WASP DEVELOPMENT STRATEGY

a. Significance

The power of massively-parallel sequencing (MPS) and other genome-wide technologies to achieve major new insights is constrained by our limitations in managing and analyzing the extraordinarily large datasets generated. While this was a constraint experienced to some degree during the microarray era, the challenges of MPS are much more substantial, not only because the sizes of the datasets generated dwarf those of microarrays, but also because of the broader range of systems generating data and diversity of assays using these systems. MPS is not only being used to resequence genomes to look for variants and mutations and to sequence new organisms *de novo*, it is also used to read out the results of assays testing how the genome is organized or functioning, for example localizing chromatin components (ChIP-seq), measuring cytosine methylation (MethylC-seq, HELP-tagging), identifying DNase hypersensitive sites (DNase-seq), and many other applications. Each of these applications can be associated with different types of molecular assays (exemplified by the many ways of testing cytosine methylation), and each assay can have competing analytical algorithms (for example, peak calling approaches in ChIP-seq).

The upshot of this diversity is a multitude of local solutions to address the challenges of MPS, each understandably focused on local needs, which serves its purpose in the immediate term but does not lend itself to collaborative projects. This is a problem, as collaboration is increasingly necessary to allow large cohorts to be assembled for studies of human disease, or to allow a sufficient breadth of experience to be combined to tackle the challenges of a multifaceted study.

Like other institutions, ours approached the MPS data problem with the usual goal of serving local needs. We made the strategic decision to invest heavily in a system that integrated a laboratory information management system (LIMS) with administrative and other functions, but departed from the norm by including automation of MPS data analysis, largely from our Illumina sequencers. The concept was straightforward – if Illumina data files were generated in a standard manner, and we knew for each lane what the assay was intended to test (*e.g.* RNA-seq, ChIP-seq), we could initiate an analytical pipeline automatically for every sample and address at least what we called the primary analysis step – turning raw MPS data into biological information. We were able to automate peak calling for ChIP-seq, quantify expression levels for RNA-seq, cytosine methylation levels for HELP-tagging and so on. As we adopted the open-source MediaWiki software for our user interface, we referred to the system as the Wiki-based Automated Sequence Processor, or WASP.

What we came to realize was that the WASP system was of great interest to colleagues in other institutions, who began to ask at an early stage whether we could implement it outside Einstein, including in locations as far afield as Tokyo, Japan and Melbourne, Australia. As some of these enquiries came from external collaborators, for whom remote implementation would be of significant value in terms of enhancing collaborative projects, we began to explore the feasibility of making WASP a distributed rather than an institution-specific system.

This caused us to question the fundamental way in which we as a community of researchers are addressing the MPS data problem. We realized that there was an exceptional initiative attempting to create a cyberinfrastructure for plant biology, the iPlant Collaborative (http://iplantcollaborative.org/) that serves as a model for what we should be attempting in other organisms. This prompted the further realization that it is an unrealistic goal for one group or institution to attempt to develop analytical or cyberinfrastructure tools that will work for every institution, and that the distributed development model of Rocks Clusters (http://www.rocksclusters.org/) was worth attempting to emulate. It goes without saying that no comparable initiative has been undertaken for MPS data analysis, although individual components like SeqWare (http://sourceforge.net/projects/seqware/) and Galaxy [Giardine, 2005 #2186] represent excellent resources towards that goal. Our belief is that there exists a major need for distributed development of tools to manage and analyse MSP data. We therefore changed our strategy of WASP distribution to accommodate distributed development of the WASP system itself and of plug-ins that would be tested and integrated in updates of the WASP system.

We also appreciate another emerging trend that is going to create further challenges for data management and analysis. The companies producing the MPS equipment have started to sell smaller units designed to be used by individual investigator labs, rather than centers or core facilities. For example, the Roche GS Junior,

the Life Technologies Ion Torrent and the Illumina MiSeq systems are all priced at a fraction of higher-throughput units and are designed to address more targeted MPS experiments. While their throughput is less than antecedent platforms, they still have significant throughput, the MiSeq for example expected to generate >3.4 million clusters with up to 150 bp paired-end sequences per run, or >1 Gb of sequence. The companies are making efforts to integrate primary data analysis into these systems, but it almost goes without saying that they will not be able to anticipate every investigator need, as the types of assays and analytical approaches will be as diverse for these instruments as for their higher-capacity counterparts. It appears that MPS technology itself is distributing, so that even if centers and core facilities have their data management and analytical needs met, there will be an increasing demand arising from this distribution of sequencing that will also need to be addressed. The more users and platforms, the greater the number of assays and approaches that will need to be developed, a scaling of the burden that will be difficult to address solely by means of using resources developed by an individual group.

If we can achieve the goals of this project, what we will have accomplished will be the harnessing of talent throughout the USA, as well as internationally, to allow a co-ordinated and collective response to the challenges of MPS data management and analysis. It is certainly an ambitious goal, and we do not expect to achieve the universal adoption of the WASP system, but we are confident that we will be able to gather a critical mass of talent that will be able to work under the umbrella of the WASP system to allow a much more efficient and creative approach to MPS data analysis than offered by any prior initiative. We do not believe that the problems of MPS data can or should be solved by a single group, no matter how talented or resourceful, the challenges are too great and diverse, and a collective, distributed response has a much greater likelihood of success. Our goal is to create an environment in which such a collective response can be successful.

b. Innovation

Our proposal represents a reasonably radical departure from the normal approach to software resource development, which usually seeks as its deliverable the provision of the final product to an institution or to a resource for open-source distribution, such as Sourceforge or Bioconductor. While we will also make our software open-source, the major goal of this proposal is to create a system that can be readily adapted to serve individual institutional needs, with plug-ins that are shared by the community of WASP developers to allow a broad range of data analytical systems to be implemented. Individual components of this broader concept currently exist, as mentioned earlier, like Galaxy as a host for analytical tools, and SeqWare which integrates a LIMS with analytical tools. What WASP will do is novel in that it operates at a level even broader than these two examples, acting as a host for all of the components of a powerful MPS data management and analysis system. WASP therefore does not need to be considered to be distinct from other resources, as, for example, we can integrate Galaxy into WASP (as we have already done in our institution). The novelty of WASP is that it offers something more broadly useful than any existing resource, and it will allow a distributed development system to integrate the development efforts of multiple groups.

Another powerful component of our approach is also quite innovative. It is obviously very difficult to create a software suite that can be implemented universally when the hardware configurations will be as numerous as the institutions in which they reside. To overcome this problem, we describe two approaches, one that recognizes that many institutional clusters have the Rocks Cluster Distribution installed, allowing us to develop WASP as a Rocks “Roll” (plug-in), ensuring its compatibility with the existing system. The second approach recognizes that there are institutions that even lack adequate hardware resources, for whom we propose to generate a cloud implementation of WASP, allowing them to use external computational resources on an as-needed basis. The cloud approach can also be adopted by the institutions with significant hardware resources, using the Eucalyptus program (http://open.eucalyptus.com/) to create a ‘local’ cloud, to which external cloud resources can be added when extra capacity is needed. We anticipate that this will be the more common implementation of WASP as it ultimately offers greater flexibility with little downside. These approaches to distribute the distribution of WASP to a very diverse range of hardware configurations is necessarily innovative.

c. Approach

Preliminary Studies for New Applications: the current WASP system.

Our current WASP system is serving the needs of the Einstein community and is Illumina-centric, but illustrates how even the current system is providing an extremely valuable resource to a large number of researchers. A rationale for this funding application is that we will be facilitating the work of a very large number of NIH-funded investigators. While this number will scale with the distribution to other institutions, it should be recognized that even as a resource for our own institution it is supporting many NIH-funded projects.

A description of the WASP system is currently submitted for publication at Genome Research where it is in review. We have created the ability to browse the resource at http://wasp.einstein.yu.edu/ with the Username **Guest** and Password **User**, although we recognize that reviewers are under no obligation to use applicant-supplied URLs when evaluating a funding application. We have therefore decided to provide substantial detail here about the current system in order to be transparent about our approach, allowing the reviewers insights into how we tackled this challenge, and providing a reference for how we intend to modify the system for distributed development.

The technical specifications can be summarized as follows: the WASP system is designed with a 3-tier architecture utilizing MediaWiki (v1.16.1) as the presentation layer, running on Apache 2 (v.2.2.9); a MySQL (distribution 5.0.51a) database, for both MediaWiki and the LIMS, and an NFS disk array as the information (data) layer; and a logical layer implemented on a Debian GNU/Linux 5.0 server. The Linux server also functions as a submit node to a local 95 node (1,360 core) Rocks 5.1 Sun Grid Engine (SGE) 6.1u5 cluster for parallelizing data processing and analytical tasks. The WASP software framework is primarily written in Perl. In addition, Python, AWK, R, and C++ are also used for scripting and processing tasks. The intelligent dynamic HTML forms for sample submission are implemented in PHP with Ajax and the forms for the LIMS are implemented in PHP.

Raw data from sequencing instruments are written to an SMB network share comprised of a write-optimized, 72 TB RAID-Z disk array, where the data are compressed on disk. Downstream operations by WASP pipelines are written to a read-optimized, 42 TB RAID-Z disk array, through a network file system (NFS). All hosts are backed up by a Legato/EMC Networker and completed runs are permanently archived to tape after 3 months.

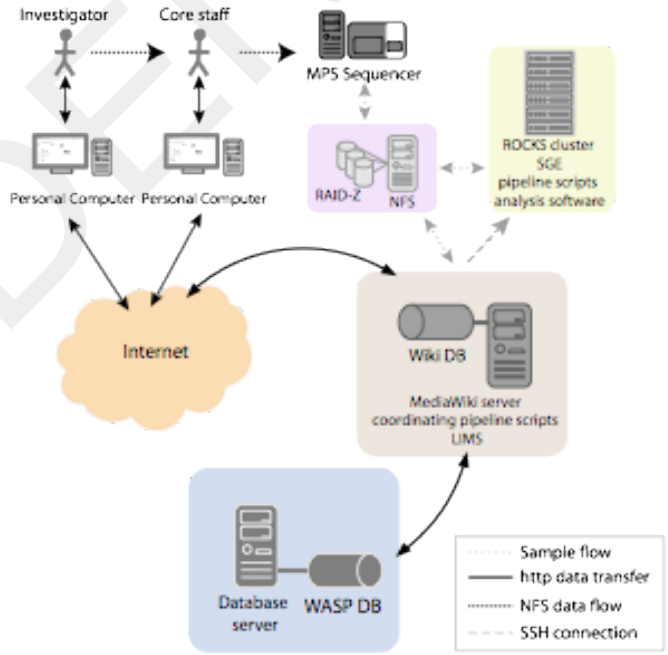

FIGURE XXX1: The basic hardware architecture of the current WASP system comprises a simple virtual Linux server running Apache to host both our MediaWiki instance and core WASP software components. Data generated by the sequencing equipment are deposited to a local network file system (NFS) which is also accessible from the host server and cluster to facilitate processing and analysis. The WASP metadata database is hosted by a dedicated virtual Linux server.

The basic architecture of the WASP system in its current manifestation is shown in **Figure XXX1**. Investigators interact with WASP primarily through an instance of MediaWiki (www.mediawiki.org), an open-source software component with a familiar front end and an advanced content management system that includes versioning and data integration capabilities. This medium allows users to create, edit and share information in an intuitive way. Investigators use the wiki to register with the system, submit job requests to the sequencing facility, track the progress of each submission, retrieve their raw, aligned, and analyzed sequencing files, and display analyzed results within a Genome Browser. Users are grouped into laboratories headed by a Principal Investigator, whose lab members are able to view, download, and share one another's data. To address data privacy, we have added an access control

extension in order to restrict viewing and editing of pages to an individual investigator or lab, with the additional flexibility necessary for setting up a secure collaboration environment. Moreover, as standards for depositing MPS data to public repositories are defined, it will be possible to adapt the pipeline to ensure all required metadata is captured to the system from the investigator and core facility staff and stored indefinitely alongside the raw sequence data and primary analysis results. The use of a wiki as a user interface enables us to provide investigators with a structured source of information about our MPS services and also enables users to provide feedback to other WASP members, for example by using the *Discussion* tab associated with a protocol page to make comments about their experiences with the protocol.

To simplify the submission of requests to the core sequencing facility, we developed Ajax-based dynamic forms that build themselves in response to user selections. Following the submission of a job request, details collected from the forms are parsed and stored within a local relational database. Wherever possible, form elements are pre-populated with data retrieved from the database based on previous submissions by an investigator, thus reducing the need for duplicate entries. Data collected include the identity of the submitter, laboratory affiliation, billing details, the assay and type of sequencing reactions being requested, specific attributes about each sample, file uploads documenting sample details, the number of lanes required per sample, and, if required, instructions related to sample multiplexing. Following submission, a summary of each request and a computer-generated invoice are automatically transmitted by email to the submitter.

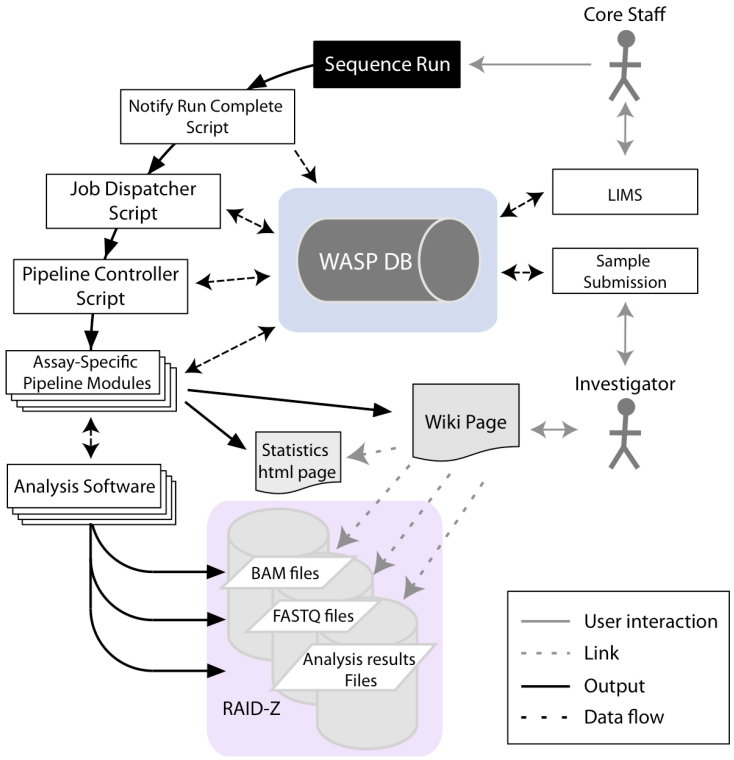

FIGURE XXX2: A schematic showing the basic use of the WASP system and data processing flow. The process starts with an investigator completing an online sample submission form on the wiki, during which metadata is captured to the database. Core facility personnel process samples in the laboratory and enter further metadata into the database. When sample libraries are ready, a sequence run is initiated and the WASP system is primed to expect sequence data to process. Data processing begins as soon as the sequence run control script calls the WASP notifyRunComplete script after completion of sequencing. The WASP system then proceeds to analyze the data, based on metadata from the database that describes the samples and assay type. Finally, output files are deposited on the NFS server, a wikipage is written containing links to access the data, and the investigator notified.

Secure, controlled access to the WASP database is available to core facility personnel through a web-based PHP application that enables viewing details for each job request including processing status pre-sequencing, or analysis status post-sequencing. The LIMS provides forms to capture downstream laboratory workflows that include library creation, cluster generation and sequence runs. These details are required to link sequencing files generated by commercial sequencers with each sample and to select the appropriate analytical pipeline for execution. The LIMS component and experimental metadata are stored in a custom MySQL database that is utilized by the WASP scheduler/dispatcher process for submission of jobs to the Sun Grid Engine (SGE) and tailoring of automated, custom analyses.

WASP incorporates modular, analytical pipelines that process and analyze raw sequencer data. **Figure XXX2** provides a basic overview of WASP software architecture. On completion of a sequence run, the relevant information, including the directory path to the sequencer output, is entered into the WASP database by execution of a custom shell script on the sequencer control computer. The raw data are transferred to NFS storage that is also accessible by the WASP server and the ROCKS-SGE cluster (**Figure XXX1**). A daemonized script on the WASP main server constantly inspects the WASP database for the presence of completed sequencing run information for jobs

marked as 'pending analysis'. When these data become available, the script executes an instance of the WASP analysis pipeline for each job for which there is at least one sample on the sequenced flow cell. If the job does not have an associated analysis pipeline and no genome was specified by the submitter, the simplest form of data processing is performed: the generation of FASTQ files. Alternatively, if a supported genome is provided, an alignment to the specified reference genome is also completed and provided as a BAM file. If the assay is supported by an analysis pipeline, the data are fully analyzed. For increased compatibility with downstream software and greater flexibility in the future, all sequence files that are made available to investigators are in community-accepted, standard formats, *e.g.* BAM files [Li, 2009 #2188] for alignments. When necessary, non-standard formatted files (generated by commercial sequencing products) are converted to a standard format. WASP uses a file naming convention that is easily parsed by pipeline scripts and is also human readable.

All pipeline output is stored in a separate, read-optimized NFS location, and includes logs of standard output and error streams from pipelined software components. The users' data are placed within a standardized folder tree organized by PI WASP ID, submitter WASP ID, Project ID and finally Job ID. Each job folder contains 3 subfolders, 'raw', 'processed' and 'analyzed,' which are used by the WASP pipeline to organize files during processing and analysis. The project folder also contains a copy of the current version of an XML file that stores both information about the software packages used for analysis and customisable parameters. Using this XML file, it is easy to update a program when a new version is released, or to replace a software component entirely, *e.g.* substituting MACS [Zhang, 2008 #1937] with PeakSeq [Rozowsky, 2009 #1964] for ChIP-seq analysis. By archiving a copy of the current revision of this file with the creation of a new project, we ensure consistency of analysis for jobs organized by the submitter into a common project.

With regard to Illumina sequencing runs, the proprietary raw sequence (QSEQ format) files are converted to standard FASTQ files with Sanger encoded quality scores. We focus here on Illumina technology here because WASP was primarily developed to support processing and analysis of data obtained from Illumina GAIIx and HiSeq 2000 sequencers installed at the Albert Einstein College of Medicine Epigenomics and Genomics Core Facilities. For multiplexing, we use custom 7 bp indices present in the adapters that are identified and trimmed from the 5' ends of sequence tags. If a lane contains multiplexed samples, they are demultiplexed during the generation of FASTQ files. The FASTQ files are then used to generate alignments to reference genomes using ELAND or Bowtie [Langmead, 2009 #1963]. The analysis pipelines implement best practises in choosing their modular components, leveraging popular, open-source tools and resources to perform assay-specific analyses, *e.g.* MACS for ChIP-seq [Zhang, 2008 #1937], TopHat [Trapnell, 2009 #2187] and Cufflinks [Trapnell, 2009 #2187] for mRNA analysis, miRBase for miRNA analysis as well as SAMtools [Li, 2009 #2188] and Picard (http://picard.sourceforge.net/) for SAM/BAM file manipulations. Additional custom scripts provide support functions such as data transformation, collation, filtration and preparation for visualization. Each pipeline is implemented as a module, which need only implement its own functionality. Common functionality is implemented in the base classes inherited by the assay-specific modules. This includes extracting commonly required metadata from the WASP database, information from sequencer configuration files, and software version information. The base classes also handle copying of sequence run quality data.

Upon completion of WASP analytical pipelines, the investigator is notified automatically by email and access to the results is provided through an email-embedded hypertext link or by navigating on the web from the investigator's wiki home page or lab page. If sequencing of samples for a particular job is incomplete, a partial analysis may, depending on the assay, be performed and data returned to the user.

Analyzed data, as well as statistics summarizing selected attributes of each sequence run, are provided to the investigator in easily interpretable, tabular and graphical formats. Applicable results can also be viewed on a local mirror of the UCSC Genome Browser (http://genome.einstein.yu.edu/), thus immediately providing the biological context of the data. For example, the major output of the ChIP-seq pipeline is a list of peaks (already subtracted from background noise) which can be visually compared alongside other publicly-available or user-supplied genomic annotations, or correlated with such data.

A standard results page is divided into four major sections: Job Description, Sequencing Quality Metrics, Sequencing and Alignment Results, and Assay-Specific Analytical Results. The Job Description section contains metadata related to the specific job and includes the assay type, the versions of all software used

during the automated analysis, and a status field that shows the current status of the job (*i.e.* incomplete or complete).

Through an intuitive graphical view generated using Google Chart Tools (http://code.google.com/apis/chart/), the Sequencing Quality Metrics section enables an investigator to assess quickly and easily the critical parameters of each sequence run, lane-by-lane. These statistics include the number and position of undetermined ('N') bases within a lane, mean quality (Phred) scores, as well as alignment and run quality scores. For short reads that require downstream adaptor trimming prior to alignment, the fraction of sequences containing identifiable adapters and the adaptor positions are given. Finally, Intensity Versus Cycle (IVC) plots are made available to depict each nucleotide's signal strength per addition cycle.

The Sequencing and Alignment Results section provides links to raw (FASTQ) and aligned (BAM/SAM) sequence files that are ordered hierarchically by flow cell ID, sample name, lane, and multiplex index. In addition, links are created to display aligned reads on the local mirror of the UCSC genome browser. MD5 checksums for all the downloadable data files are routinely generated to facilitate future submission to public repositories. Finally, this section offers links to view or download a log file recording timestamps and all software commands executed by a pipeline run, accompanied by any software output text.

The Assay-specific Analysis Results section presents links to analysis files generated by an assay-specific analytical pipeline. These files can be viewed both in tabular and graphical formats. Thus far we have implemented analytical pipelines for four assays: ChIP-seq and HELP-tag define peaks from chromatin immunoprecipitation and cytosine methylation values at HpaII sites, respectively; while microRNA-seq and RNA-seq quantify transcription of different types of RNAs. RNA-seq also can reveal qualitative information about novel splicing patterns. Tabular results are returned in tab-delimited format for simple import into spreadsheets that facilitate data exploration and manipulation. By contrast, analytical results in BED files and similar formats in can be visualized graphically by importing into a genome browser.

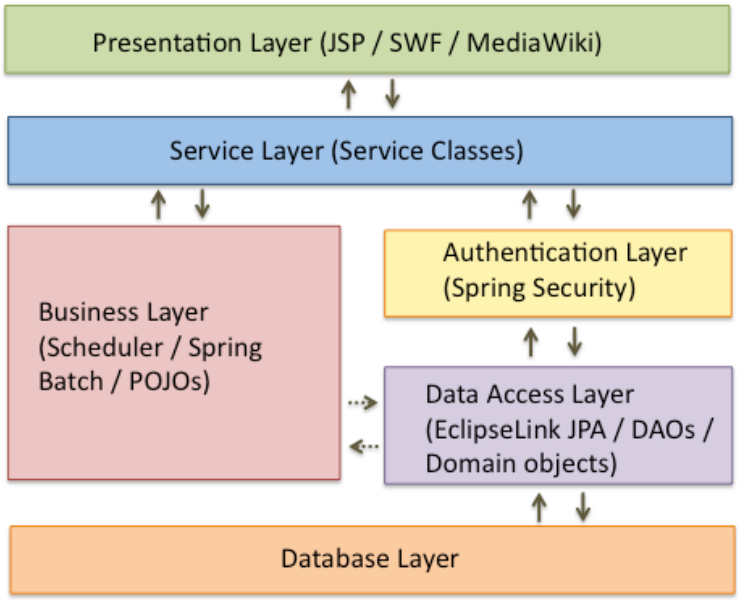

FIGURE XXX3: A diagram showing the software layers of the proposed Java/Spring-based WASP system. The user interface remains a MediaWiki instance but now contains embedded JSP pages for more dynamic interaction. Spring Web Flow (SWF) is used to manage dynamic form building. The service layer handles customized retrieval of data as required by the presentation and business layers. Spring security is used to ensure data access is authorized based on user role. The data access layer represents an API for generic interaction with the WASP database and includes domain objects which feature object relational mapping (ORM) annotations and data access objects (DAOs). The Business layer contains core plain old Java objects (POJOs) and implements the Spring Scheduler and Spring Batch framework to manage automated job pipelining.

Extensible framework development

While the above represents a working system that could be of major value to other institutions in its current form, we recognized a number of deficiencies that preclude its general utility. It is primarily Illumina-centric, is very focused on Einstein core facility needs, and relies to some extent on commercial software (from Illumina). It also required a more robust design to equip the WASP software with the flexibility, maintainability, extensibility and scalability required to permit and encourage third party institutions to develop plug-in functionality. Because of these issues, we have taken the decision to completely refactor the software (**Figure XXX3**). The refactored WASP has been designed to accommodate multiple sequencing platforms, multiple assays and even multiple languages, given the interest in the system from colleagues in Japan (see attached letter from Dr. Kunio Shiota, University of Tokyo). We also wanted to make the capture of metadata in a comprehensive and standardized way to be a part of the refactored WASP system, as there will be a

pressing need to make the WASP database interoperable with those containing information from medical records, as clinical applications represent one of the most important and challenging applications of MPS.

The WASP system is to be re-designed using the Spring Java/J2EE application framework to develop discrete software components which are then deployed as services on the Eclipse Virgo OSGi application server platform. This enforces separation of concerns, with the web model-view-controller (MVC), database, domain, security, scheduler and data analysis pipeline components developed as individual Spring application bundles within Virgo. This means it is possible to update / restart a software component without interfering with independently running components. The sample submission forms will be developed in JSP using Spring Web Flow (SWF) to direct page display based on easy configuration using XML flow-definition files. Using SWF will make it easier to dynamically build a form based on user interaction and can be readily adopted by third party institutions during creation of their own forms. Spring Batch provides a job execution environment with lifecycle management which handles job launch, scheduling and restart of jobs. This, combined with integrated web-based job tracking and management tools, makes Spring Batch an ideal framework for developing analysis pipelines.

After reimplementation, the WASP software will comprise core pipeline components including some plug-in component bundles already included for immediate use out of the box, with provision for rapid custom plug-in development within third party institutions. Third party plug-in development will require creation of bespoke OSGi bundles and where necessary, accompanied by domain-specific database /domain architecture and data access objects (DAOs). Third party assay plug-ins will also include SWF definitions, JSP forms and spring Batch context definition xml files. As these are widely used technologies, third party developers will easily be able to learn how to develop plug-ins and the framework we develop will handle most of the complexity, thus, expediting plug-in development. Application development with Spring encourages programming to interfaces, so plug-in developers will be able to derive their tools by existing functionality. For example, an institution may have developed a novel stand-alone tool for peak finding which they wish to apply to ChIP-seq analysis. They will be able to extend a Spring Batch tasklet base-class bean we provide and develop a wrapper around the tool for incorporation into their instance of WASP. By simply altering a properties file, they can then call this tasklet in their ChiP-seq analysis pipeline instead of the default peak-finder.

Spring natively supports internationalization and localization and WASP will take full advantage of these features simply by providing alternative language versions of the messages resource files that we provide in US English.

Software packaging

As mentioned earlier, we propose to package WASP for distribution in two additional ways. First, by creating a roll (plug-in) for the Rocks Cluster Environment, and second, by creating a series of Amazon machine images (AMI) that can be mounted as cloud computing resources within a private or a public cloud environment. Traditional means of distribution via tar/gzip packages will be supported indefinitely for institutions and departments desiring highly customizable manual installations.

The Rocks environment (Rocks Core Development is sponsored by NSF award #OCI-0721623) has over 1700 registered cluster installations and an active user community. Our proposal to make WASP available via a Rocks Roll is twofold. First, it will ensure the largest possible dissemination of our pipeline to already existing cluster users. Second, it will tie sequencing to existing computational resources without investigators having to build or design their own clusters.

The cloud implementation of WASP will allow private investigators to work within emerging cloud-based infrastructures. Growing genomics data sets distributed over public clouds are changing access and utilization patterns. Our pipeline incorporates this change on two levels - data transport and data analysis. WASP addresses data access by offering flexible file system and database level access for its analytical processing using Teragrid gateway transport components and, where applicable, Aspera acceleration. Processing or computational resources are provided by WASP's cloud provisioning protocol using Chef Server processes. These processes tap into Einstein's cloud data depository, on-request filesystems, and pre-configured

compute nodes. They use a self-provisioned SGE scheduling system to allow for batch processing of data. For Mapreduce processing, WASP uses Amazon's Hadoop API to tap into Amazons EC environment.

Hardware requirements:
Despite the multiple components included in the WASP pipeline, the associated hardware requirements of a basic installation are minimal. Extended installations that aim to use additional open source analytical packages and particularly large indices will require a more extensive infrastructure investment. These storage and cluster components should scale to the needs of the individual institution. While WASP does support these infrastructures, it does not in itself provide them. For a fully functioning environment the WASP pipeline assumes the existence and production readiness of storage and cluster components for data storage and sequence analysis. Requirements for various application packaging types for basic installations are as follows.

Manual Installation:
Single server or VM (virtual machine) with 16GB of memory, 72GB of disk space

Cloud Implementation:
The equivalent of High Memory EC cloud instance or Xen or Virtual Box-type instance

Rocks Roll Implementation:
A Rocks "appliance" or single server (node), or VM (virtual machine) with 16GB of memory, 72GB disk space

Operating System requirements:
Linux flavors of Debian, Centos and Redhat with kernel support 2.6.18 and higher.

Software requirements:
The distribution package contains all essential software components to successfully start a demo or production instance of the WASP pipeline. Investigators, preferring their own database services, need to use MySql 5.5 and higher.

WASP distribution

We propose to release the refactored WASP system in two phases. In the first, we will create an alpha version of WASP and release it to only a small number of collaborating institutions (see attached letters of collaboration). In this phase we will be testing how well we can implement the distribution (Rocks Roll or Cloud approach, or both) as the first step. The second goal is to get feedback from these institutions that identifies weaknesses in the initial implementation, such as non-intuitive elements that require better documentation, software bugs, unanticipated inflexibility in addressing local needs, and so on. We will include a foreign language-speaking collaborating institution in this first phase to allow testing of the WASP translation function.

Once the first phase is accomplished, the second phase will entail making WASP available to the general public. Our anticipated approach to distribution is described in the **Policies** section below. While we anticipate that the availability of WASP will become reasonably well known through word of mouth, we will also publicize it through our ongoing multi-institutional collaborative projects such as the NIH Roadmap in Epigenomics, ENCODE and other initiatives, allowing us to reach a large number of institutions efficiently. We will also prepare a manuscript describing the refactored WASP system for publication to gain attention through the literature. It is anticipated that we will continue to develop vendor relationships over time that may result in their recommending the implementation of WASP, a very effective means of reaching potential users. We would like to be able to support at least 50 new user groups per year, recognizing that some of these will be centers and core facilities but that others will be individual investigators using personalized sequencers.

The software distribution will have two components, the core WASP system and optional plug-ins that perform specialized analytical tasks. As we anticipate that an individual user group will not need the entire range of plug-ins that have been developed by the WASP developer community, we will make these plug-ins optional for installation.

Once implemented at an external site, we need to be able to provide support for the user group especially at the early stage of deployment. In keeping with the philosophy of WASP supporting and being developed by a community, we will create a number of means for communication for WASP users, including an email listserv, and a bulletin board (possibly hosted through or in conjunction with a resource like SeqAnswers (http://seqanswers.com/). Additionally, access to the WASP bug-reporting system (see: Creation of a

distributed software development environment) will be made available to quickly address software issues. Einstein developers will take the lead in moderating these resources, allowing implementation issues to be resolved efficiently.

We are embedding the web conferencing tool Yugma (https://www.yugma.com/) within WASP to allow a periodic (*e.g.* bimonthly) user convocation at which the Einstein and other developers make themselves available to answer questions, describe updates in functionalities, and describe planned future development. This may be a very effective means of encouraging communication and of resolving issues quickly, so we plan to trial a Yugma approach to communication during the early stages of distribution of the system, maintaining it more formally if it appears to be a successful approach.

We will describe below our proposed approach for further, distributed development of the WASP system, but it is worth mentioning here that once WASP is implemented in an external institution, we need to have the ability to provide updates to the software, not only entire new versions of the software with added functionalities, but also patches that fix bugs identified between version releases.

Maintenance and upgrades will be handled using two strategies: minor maintenance releases of pipeline components and installation of additional plug-in modules will routinely be managed by utilizing the OSGi lifecycle to uninstall and install updates. Since the design of components of the WASP platform is highly modularized and robust to changes in availability in external resources, most updates should be done in place, with negligible degradation of service. Major updates will require that the system be halted for reinstallation of a new Rocks roll or for transition to a new instance of a cloud-based deployment. Users will be notified through a developer's network of updates to the WASP system and given instructions and support when updating their systems.

WASP security

The primary goal of the WASP pipeline is to deliver results to private investigators at collaborating institutions by way of a friendly but secure environment. The WASP infrastructure provides a layered computational infrastructure, each layer with its own mode of security. The security philosophy is closely modeled after the developers' document *Best Practices for Planning and Design,* published by Teragrid Science Gateways (https://www.teragrid.org/web/science-gateways/security**).** The WASP architecture addresses security issues on the following levels: user access, data security, database security and access management for computational resources.

Access to the WASP pipeline is based on credentials. Following organizational policy requirements, credentials are created centrally and propagated through the WASP pipeline. Proposed credentials are checked for complexity and availability and subsequently managed by WASP's lifecycle and identity management protocol. After re-development of the wasp system using Java Spring, the Spring Security framework will be employed to manage authentication, including role-dependent access to data and services. Optionally, science gateway or community accounts can be provisioned to allow full pass-through to institutional clusters or Teragrid resources.

Data security is enforced by credential-based access to result files. Using WASP community accounts investigators can access their data both via the WASP interface and from within their computational environment. Default permissions can be modified to match local policy requirements by collaborating institutions.

Database access is designed to be restricted. It is preferable to install the database backend on a highly secured server with only standard database ports open to the WASP interface. WASP uses a limited number of credentials, which are passed in a secure fashion from the gateway to the database. WASP offers field or table level encryption for users requesting Postgresql or Oracle. The choice of database is user-selectable at the time of installation.

Analytical work on the intersecting computational resource can be initiated by a common WASP system user or by the credentials of individual user. The computational framework is secured by WASP resource access policies, which can be further customized by local system administration.

Creation of a distributed software development environment

With the WASP system distributed and functional in multiple institutions, it appears inevitable that new user needs for data management and analysis will emerge. We therefore want to facilitate the development of the core WASP system or plug-ins to allow these diverse needs to be met and accommodated in an ever-increasing suite of tools in the WASP system.

The problem with the distributed development of components of the WASP system or its plug-ins is that it places the system at risk of incorporating incompatible components that will degrade overall system performance. For this reason we propose to create an **Integrated Team Software Development Environment (ITSDE)** hosted on servers at Einstein. This will involve access to our existing software revision control system (Subversion http://subversion.tigris.org/), codebase browsing (FishEye, Atlassian), continuous integration server (Bamboo, Atlassian) and integrated bug and issue tracker (Jira, Atlassian), all working in coordination with the Eclipse integrated development environment (http://www.eclipse.org/). The project management tools of the ITSDE will help with storyboarding ideas, planning releases, monitoring development tasks, managing builds and controlling integration testing. Using Jira it is possible track bugs / improvement requests (issues) from submission through resolution, integration testing and release. Issues may be identified by developers and submitted directly to Jira, or by users through a form provided in the WASP wiki, or even by the software installations themselves after detection of abhorrent behavior.

Core software components and plug-ins developed by the 'core' team will be maintained under the core WASP project and will not be directly modifiable from outside the core team to ensure overall system stability. Third party plug-in developers (TPPDs) will develop their software based on the Eclipse OSGi plug-in model, using Einstein developed software templates that conform to the WASP development architecture. They shall be encouraged to present their products for distribution with WASP either as integrated plug-ins or for optional installation. In situations where TTPDs have provided plug-ins which have been accepted for inclusion within WASP, they will be provided with resources within the ITSDE for further developing and testing their plug-in with pre-release versions of the WASP core. Before any new WASP release, a feature development freeze on the core will allow time for TPPDs to test compatibility of their plug-ins. A subsequent freeze on plug-in development will enable the core team to complete final integration building and testing with Bamboo prior to release. TPPDs will be responsible for handling any Jira issues relating to their contribution and will retain their copyright and intellectual property of their innovation and code.

Testing will involve the use of a test harness integrated within the Bamboo continuous integration and build server to run regular unit and integration tests whenever code is updated and for individual developer mediated testing of modified code. Unit tests exercise individual software modules (classes) and ensure correct behavior given both normal and abnormal use cases and corner cases. Integration tests exercise the system as a whole and simulate complete analysis of real sequencing experiments. The Bamboo server will provision the system into a functional environment to provide frequent snapshots of the working system as a whole for distribution. Finally, a manual testing step by the WASP core team will ensure proper deployment and upgrades of the system as well as completeness of code distribution and documentation (reference and API).

Leadership and oversight

There will be two types of oversight for the distributed WASP development system. The first recognizes the requirement for supervision of grant-funded projects to be supervised by the awardees, so there will be an internal Einstein advisory group that will supervise the ongoing progress of the project. This will consist of the significant personnel listed on this proposal, the PI (Greally) and co-investigators Zhang, McLellan, Calder and Hargitai.

The second requirement for oversight comes from the need to make choices about how the system should evolve over time. There are likely to be choices necessary during the development of the WASP system, as the Einstein developers will not be able to provide unlimited services, so it will be essential to gain perspective from a representation of the broader community to determine which directions are of greatest general use.

We propose that the five Einstein personnel listed above be part of this executive committee, but that it is structured in such a way that we represent a minority, so that if the total membership is 12 people the external members (XXX more specific language: External advisory board [members]?) will have the ultimate say, while a consensus from the Einstein group will still represent a powerful voice. This model is certainly open for modification and will undoubtedly change with time, but represents a useful starting position from which to work.

It makes sense that the external members of the committee be drawn from the institutions that have adopted WASP. We propose that each institution nominates a single representative to participate in this executive steering committee, and that there is a poll each year of all registered developers to vote in the committee members. In the first year we may not yet have a critical mass of people to allow this process to take place, in which case the Einstein group will invite a group of external users to serve in an interim capacity.

We anticipate a 6 monthly major release schedule for the WASP system. We propose that the executive steering committee should convene prior to these major updates to discuss any decisions regarding what should be included, and to define strategic priorities for the subsequent 6 months.

We believe that it is essential in a distributed development project to have oversight also distributed beyond the host institution's walls, and we will pay attention to maintaining the value of this oversight process, requesting from the executive steering committee their active input, being transparent to the development community how we make decisions and what those decisions are, and ensuring that the entire project is responsive to both the oversight group and to the developer community.

Policies

There are certain policies that will need to be in place to protect a number of interests. The first has to do with software licensing. As WASP will continue to host software that is protected by the Gnu Public License (GPL), which requires that WASP itself be made freely available, we need to have a mechanism of doing so. We will therefore make the source code and disk images for cloud implementation and the Rocks Roll freely and visibly available on an Einstein server. We also need to address the rights of developers of the plug-ins and other components of the WASP system. Our policy will be that developers retain copyright of any component they develop, and they will be prompted prior to inclusion of their software in a WASP release to complete a copyright claim that protects their intellectual property. Should they wish to assign the copyright to the Gnu Public License, we will also facilitate that. Full documentation of software contributed to the WASP system will be a prerequisite for its inclusion.

The WASP system itself is being prepared for copyright protection at present, to allow Einstein to control its dissemination. Our institutional policy will be to allow free use of the software by any not for profit academic or clinical enterprise, but to retain the right to withhold its use by commercial entities. Use by for-profit enterprises will require a licensing fee that will be used to support the further development of the WASP system.

The relationship with vendors who are developing MPS or related technologies is somewhat more complicated. We will seek to develop mutually-beneficial relationships with them to allow our developer community to gain insights into their ongoing software or LIMS systems, to learn in advance of planned file format changes and any other useful information that would impact WASP functionality. From the vendor point of view we are getting feedback that suggests they could reap unexpected benefits from implementation of the WASP system, for example gaining insight into performance of batches of reagents used in multiple institutions so that a batch failure issue can be distinguished from a local institution-specific issue. We will also consider on a case by case basis whether to allow the vendors to develop tools through our software development environment, as this would represent a valuable harnessing of resources of mutual benefit to companies and end users. The input of our executive steering committee will be sought when evaluating whether access should be granted to these vendors.

Potential future directions

Finally, it is worth putting this funding proposal into the context of our other goals for the WASP system. While we have focused here on the creation of a software development environment, we also intend to pursue separately the development of what we call Discovery and Collaboration environments. The **Discovery environment** addresses the data analytical challenge omitted by the current WASP system, which only brings data analysis to the stage of primary analysis, leaving unaddressed the challenge of integration of these results with other datasets, to allow their interpretation. For example, while automating the peak identification of ChIP-seq is very valuable, the interpretation of these results depends on knowing whether the pattern is concordant or discordant with those obtained in other experiments, whether the peaks reside in a specific genomic context, whether the nearby genes share a function or pathway, all requiring comparison of the ChIP-seq peak dataset with other datasets. We are developing tools such as Hadoop's Mapreduce and distributed filesystem (http://hadoop.apache.org/) to facilitate these kinds of enquiries. We also recognize that an automated pipeline will not always serve needs universally well, and that the modules we include may need to be substituted or modified. We are therefore exploring the use of Galaxy [Giardine, 2005 #2186] as a means of allowing the user to create customised pipelines, and the Taverna application [Hull, 2006 #2189] as an alternative.

The **Collaboration environment** concept arises when it is realised that the adoption of the WASP system in different institutions facilitates the creation of completely interoperable databases, if standards are imposed on the metadata entered describing the experiments and if the database structures are made compatible. There are standards developed by caBIG that have been implemented as part of The Cancer Genome Atlas (http://tcga.cancer.gov/) that could be implemented in WASP to allow this interoperability goal to be achieved. It is conceivable that in the future collaborative projects between institutions where WASP has been adopted could allow searching of datasets across institutions, a grid computing model that would substantially increase the efficiency of these projects.

While these separate initiatives are mostly at the aspirational stage, we mention them to illustrate how we see the value of the **Software development environment** of this proposal creating a foundation for even more powerful resources for the user community. Ultimately we hope to serve as the co-ordinating hosts for a self-sustaining enterprise creating software systems that overcome the challenges of MPS datasets, hastening insights and productivity by the community of researchers as a whole.