

RunJob Project Outline

--DRAFT II--

*Anzar Afaq, Greg Graham, Gerald Guglielmo, Eric Wicklund
Fermi National Accelerator Laboratory*

*Dave Evans
Imperial College*

*Peter Love
Lancaster Univeristy*

Introduction

MCRunjob was first employed during the 1999 Monte Carlo Challenges of the DZero experiment. It was designed to cope with the configuration of multiple cooperating applications generating and processing a flow of data, to wrap these applications in a form by which they could be submit as batch jobs, and to track or facilitate the tracking of these jobs through success or failure states. The requirements on MCRunjob were that it be able to interoperate with SAM by producing the metadata needed to import produced Monte Carlo files there, that it produce scripts to run the applications in a variety of execution environments, and that it be able to execute them in those environments: ranging from mainframes at Fermilab and NIKHEF to Linux farms at Fermilab and around the world. The solution was to adopt a modular architecture with customizable interfaces to allow for easy extension to new environments and applications. Each distinct application, database, or external service was modeled internally as a Configurator. The Configurator kept the relevant metadata describing the application input parameters, results of database queries, and service invocations; and were also responsible for generating jobs to achieve a specific data processing workflow pattern. The Configurators were maintained in a container called the Linker that coordinated the actions of the Configurators, facilitated communication among them, and collected application specific job scripts into a unified “linked” job including all applications. Another Configurator abstracting the local execution services (such as LSF, PBS, or LSF) could then submit the job. Control of MCRunjob is generally achieved using macro scripts. The user will write or will use a canned script containing macros that are interpreted by the Linker into calls on the Linker and Configurator APIs. In DZero, this technology has advanced to the point that a user may submit only a very generic script and most of the actual macro script is written on the fly.

Since that time, MCRunjob has been successfully employed by the CMS experiment to do worldwide Monte Carlo production using a production control database located at CERN and new execution environments including the Condor-G/Globus environment of the Virtual Data Toolkit (VDT), Chimera, and the first middleware package of the Large Hadron Collider (LCG-1) It has also been extended within the DZero experiment to use SAM as a production control database, to use associated prototype Grid environments there, and to do data reprocessing. It is now currently being introduced to the CDF experiment as well.

A pilot Runjob project was initiated in the Spring of 2003 in order to explore similarities between the then divergent DZero and CMS branches of MCRunjob. At the time, a common complaint from developers seeking to integrate MCRunjob with CMS and DZero systems was that the

MCRunjob code was not well documented and seemed ad hoc. This was true, and was largely owing to the high pressure, external milestone driven development environment of MCRunjob in each of the experiments. Therefore the pilot MCRunjob project was initiated to address these issues. It operated mainly by analyzing code and usage from each of the experiments' MCRunjob code bases, drew out the best of the code that was still in common, cleaned it up, and deposited it into a common CVS repository code named ShahKar.

Viewed in terms of these requirements, the pilot Runjob project was a success. A ShahKar CVS repository was established and a code librarian (Eric Wicklund) was assigned to maintain the code there and implement and run regular unit tests and package tests. Base classes abstracting the core functionality of DZero and CMS MCRunjob were written and stored there, including a BaseFramework, BaseParser, BaseConfigurator, BaseLinker, and ShahKarException class. The code is much better documented and adherent to design. And finally, an integration of the CMS MCRunjob and RunJob code was achieved in November 2003 and one is currently underway for the DZero experiment.

Proposal for a RunJob Project and Lessons Learned From the Pilot RunJob Project

The integration of the RunJob core code with the CMS MCRunjob package, though successful, involved illustrative headaches. It was found that if the integration effort was not mainstreamed into the experiments' planning, then integration was difficult to propagate. In fact, although a CMS MCRunjob integration with the RunJob core code has been achieved, it has not been propagated forward in CMS even though the effort to do so was available. The integration of the RunJob code with DZero code is on the other hand fighting two issues: it is not mainstreamed into DZero computing planning and the integration effort is continually reassigned to other tasks. (We are actually in a more fortunate situation with CDF since they are essentially starting from scratch with the RunJob project.) In order to achieve a lasting level of integration and usefulness, RunJob project planning will need to be more closely coordinated with the participating experiments in order to accommodate their milestones and requirements through effort applied to this project. Only then does it make sense for the experiments to support the RunJob project with effort and for the computing division to do the same.

The most obvious point to make is that there are now three experiments at Fermilab (DZero, CMS, and CDF) using or considering using MCRunjob flavored packages to aid in large scale Monte Carlo production and batch oriented data reprocessing or analysis on diverse sets of computing resources located worldwide. While each experiment expends some effort towards developing and maintaining its own set of solutions within this framework, each experiment basically has a similar set of requirements on such solutions. It is the proposal of this project to unite these efforts and harness them to create common solutions that will benefit all of the participating experiments at a long term savings of manpower.

The pilot RunJob project was structured passively in such a way that core development continued in each experiment, and developments deemed good were ported to the RunJob common code package, ShahKar. This had the advantage of being able to clean up significantly the respective experiment code and lay the foundations for a later follow on project while exploring the feasibility of the integration efforts. However, in addition to the above mentioned integration problems, lack of focus on the core project led to difficulties in internal planning as features were driven by the needs of independent experiments that did not talk to each other. These difficulties were nonetheless overcome through the excellent technical leadership of the experiment representatives: Anzar Afaq (CMS), Dave Evans (DZero/CDF), and Peter Love (DZero).

However, it is felt that a better project structure more closely aligned with the experiments and more proactive in gathering common requirements could greatly amplify these efforts.

For the RunJob project itself, we therefore propose a structural change from the pilot project that will lead to improved focus on core issues and responsibilities and better facilitate communication between the experiments as they seek to better use resources in common ways, including but not limited to better use of Grid technology. We therefore propose that the RunJob project should include both a core development effort that is getting requirements from the experiments and a dedicated integration effort that is driven by experiments' milestones. This is a fundamental shift from the pilot RunJob project which did not do core development (beyond keeping up with core development in experiments' RunJob variants) and which assumed that integration projects would be managed by the experiments.

In order to support this model, we will propose to keep the ShahKar¹ CVS module at Fermilab as the basis of core development. The RunJob code, as already developed in the RunJob pilot project, is packaged and distributed as tar files, RPMS, and UPS/UPD. We propose to reanalyze these distribution methods in the light of experiments' requirements again, and maintain these distributions within the RunJob project. The experiments will again maintain their specific variants of RunJob based upon the common distributed RunJob project code, but the integration task will be taken over by or shared with the RunJob project. RunJob project management will be independent of the experiments' regular line management but will answer to the experiments' software and computing top management on milestones and deliverables.

Common Requirements

The most general requirements on the RunJob project come from the experiences of running and managing large Monte Carlo processing jobs on diverse resources.

- (1) RunJob will have a modular architecture that will support building and submitting of jobs for a diverse set of computing environments. This modularity will include APIs that make extension to new environments straightforward. The environments will include Grid based environments and Web Services based environments.
- (2) RunJob will have a modular architecture that will support the description of various physics applications with metadata, including user written analysis applications. The metadata architecture will include APIs that keep a record of provenance, job tracking, and job resubmission.
- (3) RunJob will have a modular architecture that will support many different services sometimes used in job building, such as production control databases or external tracking databases. RunJob may provide working stubs for each of these services, but it is understood that the experiments may also have their own preferred solutions, in which case RunJob will communicate with those external services through its APIs.
- (4) RunJob will be able to chain physics applications together. For example, a RunJob job may consist of a tree rooted at Pythia generation of events followed by a GEANT simulation followed by two digitization runs at different pileup conditions each followed by ROOT tuple makers and finished by a program that compares the ntuples. (Such an arrangement consisting of two or more steps connected by data flow is called a "workflow" in the following.)

¹ It is estimated that the cost of changing the name "ShahKar" in the code itself would cost about a man week of effort. We propose to keep the name "RunJob powered by ShahKar" ala "Netscape powered by Gecko."

- (5) RunJob will contain primitives for expressing constraints on the workflow in a systematic way. For example, RunJob expresses data flow by constraining certain input specifications to be equal to certain output specifications. RunJob may also constrain some input parameters to come from a production control database and some from a user's private home area. Finally, collections of constraints (called contexts) may be managed centrally and be allowed to vary by physics group, execution environment or site, etc.

The existing ShahKar codebase from the RunJob pilot project satisfies all of these requirements already generically: ShahKar APIs exist to satisfy each of the general requirements above, but ShahKar does not include specific application descriptions, environments, services, or contexts. Specific instances of these components exist in each experiment's RunJob variant codebase.

Project Structure and Manpower

Though we feel that the above set of requirements adequately describes the requirements in general terms, these are still too abstract to be useful in setting goals for the project. A set of comprehensive meetings need to take place with representatives from the experiments' top computing management in order to agree upon a set of more concrete requirements, deliverables, and a schedule that is in sync with experiments' milestones. We consider this interaction to be critical to the success of the RunJob project. In order to deal effectively with these requirements and stick to the schedule, the following roles are proposed for the project.

- (1) Project Leader / System Engineer – The project leader and system engineer will be the primary point of contact between the RunJob project and the experiments. The Project Leader will be responsible for negotiating for and tracking effort on project and reporting back to the computing division and experiments periodically. As System Engineer, he is also responsible for negotiating features and requirements with the experiments and for translating these into specific code requirements that the developers can understand, and must therefore be knowledgeable about the system as well as the requirements.
- (2) Chief Architect – The Chief Architect is responsible for defining the components of the core product, defining the APIs, documenting the architecture, and making important technology choices internally. The chief architect would make decisions about merging or creating new subsystems. For example, the Chief Architect would be responsible for a Web Services decomposition of the product in the new Globus WSRF if there were a need for the project to move in that direction. The Chief Architect reports to the Project Leader / System Engineer and will assist in writing down the code requirements.
- (3) Technical Leader / Feature Engineer – The Feature Engineer is responsible for coordinating the pool of developers for achieving tasks assigned by the System Engineer in the requirements. The Feature Engineer is responsible for providing time estimates for specific implementations asked for in the requirements provided by the System Engineer.
- (4) Code Librarian / Chief Quality Engineer – The Code Librarian is responsible for maintaining the ShahKar CVS module and for maintaining development branches, merging branches, making prereleases, and releases. The Code Librarian is also responsible for packaging the released code using methods determined by the System Engineer in collaboration with the experiments. As Chief Quality Engineer, the Code Librarian is also responsible for unit and system testing. This may happen as part of the release process.
- (5) Technical Writing Assistant – A Technical Writing Assistant will be in charge of organization and some production of documentary materials and their presentation on the WWW.

- (6) Core Developers – Work with Feature Engineer to implement specific requirements on the RunJob core codebase, ShahKar.
- (7) Integration Developers – Work under the direction of the Feature Engineer with Core Developers to integrate or develop experiment specific extensions of the RunJob core codebase. This is a critical category because members will require both some knowledge of RunJob internals and APIs as well as of experiment specific applications and environments. Their place on the project is absolutely required, however, in order to facilitate integration and to keep integrations up to date without experiments slipping behind. The RunJob Project will ask for a pool of dedicated Integration Developers from each experiment. This category also acts as a safety valve in that experiments can add manpower here to quickly turnaround on contingency integration or experiment specific development that was not planned for earlier.

Requirements and Milestones Coming From the Experiments

At the time of writing, we have not yet initiated dialogue with the experiments' software and computing management over what specific milestones and deliverables to include. (Nor have we identified manpower.) The first act of this project should therefore be a phase of "requirement discovery" to begin the process of input into RunJob project. Although manpower estimates are given in this plan, it is understood that deltas against the given schedule will result in deltas in the manpower/schedule estimates as well. Nonetheless, the following loose feature requests were communicated by the experiment representatives on the pilot RunJob project.

Upcoming milestones in the CMS experiment include the finishing of Monte Carlo production for the CMS Physics TDR in late 2004. During this production phase, CMS will need to scale up dramatically the number of grid computing resources under its control to keep up with production needs and to keep up with LCG related grid computing milestones. Also, CMS will participate in the ARDA (Architectural Roadmap for Analysis) project among the four LHC experiments. Finally, the US Grids are exploring an Open Science Grid (OSG) to become the critical shared grid infrastructure for US science. The RunJob project will provide work on the following requirements

- (1) Support and maintenance for the core codebase of the CMS MCRunjob variant merged with ShahKar.
- (2) New feature development in the area of LCG computing environments.
- (3) Support for interoperability between LCG computing environments and existing US Grid environments.
- (4) Support for interoperability with the OSG environment.
- (5) Seamless support across transitions in underlying grid computing environments, such as upgrades in Globus or migration to new architectures such as WSRF.
- (6) Integration support to bring these new environments into mainstream CMS MCRunjob variant.
- (7) Integration support for POOL catalog accesses and dataset naming schemes.
- (8) Integration support for new user defined analysis applications in CMS using the COBRA framework and packages with DAR.
- (9) Integration support for CMS specific catalogs and the CMS production control database.
- (10) Integration support for CMS specific monitoring tools BOSS and RMT.
- (11) User interface support: CMS Analysis Specification Tool (CAST)
- (12) Integration support for Virtual Data Language production for Chimera. (This requirement is contingent upon GriPhyN or iVDGL support of integration developers.)
- (13) Integration support for Sphinx scheduling environment. (This requirement is contingent upon GriPhyN or iVDGL support of integration developers.)

The DZero experiment is a running experiment and as such has very tight schedules for features and requirements. An important milestone coming up is Grid Monte Carlo production using DZero RunJob and the DZero RunTime Environment (RTE). Also, in the summer of 2004, another data reprocessing run will take place. The RunJob project will provide work on the following requirements

- (1) Support and maintenance for the core codebase of the DZero RunJob variant merged with ShahKar.
- (2) Support and maintenance for interface to the DZero Grid environment or JIM.
- (3) Support and maintenance for interface to the DZero RTE on local farms and then in the DZero Grid environment.
- (4) Support for sandboxing and generic descriptions of jobs that can operate locally or at remote sites.
- (5) API for specifying and setting up an RTE.
- (6) API for specification of distribution of code to remote grid sites,
- (7) Integration support and maintenance for interface to experiment specific metadata catalog SAM. Produce and query SAM metadata.
- (8) Integration support and maintenance for interface to experiment specific production control database SAM (MC Tables).
- (9) User interface support and Macro preprocessor.

The CDF experiment as a running experiment also has significant computing milestones throughout the year. However, CDF also does not have an existing RunJob code base that has to be integrated. Many support and maintenance milestones are therefore straight-up development milestones. The RunJob project will provide:

- (1) Development of a core CDF RunJob variant (begun by Dave Evans) to replace existing farm control scripts. This is a very broad category and must be explored in more detail later with CDF computing management.
- (2) Development of a metadata handling system that can interface to SAM.
- (3) Development of a catalog interface that can interface to SAM to query datasets.
- (4) Development of dataflow tracking and management tools (File Meta Brokers)

One striking aspect of these requirement sets is how much they share in common: support for grid environments, support for diverse local runtime environments, metadata tracking for data intensive applications, metadata querying and insertion, and interfaces to new or existing systems such as LCG, SAM or the CMS Reference Database. Each experiment now devotes some amount of resources to solving these problems locally and ends up effectively duplicating some effort. By working together, some common aspects such as generic approaches to catalog lookups, farm environments, grid environments, and data movement strategies can be solved in common and reduced to an integration task per experiment. It also promises to be an excellent conduit for technology transfer between the experiments.

RunJob Architecture

(To be included later. For now, please see other documentation on the Web.)

Milestones and Deliverables

The pilot RunJob project produced a core code package called ShahKar. This pilot package has been integrated with the CMS MCRunjob tool, and integration with the Dzero RunJob tool is progressing. Therefore, the early milestones will deal with these integration projects. Once the

experiments' RunJob variants are integrated with ShahKar, then we will do all the significant feature development requested by the experiments in the ShahKar project itself and continuously support the integration of the experiments' variants with integration developers. The CDF experiment is in a unique position in that they have no pre-existing RunJob variant and can begin developing with ShahKar right away. This early phase of intensive integration will coincide with a period on requirements gathering. All milestones assume on project manpower unless otherwise noted.

Before any list of specific milestones can be taken on by the project, it is important to get buy in from the experiments on a general set of requirements as listed above and to enter into an intensive period of early negotiation to agree upon a list of specific deliverables and an extended period of ongoing negotiation over new features and to feed back on existing features. That period of negotiation has not yet occurred, and therefore we have broken the project up into two phases: an "Early Integration Phase" during which the experiments learn about, integrate or develop with the core code from the RunJob pilot project and a longer term "Development and Continuing Integration Phase" where the bulk of the core development occurs. During the earlier phase, the experiments and the project should develop a suitable list of tasks together that will achieve the relevant milestones. The main deliverable of the early phase is the negotiation of an initial set of requirements.

The following list of later development milestones is a guess based upon the authors' current guesses as to the requirements of the experiments. These are probably pretty good guesses in themselves, but as a list they are probably incomplete. Nonetheless, many of the requirements can be mapped to specific items in the list.

Early Integration Phase Milestones

Milestone	Date	Comment
Final RunJob Project Web Page Designed and Available	3/15/05	Project Webpage including project plan, management documents, instructions for use, architectural documents, and other documentation.
Unit Tests Module Complete	3/31/04	A suite of comprehensive tests to be run before every major or minor release.
CMS Switchover to RunJob core	4/15/04	The CMS MCRunjob package, which has already been integrated with the RunJob core package, will be distributed with RunJob core by default.
Dzero Integration with RunJob core	4/15/04	The Dzero RunJob package will be integrated with RunJob core package. This means that essential services and base classes will be coming from RunJob core.
CDF Initial Development with RunJob core	4/31/04	CDF will develop an initial job creation, submission, and farm control facility using RunJob core to replace legacy farm scripts.
Requirements Negotiation Finished	5/15/04	All experiment specific requirements are gathered and milestone integration timetables finalized.
Dzero Switchover to RunJob core	6/15/04	The Dzero RunJob package will be distributed by default with the RunJob core package.

Later Development and Continuing Integration Phase Milestones

Milestone	Date	Comment
Design and Implement scriptObject architecture	3/31/04	The architecture of scriptObjects is finalized and implemented. ScriptObjects provide a layer of abstraction between abstract job descriptions and concrete executable jobs that greatly enhances portability between grid and execution environments,
Context mechanism prototyped and implemented	4/7/04	The RunJob core contexts will be prototyped and implemented without algebraic models for context composition. Contexts provide compact and portable descriptions of constraints relating to environments, such as site/grid constraints, application constraints, or logical (physics group) constraints.
Design and Implement Configurator Addressing	4/15/04	The architecture for interconfigurator referencing, naming, and dependencies is finalized and implemented. This is needed in order to propagate constraints and internal events. The current architecture is adequate but not extensible, and we would like to add user defined metadata to the Configurator descriptions.
File Meta Broker Design and Implemented	4/21/04	Common interface for specification of file transfers and translating these into environment appropriate mechanisms for file transfer. FMBs will be associated with workflow arrows tagged to correspond with dataflows and will be generated automatically. FMBs provide a layer of abstraction that allows dataflow to be expressed the same way in environments that support transparent data movement as in environments that need data movement to be specified explicitly. Specific FMBs can be chosen dynamically when context is resolved, transparent with respect to the user.
XML Specifications	4/21/04	XML Specifications of all key components: Configurator, Linker, scriptObjects, etc. This has been chosen as a way to exchange information with outside tools. CAST (CMS Analysis Specification Tool) is a GUI for RunJob core based variants that will communicate with core services using XML.
Support for Dynamic Framework Ordering	4/31/04	Framework calls can optionally have dependencies to specify an ordering instead of the current flat list. RunJob core has a framework driven architecture. Currently frameworks are specified in a default or user provided list. Interactions with new external services are usually represented internally by a framework call. In order to facilitate introduction of new services into a RunJob core framework, the representation should support dependencies among framework calls and allow for dynamic ordering. (Otherwise, user has to become expert and choose correctly the entire framework.)
Design and Implement Final RunJob core syntax	5/7/04	The final language syntax for the RunJob core base package is agreed upon and implemented. Users typically interact via a macro script that interprets macro commands line by line into calls on the RunJob

		core API to build jobs or contact services. The macro syntax itself is currently a mess and needs to be brought under control. (Explore using plex, a Python lexical analyzer, for this purpose.)
Common Batch Systems	5/15/04	Support for common batch systems like PBS, FBS, or LSF. This is a “no brainer” and already exists in each RunJob variant. Therefore this is an integration milestone that will make sure that the environment
RunJob core Release – I	5/15/04	First Major Release, RunJob core 1.0. All experiments (CDF, CMS, DZero) should have basic support for their local farms, applications, and services.
LCG-2 Environment	5/21/04	Support for the LCG-2 Environment. This milestone leverages CMS integration work needed by CMS external milestone for use in RunJob core.
User Interface Prototyping	6/1/04	Leverage CAST work for RunJob core. This milestone leverages work done in CMS for CAST for the use of the CAST GUI for all experiments. CAST is an Iris explorer style GUI that allows users to build workflow/dataflow diagrams with a click and drag interface and generate macro scripts to achieve them.
Web Services Architecture	6/1/04	A Web Services decomposition of RunJob core services has been designed. This milestone
Context mechanism finalized and implemented.	6/7/04	Contexts finalized with algebraic models for composition. This is a feature introduced into the core package that will enhance the specification of constraints already found in the more basic concept of contexts. While contexts can be composed informally now by a simple Cartesian product and shadowing, this functionality will guarantee users ignorant (possibly willfully so) of local execution contexts that their own private contexts can be used safely in that environment or conflicts reported. . Contexts will be used to organize and hide input parameters, so this is important to the user that has to transparently manage hundreds of input parameters without knowing all of them explicitly,
Migration to RunJob core 1.0	6/15/04	Supported experiments migrate to RunJob core 1.0. This milestone is special contingency needed to upgrade each experiment to a new major release.
Support for Metadata Flow Models	7/1/04	Support for multigraph constructs explored in GGF10 submission on workflow builders. This is a feature introduced into the core package that will enhance the specification of constraints already found in the concept of contexts. The multigraph construct allows for the design of more general constraint sets and algorithms for resolving those constraints. Constraints will be used to organize input parameters, so this is important to the user that has to manage hundreds of input parameters,
Architecture for fault tolerant operation and job tracking.	7/7/04	An architecture for fault tolerant job creation, tracking, running, and resubmission. This milestone will address fault tolerance in a very narrow way: by supporting a minimum level of job tracking needed to support job resubmission. External databases will be

		used where possible (SAM, BOSS, RefDB, etc.) but an interface must be specified in RunJob and a stub tracking system provided with limited functionality.
Runtime architecture implemented	7/15/04	Support for runtime RunJob core architecture, including job monitoring instrumentation. (See Dave Evan's shREEK architecture.)
Web Services Implementation	7/30/04	Implementation of Web Services model. This milestone assumes that we are also able to find facilities resources to keep the web services alive.
Integration with External environments: GAE	8/7/04	Integration and support for ROOT clients based on CLARENS. This milestone leverages work done in CMS to bring a RunJob User Interface into a ROOT environment. This would support batch style analysis job specification directly from the familiar ROOT shell environment. (CLARENS is a web services standard that brings generic web services into a ROOT environment. This depends upon the Web Services decomposition.)
Support for Grid Operations in RunJob core	8/15/04	Support for job submission using the Web Services Resource Framework (WSRF). This will enhance grid job submission and bring us up to date with GTK4. (The WSRF and GTK4 are not expected until Fall 2004.)
MOP Integration	8/15/04	Integration of the MOP environment into ShahKar. This will leverage work done by PPDG in the CMS experiment to bring in generic grid submission services using bare minimum Condor-G/DAGMan and Globus 2.4 era gatekeepers.
RunJob core Release - II	8/15/04	Second Major Release, RunJob core 2.0. This release should have full support for a variety of generic grid environments and enhances user interfaces for managing metadata and constraints. Continued development in the experiments dealing with new applications will be included as well.
Common Interface to catalogs	9/1/04	Support for interface into external catalogs for metadata input and file inputs. (While we expect that RunJob will support many such interfaces already by this time, a common architecture will be developed and implemented.) This could allow for communication between different catalogs, such as SAM and the CMS RefDB.
User Interface Finalized and Implemented	9/7/04	User interface is implemented and finalized, both GUI and non-GUI. This will be the final integration of the CAST Tool into the RunJob core.
Migration to RunJob core 2.0	9/15/04	Supported experiments migrate to RunJob core 2.0. This milestone is special contingency needed to upgrade each experiment to a new major release.
Support for simple job tracking in RunJob core	10/1/04	A simple job tracking mechanism or module that can support job resubmission. We expect that this functionality will already be in place in each experiments' codebase, but a common architecture for doing job resubmission in the RunJob core will be finalized and implemented here.
Fault tolerance implemented	10/15/04	The architecture for fault tolerance is implemented and integrated with other systems like the simple job

		tracker and provenance. The fault tolerance needs of the experiments must be discussed. For the purposes of RunJob core, we assume that this means the a user is made aware of job failure and options for job resubmission or cancellation are presented.
Provenance tracking in RunJob core	10/21/04	Provenance and meta-provenance to be recorded in XML database. Includes clients to browse provenance. Provenance for example will track input parameters for each application and meta-provenance in addition will track the constraints. This provides useful information not only about raw values, but also about why the raw values are set the way they are.
RunJob core Release - III	10/21/04	Third Marjor Release, RunJob core 3.0. This release will have the final integrated
Integration with External Environments: GriPhyN	11/1/04	Provide VDL support and Sphinx environment support. This will leverage existing CMS ability to work with the Sphinx scheduler and generate VDL for Chimera. This milestone depends upon support from the GriPhyN or iVDGL project.
SC2004 Demonstrations	11/15/04	SC2004 Demonstrations
Migration to RunJob core 3.0	12/15/04	Supported experiments migrate to RunJob core 3.0. This milestone is special contingency needed to upgrade each experiment to a new major release.

Opportunities for Collaboration

The GridPP project in the UK and the PPDG project in the US are important contributors to the pilot RunJob project. GridPP provided much of the integration and support effort on the Dzero side, while PPDG provided experience and expertise on VDT environments. It is a GridPP milestone to have users run analysis on SamGrid by 9/1/2004. And GridPP2 has just been approved for 3 years. The establishment of a regular well defined interface for job building on top of different local and grid environments presents an exciting opportunity for collaboration with outside Computer Science groups wishing to work directly with the experiments. The RunJob project essentially provides interfaces that the outside CS groups can leverage in order to test new environments with existing experiment code.

Of course, the RunJob Project itself will not collaborate directly with any outside Grid group on its own, but will do this instead through the experiments at the direction of the experiments' software and computing management. The RunJob project exists to serve the needs of the experiments and not to help external projects do R&D. However, if R&D effort is provided by outside CS groups to the pool of Integration Developers, then they can effectively work with all participating experiments at once. (For example, the milestones above include a "Virtual Data Language" milestone. However, this milestone is left unfunded unless effort from GriPhyN or iVDGL is found.)

Manpower Estimates

The milestones listed above, not including management overhead, require an estimated 40 man months of effort². Of that, approximately 10 man months of effort is dedicated maintenance and integration tasks, 8 man months is additional boost required at the beginning to achieve integration or initial experiments' RunJob variants, and 22 man months for feature planning and development. Over a nine month project span, this corresponds to an annualized level of effort (LOE) of 4.4 FTE. In addition to this figure, an estimated 0.8 FTE (LOE) of project management and 0.8 FTE (LOE) for core release management, unit testing, and technical writing, 0.6 FTE (LOE) for dedicated on-project "end to end" testing and 0.6 FTE (LOE) integration testing experts from the experiments is needed.

The effort sources listed below are coming from the RunJob project itself and the experiments. The RunJob project itself will ask for 2.4 FTE (LOE) from the Computing Division itself plus 2.4 FTE (LOE) (0.6 FTE per participating experiment) for project management, release management, testing, end to end testing, and core development. An additional 2.4 FTE (LOE) (0.8 FTE per participating experiment) is asked for to support directly on-project the integration of the RunJob core architecture into the experiments' infrastructure. We estimate that most of the core development effort could go away after the end of this project, depending on whether or not additional features are requested or not during the course of the project.

Role	FTE Estimate (LOE for 9 month schedule)	Sources	Roles
1. Project Management	0.8	RunJob project	PL/System Engineer
2. Release Management and Unit Testing	0.8	RunJob project	Code Librarian and Technical Writer
3. Feature Testing, End to End Testing	0.6	RunJob Project	Quality Assurance Engineer
4. Core Architecture, Planning, and Development	2.0	RunJob project	Chief Architect, Feature Engineer, Core Developers
5. Integration Development	2.4	CMS, Dzero, CDF	Integration Developers
6. Integration Testing	0.6	CMS, DZero, CDF	Special End Users, Testers

(Note: "LOE" is annualized FTE. To get actual FTE in man years, multiply the LOE by schedule length in years.) The distribution of manpower throughout the year will be a bump in the beginning to achieve integration followed by a lower level of sustained effort. However, this also depends upon the outcome of the requirements and negotiation phase with the experiments.

Effects of Schedule Changes

A first order formula for gauging the effect of schedule changes is as follows. The Level of Effort for the management tasks (lines 1-2) is assumed to be constant. The Level of Effort for the development and testing tasks scales (lines 3-6) to first order with the schedule length. Thus, the

² Each feature or milestone above was given a SWAG estimate. These were summed and the final result was inflated by 25% for contingency.

formula for Level of Effort is given by $1.6 + 4.2 / (\text{length})$ where length is the desired schedule length in years. However, it is not yet analyzed how much the LOE could be decreased while maintaining the usefulness for the experiments. (We haven't seen yet if this causes "critical" milestones to slip.)

Schedule Length	FTE (Level of Effort)
0.75	7.2
1.0	5.8
1.25	5.0

Conclusion

The RunJob project will be an important project in the Computing Division because it will draw together and address common issues in three large experiments at the lab. With an initial cost of developers to do integration and additional features development throughout the nine month schedule, the RunJob project will provide a common system for job creation for the CDF, CMS, and DZero experiments based on the existing RunJob core (ShahKar) product developed in the RunJob pilot project. RunJob will help each experiment adapt their applications to diverse resource environments and provide services to help track jobs, control production, or access data catalogs. Although specific modules needed to address specific environments at the experiments will still live in their respective RunJob variants, the task of integrating these with RunJob core will be managed in the RunJob project itself. Communication with the experiments is a significant feature of this project plan, and is perhaps the most important factor.

It also presents the exciting feature that Grid environments used in each experiment can be made accessible to the others and made interoperable at a high level. The RunJob project can become a place (in addition to facilities planning going on elsewhere and the important grid architectural planning going on in GridPP and PPDG) where common planning of approaches to Grid technology can be done, especially in the user interface. The Fermilab Computing Division is a logical place to start this effort both because three big collider collaborations are based at Fermilab (CDF, DZero, and USCMS) and because of the history that two of them already have with RunJob development.