algebraic operators for describing bioinformatics workflows (Garcia et al., 2005) to achieve fully customizable workflows with forks, conditional statements, loops and treelike workflows composed of multiple workflows linked together with logical commands. The use of JSON (JavaScript Object Notation) as the data interchange format makes Aqwa easier to extend to complex workflow patterns. This is because, unlike XML, JSON-encoded objects need no further parsing to define them at the object level. This greatly facilitates chaining together objects in complex interdependencies. Following the evaluation of existing workflow control methods and their applicability to the Aqwa system, a core set of operators will be implemented in Phase 3 (Aqwa gamma).

**workflows**

Luciano A. Digiampietri

**Abstract:**

The proliferation of bioinformatics activities brings new challenges

– how to understand and organise these resources, how to exchange and

reuse successful experimental procedures, and to provide interoperability

among data and tools. This paper describes an effort toward these directions.

It is based on combining research on ontology management, AI and

scientific workflows to design, reuse and annotate bioinformatics experiments.

The resulting framework supports automatic or interactive composition of tasks

based on AI planning techniques and takes advantage of ontologies to support

the specification and annotation of bioinformatics workflows. We validate our

proposal with a prototype running on real data.

…

**We use domain ontologies as the basis for attacking these problems. The first**

**question – provenance of data and software tools – directly affects the acceptance of the**

**results of experiments. The quality of bioinformatics experiments depends on properly**

**identifying data origins and the processes that produced these data (Buttler et al., 2002).**

**At most times, provenance is indicated by laborious manual annotations, which often**

**vary across laboratories.**

The second issue concerns tool/task composition while constructing the workflows

(Cavalcanti et al., 2005; Yu and Buyya, 2005; Medeiros et al., 2005). We highlight three

kinds of composition: manual (supervised), iterative (using top-down design practices)

and automatic. In a scenario where several software tools are being made available on the

web, the composition problem has become more important. **To help this issue, many tools**

**invoked by such workflows are now encapsulated into web services**

**Check out**[**http://www.dossier-andreas.net/software\_architecture/mvc.html**](http://www.dossier-andreas.net/software_architecture/mvc.html)

**Garcia 2005 GPIPE**

***Workflow generation, an implementation***

We have developed GPIPE, a flexible workflow generator

for PISE. GPIPE extends the capabilities of PISE to allow

the creation and sharing of customised, reusable and shareable

analytical workflows. So far we have implemented

and tested GPIPE over only the EMBOSS package,

although extension to other algorithmic implementations

is possible where there is an XML file describing the command-

line user interface.

Workflows automate businesses procedures in which

information or tasks are passed between conforming entities

according to a defined set of rules; some of these business

rules are defined by the user, and in our

implementation are managed *via* GPIPE. For our purposes,

the conforming entities are analytical methods

(Clustal, Protpars, etc.). Syntactic rules drive the interaction

between these entities (*e.g.* to ensure syntactic coherence

between heterogeneous file formats). GPIPE also

assures the execution of the workflow, and makes it possible

to distribute different jobs over a grid of servers. GPIPE

addresses these requirements using mostly Bioperl.

In GPIPE, each analysis protocol (including any annotations,

*i.e.* meta-data) is defined within an XML file. A Java

applet provides the user with an exploratory tool for

browsing and displaying methods and protocols. Synchronization

is maintained between client-side display

and server-side storage using Javascript. Server-side

persistence is maintained through serialized Perl objects

that manage the workflow execution. GPIPE supports

independent branched tasks in parallel, and reports errors

and results into an HTML file. The user selects the

methods, sets parameters, defines the chaining of differ- ent methods, and selects the server(s) on which these will

be executed. GPIPE creates an XML file and a Perl script,

each of which describes the experiment. The Perl file may

later be used on a command-line basis, and customized to

address specific needs. The user can monitor the status of

workflow execution, and access intermediary results. A

workflow built with GPIPE can distribute its analyses onto

different, geographically disperse GPIPE/PISE servers.

**Later add web services to workflows**

**Workflows in which concrete executions take place**

**over a grid of web services involve basically the same syntactic**

**structure and algebraic operators; however, a clear**

**business logic needs to be defined beforehand for those**

**web services in order to deepen the integration beyond**

**simply the fact of remote execution.** A higher level of

sophistication for the pipe component as well as for the

conditional operator may be needed, since remote execution

requires (for example) assessment and availability of

the service for the job to be successfully dispatched and

processed.

For our implementation we use two agents,

one on the client side and the other on the server side,

with the queue handled by PBS (Portable Batch System).

It is possible to add a semantic layer, thereby allowing

conceptual selection of the transformers; clear separation

between the operational domain and the knowledge domain be would then be achieved naturally.

**Workflow syntax/semantics**

These may be grouped as equivalent,

but are syntactically different. Selection should be left in

the hands of the user, but the system should at least

inform about this similarity.

Despite agreement on the importance of semantic layers

for integrative systems, such a level of sophistication is far

from being achieved. Lack of awareness of the practical

applications of such technologies is well illustrated with a

traditional and well-studied product: Microsoft Word®.

With Word, syntactic verification can take place as the user

composes text, but no semantic corroboration is done. For

two words like "purpose" and "propose", Word advises on

syntactic issues, but gives no guidance concerning the context

of the words. Semantic issues in bioinformatic workflows

are more complex, and it is not clear if existing

technologies can effectively overcome these problems.

More info reduces mistakes

Semantic and syntactic problems were also identified

from the case workflow described in Figure 4. There, we

saw that to support the extraction of meaningful information

and its presentation to the user, formats should be

ontology-based and machine-readable, *e.g.* in XML format.

Lack of these functional features makes

manipulation of the output a difficult task that is usually

addressed by use of parsers specific to each individual

case. For workflow development, human readability can

be just as important. Consider, for example, a ClustalW

output where valid elements could be identified by the

machine and presented to the user together with contextual

menus including different options over the different

data types. In this way the user would be able to decide

what to do next, where to split a workflow, and over

which part of the output to continue or extend the analysis.

Inclusion of this functionality would allow the workflow

to become more concretely defined as it is used.

**Failure management - May require writing wrappers for applications**

Failure management is an area in which we can see a clear

difference between the business world and bioinformatics.

In the former, processes rarely take longer than an

hour and are not so computationally intensive, whereas in

bioinformatics, processes tend to be computationally

intensive and may take weeks or months to complete.

How failures can be managed to minimize losses will

clearly differ between the two domains. Due to the immaturity

of both web services and workflows in bioinformatics,

it is still in most cases more practical to hard-code

analytical processes. Improved failure management is one

of the domain-specific challenges that faces the application

of workflows in bioinformatics.

**Loops, conditional branching**

Among these types of requirements, we focus our analysis

only on those closely related to workflow design issues,

more specifically (a) the piping of data, (b) the availability

of conditional statements, (c) the need to iterate one

method over a set of different inputs, (d) the possibility of

recursion over a parameter space for a method, (e) and the

need for stop/break management. Algebraic operators can

accurately capture the meaning of these functional

requirements. To describe an analytical workflow, it is

necessary to consider both algebraic operators and syntactic

components. In Table 1 we present the definition of

those algebraic operators we propose, and in Figure 2 we

illustrate how these operators and syntactic elements

together can describe an analytical workflow.



**F**Sy**ig**nt**u**a**r**c**e**ti c**1** components describing bioinformatic analysis workflows

Syntactic components describing bioinformatic analysis workflows.



**F**y**ig**nt**u**a**r**c**e**ti c**2** components and algebraic operators

Syntactic components and algebraic operators.



P**F**h**ig**y**u**lo**r**g**e**e n**3**etic analysis workflow

Phylogenetic analysis workflow.

So far we have intentionally referred to GUIs and workflows

as more-or-less independent. A glimpse into the corresponding

metadata reveals that GUIs are themselves

components of workflow systems. In the bioinformatics

domain this relationship is particularly attractive, since

algebraic operations are usually highly nested.

The interface

system should therefore provide a programming

environment for non-programmers. The language as such

is not complex, but makes extensive use of statements

such as *while...do*, *if...then...else*, and *for...each*. The representation

should be natural to the researcher, separating

the knowledge domain from the operational domain.

**BioMOBY & Taverna**

Automatic service discovery is an intrinsic feature of web

services. The accuracy of the discovery process necessarily

depends on the ontology supporting this service. Systems

such as BioMoby and TAVERNA make extensive use of

service discovery; however, due to the difficulty in describing

biological data types, service discovery is not yet accurate.

It is not yet clear whether languages such as OWL can

be developed to describe relations between biological

concepts with the required accuracy. Integrating information

is as much a syntactic as a semantic problem, and in

bioinformatics these boundaries are particularly illdefined.

# Bioinformatics process management: information flow via a computational journal

**Lance Feagan**\* , **Justin Rohrer**\* , **Alexander Garrett**\* , **Heather Amthauer**\* , **Ed Komp**\* , **David Johnson**\* , **Adam Hock**\* , **Terry Clark**\* , **Gerald Lushington**\* , **Gary Minden**\* and **Victor Frost**\*



Information and Telecommunication Technology Center, University of Kansas, Lawrence, Kansas, USA

author email corresponding author email\* Contributed equally



Source Code for Biology and Medicine 2007, **2:**9doi:10.1186/1751-0473-2-9

The electronic version of this article is the complete one and can be found online at: <http://www.scfbm.org/content/2/1/9>

|  |  |
| --- | --- |
| Received: | 21 January 2007 |
| Accepted: | 3 December 2007 |
| Published: | 3 December 2007 |

© 2007 Feagan et al; licensee BioMed Central Ltd.  
This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/2.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

### Abstract

This paper presents the Bioinformatics Computational Journal (BCJ), a framework for conducting and managing computational experiments in bioinformatics and computational biology. These experiments often involve series of computations, data searches, filters, and annotations which can benefit from a structured environment. Systems to manage computational experiments exist, ranging from libraries with standard data models to elaborate schemes to chain together input and output between applications. Yet, although such frameworks are available, their use is not widespread–ad hoc scripts are often required to bind applications together. **The BCJ explores another solution to this problem through a computer based environment suitable for on-site use, which builds on the traditional laboratory notebook paradigm. It provides an intuitive, extensible paradigm designed for expressive composition of applications. Extensive features facilitate sharing data, computational methods, and entire experiments. By focusing on the bioinformatics and computational biology domain, the scope of the computational framework was narrowed, permitting us to implement a capable set of features for this domain. This report discusses the features determined critical by our system and other projects, along with design issues. We illustrate the use of our implementation of the BCJ on two domain-specific examples**.

The BCJ is designed around characteristics of bioinformatics

research, which often involves: (1) stepwise application

of a series of methods, (2) re-application of the series

of methods in search and retry iterations using different

parameters, (3) development of new procedures and combinations

in the rapidly growing discipline, (4) an open

environment crucially dependent on web-based application

servers and public databases, and (5) rapidly growing

data collections and evolving vocabularies. The volume of

data and the number of computational experiments create

significant processing and management complications.

The BCJ framework reported here addresses these problems

through data and process management, with mechanisms

for experiment specification and composing

computations, tracking data provenance and retaining the

context of computations. Activities such as entering and

locating data, applications, and results are facilitated by

the framework. The BCJ fosters collaboration among

project teams through structured privacy mechanisms and

naming constructs, increasing productivity in computational

experiments for both computationally expert and non-expert researchers.

**From the user perspective, the top-level structure of the**

**computational journal is a collection of** journal entries

and relationships among these entries. More formally, an

*entry* is the basic unit of information provided by the end

user for the BCJ. Each entry is characterized by a single *content*

*type* that identifies the format of this information. The

number of content types is user extensible, allowing the

BCJ to support new data formats and programs. Relations

among entries provide an organizational structure to enable

tracking entry dependencies, e.g., entry A "is an execution

input" for entry B. These relationships are defined

automatically as the entries are created and then they are maintained by the BCJ environment.

From the user perspective, the BCJ provides a mechanism

to navigate among the potentially large number of entries.

Each navigator provides a virtual organization of the

entries, based on one or more relations among the entries.

This allows the user to see the data organized to match a

specific activity, by selecting the navigator most appropriate

for the task. At creation time, each entry is assigned a

parent journal. This provides an immediate and natural

hierarchical view of all entries. This journal relationship

provides a view of the entries that closely resembles the

file directory hierarchy familiar to all computer users. This

is the most basic of the BCJ navigators. Graphically, it uses

a tree-list display that is widely used on computers to

present such hierarchies, but with annotations that provide

more detail than a basic hierarchical file system interface.

When an entry is selected, a plug-in associated with the

entry's content type is selected to edit or view the content.

Graphical editors are provided for workflow definitions

and experiment definitions. When a new content type is

defined in the BCJ environment, it is natural to incorporate

Some data formats, particularly binary data formats, are

supported by sophisticated and optimized viewer programs.

For example, three-dimensional graphical viewers

are available for molecular dynamics simulation outputs.

It is not practical to convert these programs to operate

directly inside the framework. The BCJ environment provides

an interface that allows a user to associate an external

viewer, a program installed on his/her client machine,

as the default viewer for a specified content type. Whenever

an entry of one of these content types is selected for

viewing, the BCJ environment automatically generates a

local copy of the data on the client machine and invokes

the associated program with this input.

In addition to editors and navigators, the BCJ environment

provides a variety of additional views that present

auxiliary information to support these primary tasks, such

as the Queue Manager view, which displays the progress

of currently executing experiments.

Precise descriptions of BCJ programs are kept in XML documents.

This description is called a *resource definition* in

the BCJ environment. The Document Type Definition

(DTD) for the XML data can be found in [10]. The BCJ

environment currently contains about sixty programs/

resource definitions (see Table 5). Many resource definitions

were derived from the XML descriptions for the corresponding

programs in PISE [11].

The graphical user interface paradigm for specifying the

experiment design is central to the usability of the environment.

Ease of use is achieved with an intuitive and

consistent interface to applications, data, and results.

**Experiment-management issues including archiving,**

**searching, modifying and retrieving existing workflows,**

**searching stored data, searching annotated workflows/**

**experiments, and the ability to export data to and import**

**data from external sources impact the usability of the**

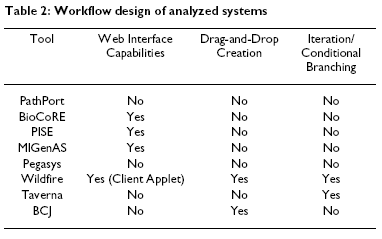
**environment.**

Finally, other miscellaneous features, such

as **access controls on data, data provenance, experiment**

**repeatability, collaboration, and annotation of experiments are important.**

**Feagan et a. 2007 Table 2**



**Design and implementation of the BCJ**

***Abstract data interface***

There has been a proliferation of specialized tools, interfaces,

and *ad hoc* scripts to handle specific, repetitive tasks

in narrow domains. **The generality of these tools is sometimes**

**limited by implicit data formatting, organization,**

**and naming conventions.** The presence of these low-level

details can impede a system's usefulness. **In the BCJ environment,**

**the interface has been designed to shield the**

**user from these details; every data element is simply an**

**entry, regardless of its specific content type, providing a**

**uniform, high-level abstraction to the user.**

**Users are insulated**

**from tedious low-level abstractions such as files systems**

**and their computational hardware resources.**

**The BCJ environment maintains every entry created by**

**every user. An entry can be a bioinformatics application, a**

**workflow, an experiment, or a data set. The underlying**

**framework collects and maintains a set of meta-data for**

**each entry, including its name, author, creation and modification**

**dates, and content type.**

**Every application that plugs into the BCJ environment is an entry and is accessible**

**through the standard entry interface. Many BCJ functions**

**manipulate only meta-data fields of an entry, other**

**BCJ functions are designed to manipulate only a specific**

**content type, and are enabled only when an entry containing**

**this content type is selected.**

The BCJ environment provides a variety of *navigators*.

These are functions that provide a hierarchical view of a

collection of entries. Navigators establish a virtual organization

of the entries. This allows the user to see the data

organized to match a specific activity, by selecting the navigator

most appropriate for the task. The most basic navigator,

the Journal navigator, displays entries according to

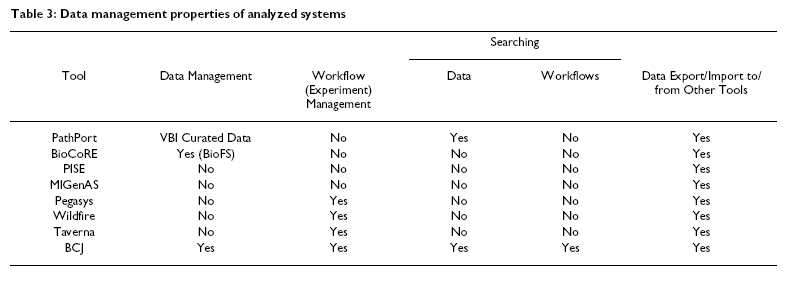
the journal in which entries are created. This navigator

provides a view of entries, similar to a common file hierarchy

with journals corresponding to file directories.

Other navigators display entries related to a specific entry.

**Collaboration**



*Additional Features*

Of the systems reviewed, only the BCJ and Taverna provided

the ability to track the provenance of data created

within it and provided the ability to repeat an experiment

under the same conditions. Note that for the BCJ to support

experiment repeatability, program versioning needs

to be implemented. This provides the ability to re-run

computational experiments. The BCJ and BioCoRE

encrypt information exchanges between the client and the

server. Additionally, the collaborative features in several

reviewed systems consisted of data sharing by exporting

data/workflows to a shared file and sharing the file

directly. Collaboration at the system level is offered by the

BCJ. BioCoRE offered many collaborative features, including

message boards and chatting, while the BCJ natively

handled collaboration through the sharing of data, workflows,

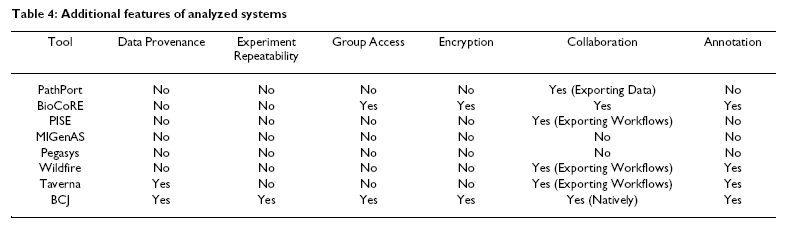
and experiment journals. Annotation of data and

experiments, like in BioCoRE, Wildfire, Taverna, and the

BCJ, is a valuable feature and is an integral part of workflow

and data management. Table 4 summarizes these features.

Feagan 2007 Table 4



The BCJ environment provides a variety of *navigators*.

These are functions that provide a hierarchical view of a

collection of entries. Navigators establish a virtual organization

of the entries. This allows the user to see the data

organized to match a specific activity, by selecting the navigator

most appropriate for the task. The most basic navigator,

the Journal navigator, displays entries according to

the journal in which entries are created. This navigator

provides a view of entries, similar to a common file hierarchy

with journals corresponding to file directories.

Other navigators display entries related to a specific entry.

For example, the dependency navigator displays entries

that depend on the selected entry.

The BCJ can be extended through the addition of navigators.

As new tools and techniques are incorporated into

the environment, it is likely that some will implement

novel data organization schemes requiring a new navigator.

Shared access to all entries must be balanced with the

individual's need for security and privacy. The BCJ security

model only allows the creator of an entry to modify it.

This simple model is consistent with the larger computational

journal paradigm. The journal is primarily a record

of experimental steps. Once completed, a step may no

longer be modified, however, it may be followed by additional

experimental steps to refine results and investigate

variations on the base experiment. This characteristic aids

in maintaining the *replication recipe* provenance. If another

user wants to make changes to data or an experiment created

by another user, the user may create a new entry as a

copy of the original one using the SaveAs operation,

thereby maintaining data pedigree.

**UNIVERSAL DATABASE SCHEMA – NOT REQUIRED**

Designing a database schema broad enough to handle all

forms of biological data can be a daunting undertaking.

Specialized systems such as the Genomic Unified Schema

(GUS) are designed for this [9]. For the BCJ role in performing

and recording computational experiments, this

generality is not a critical requirement.

**cONCLUSION**

Bioinformatics research involves computational processing

of multi faceted biological data at scales from atomic

level to cellular. Sophisticated environments are needed

to support the diverse and expanding range of computational

methods, scales, models, and data. The inherently

collaborative work requires managing data and methods

between colleagues and public-domain databases. Complex

experiments are often involved, which need to be

checked, altered, and documented.

The Bioinformatics

Computational Journal (BCJ) reported here provides support

for these activities. The BCJ handles data management

and process specification in a single framework,

demonstrated here with two representative bioinformatics

problems.

The system was founded on Eclipse, providing

a rigorously tested and maintained software

infrastructure.

**The user paradigm was designed to resemble**

**a laboratory notebook, inline with the target users. We**

**found this paradigm a natural basis for describing computational**

**experiments. Features are provided in the BCJ to**

**authenticate and establish pedigree of experiments. This is**

**a crucial feature toward verifying and documenting computational**

**results.**

Future efforts include adding iteration and branching functionality and enabling for grid collaborations.

#### Comparisons

Four aspects were used to analyze the environments discussed above: execution environment, workflow design, data management, and additional features. A comparison based on these aspects with other computations is discussed below. (See the appendix for introductions to the systems discussed in this section: PathPort, Pegasys, PISE, BioCoRE, MIGenAS, Taverna, and Wildfire.)

#### Execution

Most surveyed solutions were designed to be run locally or on the grid–few were designed for cluster-based systems. Neither PathPort nor PISE were implemented for cluster-based execution. MIGenAS was written to be run on only one specific cluster. BioCoRE, Pegasys, Wildfire, Taverna, and BCJ allowed execution on a local cluster. Only Pegasys, Wildfire, Taverna, and the BCJ offer workflow capabilities. PISE and MIGenAS allow for linking program input and output, but neither goes beyond a serial pipeline. Furthermore, BioCoRE, MIGenAS, and Wildfire do not offer the ability to add tools to the server's execution environment, while only PathPort, Wildfire, and the BCJ allow client-side, external tools to be added. Only the BCJ and PathPort allow for both internal and external extensibility. Table [1](http://www.scfbm.org/content/2/1/9/table/T1) summarizes the execution capabilities of the environments.

[**Table 1.**](http://www.scfbm.org/content/2/1/9/table/T1) Execution capabilities of analyzed systems

#### Workflow Design

All of the surveyed solutions provide a GUI, however only Pegasys, Wildfire, and the BCJ provide an interactive, drag-and-drop interface. Unlike the BCJ, all surveyed solutions use input forms that are loosely coupled to the workflow on a separate page from the workflow specification. The input of parameters for each element in the BCJ is closely coupled to the associated workflow, as is discussed below. Wildfire and Taverna provide iterations and branching based on user specifications. Wildfire iterations are essentially while loops based on user-supplied imperative programs; Taverna iterations are defined graphically. Table [2](http://www.scfbm.org/content/2/1/9/table/T2) summarizes the workflow design capabilities of these environments.

[**Table 2.**](http://www.scfbm.org/content/2/1/9/table/T2) Workflow design of analyzed systems

#### Data Management

PathPort, BioCoRE, and the BCJ are the only environments that provide data management. Additionally, only Pegasys, Wildfire, Taverna, and the BCJ provide workflow management. The combination of workflow and data management increases the usefulness of the environment. Table [3](http://www.scfbm.org/content/2/1/9/table/T3) summarizes the data management capabilities.

[**Table 3.**](http://www.scfbm.org/content/2/1/9/table/T3) Data management properties of analyzed systems

#### Additional Features

Of the systems reviewed, only the BCJ and Taverna provided the ability to track the provenance of data created within it and provided the ability to repeat an experiment under the same conditions. Note that for the BCJ to support experiment repeatability, program versioning needs to be implemented. This provides the ability to re-run computational experiments. The BCJ and BioCoRE encrypt information exchanges between the client and the server. Additionally, the collaborative features in several reviewed systems consisted of data sharing by exporting data/workflows to a shared file and sharing the file directly. Collaboration at the system level is offered by the BCJ. BioCoRE offered many collaborative features, including message boards and chatting, while the BCJ natively handled collaboration through the sharing of data, workflows, and experiment journals. Annotation of data and experiments, like in BioCoRE, Wildfire, Taverna, and the BCJ, is a valuable feature and is an integral part of workflow and data management. Table [4](http://www.scfbm.org/content/2/1/9/table/T4) summarizes these features.

[**Table 4.**](http://www.scfbm.org/content/2/1/9/table/T4) Additional features of analyzed systems

### Conclusion

Bioinformatics research involves computational processing of multi faceted biological data at scales from atomic level to cellular. Sophisticated environments are needed to support the diverse and expanding range of computational methods, scales, models, and data. The inherently collaborative work requires managing data and methods between colleagues and public-domain databases. Complex experiments are often involved, which need to be checked, altered, and documented. The Bioinformatics Computational Journal (BCJ) reported here provides support for these activities. The BCJ handles data management and process specification in a single framework, demonstrated here with two representative bioinformatics problems. The system was founded on Eclipse, providing a rigorously tested and maintained software infrastructure. The user paradigm was designed to resemble a laboratory notebook, inline with the target users. We found this paradigm a natural basis for describing computational experiments. Features are provided in the BCJ to authenticate and establish pedigree of experiments. This is a crucial feature toward verifying and documenting computational results. Future efforts include adding iteration and branching functionality and enabling for grid collaborations.

here is thus a clear need for a framework that will enable protocol-based bioinformatics analysis to be not only describable on paper but easily replicable and distributable as well. We argue that this creates greater transparency, encourages good scientific practices, and increases the rigorousness of bioinformatics analysis.

**The Approach:** *How will you attempt to explore or solve the stated research problem? How will your rationale and/or approach overcome existing challenges or barriers in the field? If you propose to improve existing technologies or to develop new technologies, which needs are being addressed and what is unconventional and exceptionally innovative about your approach? Provide enough information for reviewers to determine what you are proposing to do, but do not include a detailed experimental plan.*

**Requirements**

1. Low barriers to usage – easy access, user friendly interface
2. Workflows
   1. Predefined workflows (transcriptome, SNP verification, gene networks, file conversion utilities, ID conversion utilities, etc.)
   2. Customizable workflows
   3. Persistent data and workflow configurations
3. Customizable report extraction from workflow output
4. Customizable genomic views of report data with integrated genomic features
5. Sharing
   1. User-defined groups with customizable permissions
   2. Workflow, report and view sharing among groups
6. Import external biological data and genomic features into workflow
7. Integration with external software (e.g., Cytoscape, R, GeneSet Analyzer)
8. Programmatic remote access (API and Web Service)

View – Interface design

Model – Data design (API, SQL tables)

Controller –

Model-View-Controller



This architecture is used in simple GUI applications. The architecture is event-driven, which means that all activity starts by some event and is propagated by some other events.  
The architecture usually contains a large number of components (hereafter called MVC components), each of which is built out of these three items:

1. The Model
2. The View
3. The Controller

The **model** is the information contents of the component. It can be as simple as an integer or a string, and as complex as an interface to an external database. The MVC component does not need to contain the model itself. It may contain just a link to the model. And this is common if multiple MVC components are connected to a model.  
In a MVC architecture, where the model is shared by multiple MVC components, the model should keep track of any number of views. These views need be notified of changes that occur in the model.

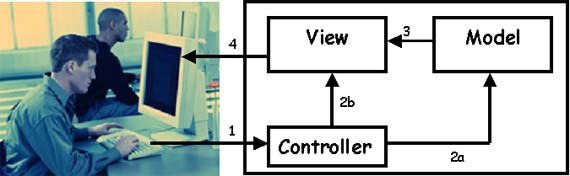
The **view** is the visual representation of the component, usually on a computer screen. It can be a textbox or a button in a form on the screen, anything that represents the model or part of it. Try to think of the view as *only* the visual aspect of the component. I.e. just the picture of the textbox that can in itself not be manipulated by a keypress or mouseclick.  
The view also keeps track of the state of the component itself. Which means that it keeps track of the cursor position in a textbox, the position of a scrollbar, the state of a button, etc. This way it not only knows *what* to show (the model) but also *how* to show it.

The **controller** takes care of all the action. It is sent external events like keypresses, mouseclicks, mousemoves, timer updates, etc. It knows how to change the model in response to these events and it it also knows how to change the component's state in the view. If one for example presses a key when a textbox is active, the controller tells the model to add the character to the string and it tells the view to advance the cursor position.

An MVC component is created in these steps

1. The model can exist prior to the creation of the component, if so, it is included in the component merely by reference
2. The view is created, to which a reference to the model is passed
3. The view registers itself with the model as a changelistener
4. The view creates the controller, which is passed the reference to the model, and a reference to the view

The control flow within an MVC component.



The controller receives an event, say a keypress from the user (1). Depending on its own logic, it processes the event, and updates the model (2a) or only the characteristics of the view (2b). The model notifies the view(s) (3). The view updates itself as can be seen by the user (4).

**Use cases**

**Feagan 2007**

**From the user perspective, the top-level structure of the**

**computational journal is a collection of** journal entries

and relationships among these entries. More formally, an

*entry* is the basic unit of information provided by the end

user for the BCJ. Each entry is characterized by a single *content*

*type* that identifies the format of this information. The

number of content types is user extensible, allowing the

BCJ to support new data formats and programs. Relations

among entries provide an organizational structure to enable

tracking entry dependencies, e.g., entry A "is an execution

input" for entry B. These relationships are defined

automatically as the entries are created and then they are maintained by the BCJ environment.

From the user perspective, the BCJ provides a mechanism

to navigate among the potentially large number of entries.

Each navigator provides a virtual organization of the

entries, based on one or more relations among the entries.

This allows the user to see the data organized to match a

specific activity, by selecting the navigator most appropriate

for the task. At creation time, each entry is assigned a

parent journal. This provides an immediate and natural

hierarchical view of all entries. This journal relationship

provides a view of the entries that closely resembles the

file directory hierarchy familiar to all computer users. This

is the most basic of the BCJ navigators. Graphically, it uses

a tree-list display that is widely used on computers to

present such hierarchies, but with annotations that provide

more detail than a basic hierarchical file system interface.

When an entry is selected, a plug-in associated with the

entry's content type is selected to edit or view the content.

Graphical editors are provided for workflow definitions

and experiment definitions. When a new content type is

defined in the BCJ environment, it is natural to incorporate

Some data formats, particularly binary data formats, are

supported by sophisticated and optimized viewer programs.

For example, three-dimensional graphical viewers

are available for molecular dynamics simulation outputs.

It is not practical to convert these programs to operate

directly inside the framework. The BCJ environment provides

an interface that allows a user to associate an external

viewer, a program installed on his/her client machine,

as the default viewer for a specified content type. Whenever

an entry of one of these content types is selected for

viewing, the BCJ environment automatically generates a

local copy of the data on the client machine and invokes

the associated program with this input.

In addition to editors and navigators, the BCJ environment

provides a variety of additional views that present

auxiliary information to support these primary tasks, such

as the Queue Manager view, which displays the progress

of currently executing experiments.

Precise descriptions of BCJ programs are kept in XML documents.

This description is called a *resource definition* in

the BCJ environment. The Document Type Definition

(DTD) for the XML data can be found in [10]. The BCJ

environment currently contains about sixty programs/

resource definitions (see Table 5). Many resource definitions

were derived from the XML descriptions for the corresponding

programs in PISE [11].

The graphical user interface paradigm for specifying the

experiment design is central to the usability of the environment.

Ease of use is achieved with an intuitive and

consistent interface to applications, data, and results.

**Experiment-management issues including archiving,**

**searching, modifying and retrieving existing workflows,**

**searching stored data, searching annotated workflows/**

**experiments, and the ability to export data to and import**

**data from external sources impact the usability of the**

**environment.**

Finally, other miscellaneous features, such

as **access controls on data, data provenance, experiment**

**repeatability, collaboration, and annotation of experiments are important.**

**Software design strategy**

1. Human-centered development process
2. Classification of bioinformatics tasks
3. General query and workflow requirements
4. Usability analysis (e.g., User tests)(30)
5. Project-based structure

**Web-based**

* **Flexible, mature technology (multi-media)**
* **Mature GUI tools (DHTML, AJAX)**
* The three major genome viewers – UCSC Genome Browser (REF), Ensembl genome browser (REF) and GBrowse (REF) – are based on the traditional client-server model where user page clicks result in a reloaded image file sent from the server. Java-based applications such as Apollo (REF) lack a concerted approach to data sharing although newer applications IGV (REF) do allow for limited filtering of the displayed features (see Figure 2).

The Firegoose incorporates Mozilla Firefox into the Gaggle

environment providing coordinated access to web

applications and programmatic data sources. Performing

data integration in the browser has several advantages and

is perhaps the most interesting feature of the Firegoose.

Browsers excel at search and navigation. Using the Firegoose,

a biologist can search and navigate web resources

using familiar browser-based interfaces with the additional

capability of easily moving data from one webbased

resource to another as well as between the web and

the desktop. Interactively integrating specific information

as needed replaces the cumbersome process of maintaining

local copies of large databases and manually coercing

data from diverse sources into a compatible format.

Using

the Gaggle data types as intermediaries lowers the barrier

between web resources and desktop tools, allowing the

scientist to creatively combine and re-use data in ways that

go beyond those provided by the curators of individual

data sources.

The Firegoose positions the Gaggle to take advantage of

increasing use of web protocols to transmit structured

data. The Firegoose provides a framework in which new

web resources can be integrated into the Gaggle in a

straightforward and easily implemented manner, accommodating

a variety of protocols. In supporting a number

of protocols, we hope to encourage data providers to

make available structured data in the format of their

choice and to provide the necessary information to link

web interfaces with the underlying data allowing browsing

and programmatic access to become seamlessly integrated.

If the web is becoming a channel for structured data,

applications that share data between diverse web

resources and software tools will be of increasing importance.

The Firegoose aims to fill this role for the systems

biology domain.

JS vs Java

Most of the desktop components of the Gaggle

are deployed as Java webstarts, which can be launched by

clicking a link in the browser.

The toolbar is compatible with versions 1.5.x and 2.0.x of

Mozilla Firefox. We anticipate maintaining compatibility

with Firefox 3.x when released.

Problem with Java

Java version 5 [35] or higher runtime environment is

required and the Java browser plug in for Firefox must be

installed. Extra attention is often required to install the

Java browser plug-in on Linux. Specific instructions for

most distributions are available on the web.

**Perl**

* **Large community, many support groups for novice users**
* **Abundant modules, including Bioinformatics modules**
* **Accessible to Biologists**
* **Great with strings (sequence data)**
* **Good for sysadmin-type tasks**

**Hardware requirements** (Joel)

8-node (8-core each) cluster

HPC cluster time

Production server

Development server

**Timeline and Milestones:** Provide a timeline for the proposed research indicating points where intermediate objectives will be assessed and decisions will be made regarding the course and direction of the continuing research effort. Possible alternative paths that may be followed at critical junctures in the project plan should be described and indicated on the timeline.

Preliminary data are not required but may be included, if necessary to demonstrate the feasibility of the proposed studies. The presentation must be clear and particularly compelling. No detailed scientific plan should be provided, but timelines must be presented.

* **Iteration on both functional and quality requirements**
* **Many stakeholders involved**
* **Balancing of functional and quality requirements**

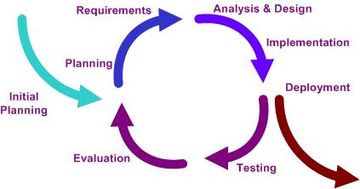
# <http://en.wikipedia.org/wiki/Iterative_and_incremental_development>

# Iterative and incremental development

### From Wikipedia, the free encyclopedia

Jump to: [navigation](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#column-one), [search](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#searchInput)

**Iterative and Incremental development** is a cyclic [software development process](http://en.wikipedia.org/wiki/Software_development_process) developed in response to the weaknesses of the [waterfall model](http://en.wikipedia.org/wiki/Waterfall_model). It starts with an initial planning and ends with deployment with the cyclic interaction in between.



An iterative development model

|  |
| --- |
| [**Software development process**](http://en.wikipedia.org/wiki/Software_development_process) |
| **Activities and steps** |
| [Requirements](http://en.wikipedia.org/wiki/Requirements_analysis)**·** [Specification](http://en.wikipedia.org/wiki/Functional_specification) [Architecture](http://en.wikipedia.org/wiki/Software_architecture)**·** [Design](http://en.wikipedia.org/wiki/Software_design) [Implementation](http://en.wikipedia.org/wiki/Computer_programming)**·** [Testing](http://en.wikipedia.org/wiki/Software_testing) [Deployment](http://en.wikipedia.org/wiki/Software_deployment)**·** [Maintenance](http://en.wikipedia.org/wiki/Software_maintenance) |
| **Models** |
| [Agile](http://en.wikipedia.org/wiki/Agile_software_development)**·** [Cleanroom](http://en.wikipedia.org/wiki/Cleanroom_Software_Engineering)**·** [DSDM](http://en.wikipedia.org/wiki/Dynamic_Systems_Development_Method) **Iterative ·** [RAD](http://en.wikipedia.org/wiki/Rapid_application_development) **·** [RUP](http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process) **·** [Spiral](http://en.wikipedia.org/wiki/Spiral_model) [Waterfall](http://en.wikipedia.org/wiki/Waterfall_model)**·** [XP](http://en.wikipedia.org/wiki/Extreme_Programming)**·** [Scrum](http://en.wikipedia.org/wiki/Scrum_%28development%29) **·** [V-Model](http://en.wikipedia.org/wiki/V-Model_%28software_development%29) [FDD](http://en.wikipedia.org/wiki/Feature_Driven_Development) |
| **Supporting disciplines** |
| [Configuration management](http://en.wikipedia.org/wiki/Software_configuration_management) [Documentation](http://en.wikipedia.org/wiki/Software_documentation) [Quality assurance (SQA)](http://en.wikipedia.org/wiki/Software_quality_assurance) [Project management](http://en.wikipedia.org/wiki/Software_project_management) [User experience design](http://en.wikipedia.org/wiki/User_experience_design) |
| **Tools** |
| [Compiler](http://en.wikipedia.org/wiki/Compiler) **·** [Debugger](http://en.wikipedia.org/wiki/Debugger) **·** [Profiler](http://en.wikipedia.org/wiki/Performance_analysis) [GUI designer](http://en.wikipedia.org/wiki/Graphical_user_interface_builder) [Integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) |
| This box: [view](http://en.wikipedia.org/wiki/Template:Software_development_process) • [talk](http://en.wikipedia.org/wiki/Template_talk:Software_development_process) |

The iterative and incremental development is an essential part of the [Rational Unified Process](http://en.wikipedia.org/wiki/Rational_Unified_Process), the [Dynamic Systems Development Method](http://en.wikipedia.org/wiki/Dynamic_Systems_Development_Method), [Extreme Programming](http://en.wikipedia.org/wiki/Extreme_Programming) and generally the [agile software development](http://en.wikipedia.org/wiki/Agile_software_development) frameworks.

|  |
| --- |
| Contents [[hide](javascript:toggleToc())]   * [1 Overview](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Overview) * [2 Development topics](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Development_topics)   + [2.1 The Basic idea](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#The_Basic_idea)   + [2.2 Iterative development](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Iterative_development)   + [2.3 Waterfall vs. Iterative Development](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Waterfall_vs._Iterative_Development)   + [2.4 Implementation guidelines](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Implementation_guidelines) * [3 See also](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#See_also) * [4 References](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#References) * [5 Further reading](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Further_reading) * [6 External links](http://en.wikipedia.org/wiki/Iterative_and_incremental_development#External_links) |

## [[edit](http://en.wikipedia.org/w/index.php?title=Iterative_and_incremental_development&action=edit&section=1)] Overview

Incremental development is a scheduling and staging strategy, in which the various parts of the system are developed at different times or rates, and integrated as they are completed. It does not imply, require nor preclude iterative development or waterfall development - both of those are rework strategies. The alternative to incremental development is to develop the entire system with a "big bang" integration.

Iterative development is a rework scheduling strategy in which time is set aside to revise and improve parts of the system. It does not presuppose incremental development, but works very well with it. A typical difference is that the output from an increment is not necessarily subject to further refinement, and its testing or user feedback is not used as input for revising the plans or specifications of the successive increments. On the contrary, the output from an iteration is examined for modification, and especially for revising the targets of the successive iterations.[[*clarification needed*](http://en.wikipedia.org/wiki/Wikipedia:Please_clarify)]

The two terms were merged in practical use in the mid-1990s. The authors of the Unified Process (UP) and the [Rational Unified Process](http://en.wikipedia.org/wiki/Rational_Unified_Process) (RUP) selected the term "iterative development", and "iterations" to generally mean any combination of incremental and iterative development. Most people saying "iterative" development mean that they do both incremental and iterative development. Some project teams get into trouble by doing only one and not the other without realizing it.

## [[edit](http://en.wikipedia.org/w/index.php?title=Iterative_and_incremental_development&action=edit&section=2)] Development topics

### [[edit](http://en.wikipedia.org/w/index.php?title=Iterative_and_incremental_development&action=edit&section=3)] The Basic idea

The basic idea behind iterative enhancement is to develop a [software](http://en.wikipedia.org/wiki/Software) system incrementally, allowing the [developer](http://en.wikipedia.org/wiki/Software_developer) to take advantage of what was being learned during the development of earlier, incremental, deliverable versions of the system. Learning comes from both the development and use of the system, where possible. Key steps in the process were to start with a simple implementation of a subset of the software requirements and iteratively enhance the evolving sequence of versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added.

The Procedure itself consists of the Initialization step, the Iteration step, and the Project Control List. The initialization step creates a base version of the system. The goal for this initial implementation is to create a product to which the user can react. It should offer a sampling of the key aspects of the problem and provide a solution that is simple enough to understand and implement easily. **To guide the iteration process, a project control list is created that contains a record of all tasks that need to be performed. It includes such items as new features to be implemented and areas of redesign of the existing solution. The control list is constantly being revised as a result of the analysis phase.**

The iteration involves the redesign and implementation of a task from the project control list, and the analysis of the current version of the system. The goal for the design and implementation of any iteration is to be simple, straightforward, and modular, supporting redesign at that stage or as a task added to the project control list. The level of design detail is not dictated by the interactive approach. In a light-weight iterative project the code may represent the major source of [documentation](http://en.wikipedia.org/wiki/Software_documentation) of the system; however, in a mission-critical iterative project a formal [Software Design Document](http://en.wikipedia.org/wiki/Software_Design_Document) may be used. The analysis of an iteration is based upon user feedback, and the program analysis facilities available. It involves analysis of the structure, modularity, [usability](http://en.wikipedia.org/wiki/Usability), reliability, efficiency, & achievement of goals. The project control list is modified in light of the analysis results.



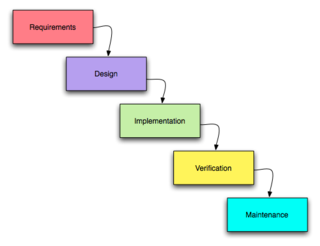
Iterative development.

### [[edit](http://en.wikipedia.org/w/index.php?title=Iterative_and_incremental_development&action=edit&section=4)] Iterative development

Iterative development slices the deliverable business value (system functionality) into iterations. In each iteration a slice of functionality is delivered through cross-discipline work, starting from the model/requirements through to the testing/deployment. The unified process groups iterations into phases: inception, elaboration, construction, and transition.

* Inception identifies project scope, risks, and requirements (functional and non-functional) at a high level but in enough detail that work can be estimated.
* Elaboration delivers a working architecture that mitigates the top risks and fulfills the non-functional requirements.
* Construction incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
* Transition delivers the system into the production operating environment.

Each of the phases may be divided into 1 or more iterations, which are usually time-boxed rather than feature-boxed. Architects and analysts work one iteration ahead of developers and testers to keep their work-product backlog full.



The unmodified "[waterfall model](http://en.wikipedia.org/wiki/Waterfall_model)". Progress flows from the top to the bottom, like a waterfall.

### [[edit](http://en.wikipedia.org/w/index.php?title=Iterative_and_incremental_development&action=edit&section=5)] Waterfall vs. Iterative Development

Waterfall development completes the project-wide work-products of each discipline in a single step before moving on to the next discipline in the next step. Business value is delivered all at once, and only at the very end of the project.

### [[edit](http://en.wikipedia.org/w/index.php?title=Iterative_and_incremental_development&action=edit&section=6)] Implementation guidelines

Guidelines that drive the implementation and analysis include:

* Any difficulty in design, coding and testing a modification should signal the need for redesign or re-coding.
* Modifications should fit easily into isolated and easy-to-find modules. If they do not, some redesign is needed.
* Modifications to tables should be especially easy to make. If any table modification is not quickly and easily done, redesign is indicated.
* Modifications should become easier to make as the iterations progress. If they are not, there is a basic problem such as a design flaw or a proliferation of [patches](http://en.wikipedia.org/wiki/Patch_%28computing%29).
* Patches should normally be allowed to exist for only one or two iterations. Patches may be necessary to avoid redesigning during an implementation phase.
* The existing implementation should be analysed frequently to determine how well it measures up to project goals.
* Program analysis facilities should be used whenever available to aid in the analysis of partial implementations.
* User reaction should be solicited and analysed for indications of deficiencies in the current implementation.

We propose to address the bioinformatics challenges, namely the development of 1) improved NextGen sequence assembly pipelines based on objective evaluations of assemblers/aligners, 2) optimized SNP, transcriptome and genetic interaction pathways, 3) .

Specific Aim 1

**1. Evaluation of next gen assemblers**

We compared several commonly used short read assembly tools and propose a method for reducing these errors by combining different assemblies for the final result.

**METHOD**

Human mtDNA and whole-genome mRNA short reads produced using the Illumina/Solexa Genome Analyzer I platform were used, as well as E. coli, Herpes simplex and bacteriophage PhiX. Seven commercial and open-source short read assemblers were first assessed for assembly capacity in terms of the maximum number of reads that can be effectively assembled using relatively high-end computer hardware. We investigated the performance of Eland (GAPipeline v0.30, Illumina), Velvet v0.7.16 (28), Mira v2.9.25 (21), Genomics Workbench (CLC Bio) v1.2, Seqman NGen (DNAStar) 1.1, NextGene (Soft Genetics) 1.0 and MAQ v 0.6.8 (22) (23). Assemblies produced by the different programs were compared and a consensus determined based on read identity and divergence from the relevant reference sequence. The overall combination of assemblies was viewed for quality control purposes using a sequence viewer that we developed to handle the huge data depth and breadth of sample types.

**RESULTS**Our results indicate that there are significant differences in the capabilities of the different reference and *de novo* short read assembly tools as shown in Figures 1, 2 and 3. The resulting assemblies showed significant differences in read matching against the reference sequence in particular locations. For human data, Seqman NGen, Genomics Workbench and NextGen showed better performance in terms of the number of reads assembled however this may result in less accurate contigs. A combination of different assemblies can provide more reliable estimates of genetic aberrations by flagging dubious assembly regions that are not represented in a majority of the different assemblies. Conversely, regions that are matched identically by a majority of the different algorithms can be accorded greater confidence with regard to their predicted SNPs, indels and breakpoints. Based on this research, we are currently developing a new visualization and analysis tool to meet the needs of next generation sequencing data analysis.

**CONCLUSION**

Effective use of these technologies depends on the correct interpretation of differing assembly results based on a consensus of reads incorporated by the various assembly algorithms. Manual verification of the assembly combination using a Next Generation sequence viewer improved confidence in the resulting assembly and aided comprehension of the strengths and weaknesses of this approach.

**Feasibility (***Discussion of the strengths and weaknesses of the proposed study)*

Goes here…

Risks

At the root of most UML fevers is a lack of practical experience in those individuals responsible for selecting and applying the technologies and processes underlying a program’s software-development efforts. This lack of experience translates into both unrealistic expectation and misapplication of technology, often aggravated by nonexistent or bad software-development processes, a perfect breeding ground for UML fever. If a software organization’s battle against UML fever is to be successful, it is absolutely critical that people with practical experience are in place driving the selection of technologies, as well as developing the processes for their associated usage.

**Aqwa roadmap (***Realistic time line***)**

Goes here…

**06-HG-101\* New computational and statistical methods for the analysis of large data sets from next-generation sequencing technologies. (High priority grant)s**

The introduction of new methods for DNA sequencing has opened new avenues, including large-scale sequencing studies, metagenomics, transcriptomics, genetic network analysis, and determination of the relationship of sequence variation and phenotypes to disease, to address heretofore unapproachable problems in biomedical research. However, since the large amounts (terabases) of data generated overwhelm existing computational resources and analytic methods, urgent action is needed to enable the translation of this rich new source of genomic information into medical benefit. Contact: Dr. Lisa Brooks, 301 496-7531, [brooksl@mail.nih.gov](mailto:brooksl@mail.nih.gov)

(*See end of document for Research Plan structure and format*)

**http://grants.nih.gov/grants/guide/rfa-files/RFA-OD-09-003.htmL**

**Special Instructions for PHS398 Research Plan Component (Section 5.5 of SF424 (R&R) Application)**

**Research Plan:** The Research Plan is comprised of special sections noted below and is limited to a total of **12** pages, including tables, graphs, figures, diagrams, and charts. The Research Plan should be self-contained and uploaded as a single attachment in the Research Designs and Methods item.

**PHS398 Research Plan Component Sections**

|  |  |
| --- | --- |
| **Item Number and Title** | **Instructions** |
| 1. Introduction to Application | Omit (N/A: Resubmissions and Revisions not allowable) |
| 2. Specific Aims | One page maximum. Separate PDF attachment |
| 3. Background and Significance | Omit |
| 4. Preliminary Studies/Progress Report | Omit |
| 5. Research Design and Methods | Item 5 consists of the following 4 elements and is limited to 12 pages: A statement of the Challenge Area and specific Challenge Topic; The Challenge and Potential Impact; The Approach; and Timeline and Milestones. Attach the 12- page Research Plan encompassing all of these elements as a single PDF document. Figures and illustrations may be included but must fit within the 12-page limit. Do not include links to Web sites for further information. Do not include animations. |

Excluded from the 12-page Research Plan limit are the following items:

* Specific Aims (1 page maximum)
* Inclusion Enrollment Report
* Protection of Human Subjects
* Inclusion of Women and Minorities
* Targeted/Planned Enrollment
* Inclusion of Children
* Vertebrate Animals
* Select Agent Research
* MPI Leadership Plan
* Consortium/ Contractual Arrangements
* Letters of Support
* Resource Sharing Plans

Note the 12-page limit also excludes the Project Summary/Abstract; Bibliography and Literature Cited; and Biographical Sketches (separate PDFs).

Organize the Research Plan in the specified order using the instructions provided below. Start each section with the appropriate section heading (i.e., Statement of the Challenge Area and the specific Challenge Topic, The Challenge and Potential Impact, The Approach, Timeline and Milestones.)

**Research Area:** State which broad Challenge Area (e.g., (01: Behavior, Behavioral Change, and Prevention) described within this FOA and specific Challenge Topic (e.g., *Mechanisms of Behavior Change Research*: *01-GM-104*) will be addressed. Also include the project title on the first page.

**The Challenge and Potential Impact:** What is the research opportunity, scientific knowledge gap or technology that will be addressed? How broad is the potential impact in science and/or health? Which community (ies) will be affected? What is (are) the size(s) of the community(ies)? Will the potential impact be major?

**The Approach:** How will you attempt to explore or solve the stated research problem? How will your rationale and/or approach overcome existing challenges or barriers in the field? If you propose to improve existing technologies or to develop new technologies, which needs are being addressed and what is unconventional and exceptionally innovative about your approach? Provide enough information for reviewers to determine what you are proposing to do, but do not include a detailed experimental plan.

**Timeline and Milestones:** Provide a timeline for the proposed research indicating points where intermediate objectives will be assessed and decisions will be made regarding the course and direction of the continuing research effort. Possible alternative paths that may be followed at critical junctures in the project plan should be described and indicated on the timeline.

Preliminary data are not required but may be included, if necessary to demonstrate the feasibility of the proposed studies. The presentation must be clear and particularly compelling. No detailed scientific plan should be provided, but timelines must be presented.

**Inclusion of Women, Minorities, and Children in Challenge Grant Studies**

For Challenge Grant applications that propose human subjects research, applicants are expected to set forth sex/gender-based hypotheses and plans for data analysis based on a consideration of the relevant literature if the proposed study has the potential for such consideration. The purpose of this approach is three-fold: to ensure compliance with the NIH Guidelines for Inclusion of Women and Minorities in Clinical Research; to capitalize on the growing body of research demonstrating sex/gender differences in all areas of NIH research from basic to clinical and translational; and to ensure that any sex/gender-specific solutions/answers to the stubborn questions are not overlooked, thus resulting in incorrect conclusions/generalizations with respect to men or women. If these sex/gender-based hypotheses are not relevant to the proposed research, applicants should provide scientific justification for why sex/gender analysis would not be relevant.

Applicants for Challenge Grants are expected to address the inclusion of members of minority groups and their subpopulations in developing a research design appropriate to the scientific objectives of the study and set forth racial/ethnic-based hypotheses and plans for data analyses based on a consideration of the relevant literature.

The purpose of this approach is to: 1) ensure compliance with the NIH Guidelines for Inclusion of Women and Minorities in Clinical Research; 2) address gaps in what is known about health disparities between racial/ethnic groups; and 3) ensure that any potential answers to stubborn questions are not overlooked, thus resulting in incorrect conclusions and/or generalizations. If the inclusion of members of minority groups and their subpopulations is not relevant to the proposed research, applicants should provide scientific justification for why racial/ethnic analyses would not be relevant.

Applicants for Challenge Grants that include children are expected, consistent with the "[NIH Policy and Guidelines on the Inclusion of Children as Participants in Research Involving Human Subjects](http://grants.nih.gov/grants/guide/notice-files/not98-024.html)," to set forth age-appropriate hypotheses and plans for data analyses based on a consideration of the relevant literature. This approach is designed: 1) to promote better compliance with the NIH Pediatric Inclusion policy; 2) to address wide gaps in what is known about clinically significant differences, between children and adults and among children of different ages and developmental stages, in the diagnosis and treatment of diseases and conditions; and 3) to ensure that any potential answers to stubborn questions in pediatrics, as well as in early origins of adult disease, are not overlooked. If age-appropriate hypotheses are not relevant to the proposed research, applicants should provide a specific, scientific justification for why age-appropriate analyses would not be relevant.

**PHS 398 Research Plan structure and format**

**(part of SF 424 (R&R) Application for Federal Assistance)**

*1. Introduction to Application* ***XXX NOT REQUIRED*** *(for RESUBMISSION or REVISION only)*

2. Specific Aims

3. Background and Significance

4. Preliminary Studies / Progress Report

5. Research Design and Methods

Notes on Required Format ( <http://grants.nih.gov/grants/funding/424/SF424_RR_Guide_General_Adobe_VerA.doc>)

Text attachments should be generated using word processing software and then converted to PDF using PDF generating software. Additional tips for creating PDF files can be found at <http://era.nih.gov/ElectronicReceipt/pdf_guidelines.htm>.

When attaching a PDF document to the actual forms, please note you are attaching an actual document, not just pointing to the location of an externally stored document. Therefore, if you revise the document after it has been attached, you **must** delete the previous attachment and then reattach the revised document to the application form. Use the “**View Attachment**” button to determine if the correct version has been attached.

**Font:** Use an Arial, Helvetica, Palatino Linotype, or Georgia typeface

**Color:** Black

**Size:** 11 points or larger. (A Symbol font may be used to insert Greek letters or special characters; the font size requirement still applies.)

**Type density:** including characters and spaces, must be no more than 15 characters per inch.Type may be no more than six lines per inch.

**Page Margins:** Use standard paper size (8 ½" x 11). Use at least one-half inch margins (top, bottom, left, and right) for all pages. No information should appear in the margins, including the PI’s name and page numbers.

**Header/footer:** Do not include any information in a header or footer of the attachments. Page numbers for the footer will be system-generated in the complete application, with all pages sequentially numbered.

**Figures, Graphs, Diagrams, Charts, Tables, Figure Legends, and Footnotes**

You may use a smaller type size but it must be in a black font color, readily legible, and follow the font typeface requirement. Color can be used in figures; however, all text must be in a black font color, clear and legible.

**Acronyms/Abbreviations:** If terms are not universally known, spell out the term the first time it is used and note the appropriate abbreviation in parentheses. The abbreviation may be used thereafter.

**Separate Attachments**

Separate attachments have been designed for the Research Plan sections to maximize automatic validations conducted by the eRA system. When the application is received by the agency, all of the Research Plan sections will be concatenated in the appropriate order so that reviewers and agency staff will see a single cohesive Research Plan.

While each section of the Research Plan needs to eventually be uploaded separately, applicants are encouraged to construct the Research Plan as a single document, separating sections into distinct PDF attachments just before uploading the files. In this way the applicant can better monitor formatting requirements such as page limits. When validating for page limits, the eRA Commons will not count the white space created by breaking the text into separate files for uploading.

**Page Limits**

Although many of the sections of this application are separate text (PDF) attachments, page limitations referenced in these instructions and/or funding opportunity announcement must still be followed. Agency validations will include checks for page limits. Some accommodation will be made for sections that when combined must fit within a specified limitation. Note that while these computer validations will help minimize incomplete and/or non-compliant applications, they do not replace the validations conducted by NIH staff. Applications found not to comply with the requirements may lead to rejection of the application during agency validation or delay in the review process.

All applications and proposals for NIH and other PHS agency funding must be self-contained within specified page limitations. Unless otherwise specified in an NIH solicitation, Internet website addresses (URLs) may not be used to provide information necessary to the review because reviewers are under no obligation to view the Internet sites. Moreover, reviewers are cautioned that they should not directly access an Internet site as it could compromise their anonymity.

1. NHGRI. NHGRI Seeks DNA Sequencing Technologies Fit for Routine Laboratory and Medical Use. 2008 [updated 2008; cited]; Available from: <http://www.genome.gov/27527585>.

2. Mardis ER. Next-Generation DNA Sequencing Methods. Annual Review of Genomics and Human Genetics. 2008;9(1):387-402.

3. Schuster SC. Next-generation sequencing transforms today's biology. Nat Meth. 2008;5(1):16-8.

4. Bennett S. Solexa Ltd. Pharmacogenomics. 2004;5(4):433-8.

5. Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, Bemben LA, et al. Genome sequencing in microfabricated high-density picolitre reactors. Nature. 2005;437(7057):376-80.

6. Shendure J, Porreca GJ, Reppas NB, Lin X, McCutcheon JP, Rosenbaum AM, et al. Accurate Multiplex Polony Sequencing of an Evolved Bacterial Genome. Science. 2005;309(5741):1728-32.

7. Shendure J, Ji H. Next-generation DNA sequencing. Nat Biotech. 2008;26(10):1135-45.

8. Bashiardes S, Veile R, Helms C, Mardis ER, Bowcock AM, Lovett M. Direct genomic selection. Nat Methods. 2005;2(1):63-9.

9. Okou DT, Steinberg KM, Middle C, Cutler DJ, Albert TJ, Zwick ME. Microarray-based genomic selection for high-throughput resequencing. Nat Methods. 2007;4(11):907-9.

10. Fredriksson S, Baner J, Dahl F, Chu A, Ji H, Welch K, et al. Multiplex amplification of all coding sequences within 10 cancer genes by Gene-Collector. Nucleic Acids Res. 2007;35(7):e47. PMCID: 1874629.

11. 't Hoen PAC, Ariyurek Y, Thygesen HH, Vreugdenhil E, Vossen RHAM, de Menezes RX, et al. Deep sequencing-based expression analysis shows major advances in robustness, resolution and inter-lab portability over five microarray platforms. Nucl Acids Res. 2008;36(21):e141-.

12. Mortazavi A, Williams BA, McCue K, Schaeffer L, Wold B. Mapping and quantifying mammalian transcriptomes by RNA-Seq. Nat Methods. 2008;5(7):621-8.

13. Nagalakshmi U, Wang Z, Waern K, Shou C, Raha D, Gerstein M, et al. The Transcriptional Landscape of the Yeast Genome Defined by RNA Sequencing. Science. 2008;320(5881):1344-9.

14. Kidd JM, Cooper GM, Donahue WF, Hayden HS, Sampas N, Graves T, et al. Mapping and sequencing of structural variation from eight human genomes. Nature. 2008;453(7191):56-64. PMCID: 2424287.

15. Campbell PJ, Stephens PJ, Pleasance ED, O'Meara S, Li H, Santarius T, et al. Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing. Nat Genet. 2008;40(6):722-9.

16. Sebat J, Lakshmi B, Troge J, Alexander J, Young J, Lundin P, et al. Large-scale copy number polymorphism in the human genome. Science. 2004;305(5683):525-8.

17. Iafrate AJ, Feuk L, Rivera MN, Listewnik ML, Donahoe PK, Qi Y, et al. Detection of large-scale variation in the human genome. Nat Genet. 2004;36(9):949-51.

18. Cooper GM, Nickerson DA, Eichler EE. Mutational and selective effects on copy-number variants in the human genome. Nat Genet. 2007;39(7 Suppl):S22-9.

19. Chiang DY, Getz G, Jaffe DB, O'Kelly MJ, Zhao X, Carter SL, et al. High-resolution mapping of copy-number alterations with massively parallel sequencing. Nat Methods. 2009;6(1):99-103. PMCID: 2630795.

20. Wang C, Mitsuya Y, Gharizadeh B, Ronaghi M, Shafer RW. Characterization of mutation spectra with ultra-deep pyrosequencing: application to HIV-1 drug resistance. Genome Res. 2007;17(8):1195-201. PMCID: 1933516.

21. Chevreux B, Pfisterer T, Drescher B, Driesel AJ, Müller WEG, Wetter T, et al. Using the miraEST Assembler for Reliable and Automated mRNA Transcript Assembly and SNP Detection in Sequenced ESTs. Genome Res. 2004;14:1147-59.

22. Li H, Durbin R. MAQ: Mapping and Assembly with Qualities. 2007 [updated 2007; cited 2008 November 11th]; Available from: <http://maq.sourceforge.net/>.

23. Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome Res. 2008;18(11):1851-8. PMCID: 2577856.

24. Hernandez D, Francois P, Farinelli L, Osteras M, Schrenzel J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res. 2008;18(5):802-9. PMCID: 2336802.

25. Chaisson MJ, Pevzner PA. Short read fragment assembly of bacterial genomes. Genome Res. 2008;18(2):324-30. PMCID: 2203630.

26. Dohm JC, Lottaz C, Borodina T, Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res. 2007;17(11):1697-706. PMCID: 2045152.

27. Warren RL, Sutton GG, Jones SJ, Holt RA. Assembling millions of short DNA sequences using SSAKE. Bioinformatics. 2007;23(4):500-1.

28. Zerbino DR, Birney E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 2008;18:821-9.

29. Huang da W, Sherman BT, Stephens R, Baseler MW, Lane HC, Lempicki RA. DAVID gene ID conversion tool. Bioinformation. 2008;2(10):428-30. PMCID: 2561161.

30. Bolchini D, Finkelstein A, Perrone V, Nagl S. Better bioinformatics through usability analysis. Bioinformatics. 2009;25(3):406-12.