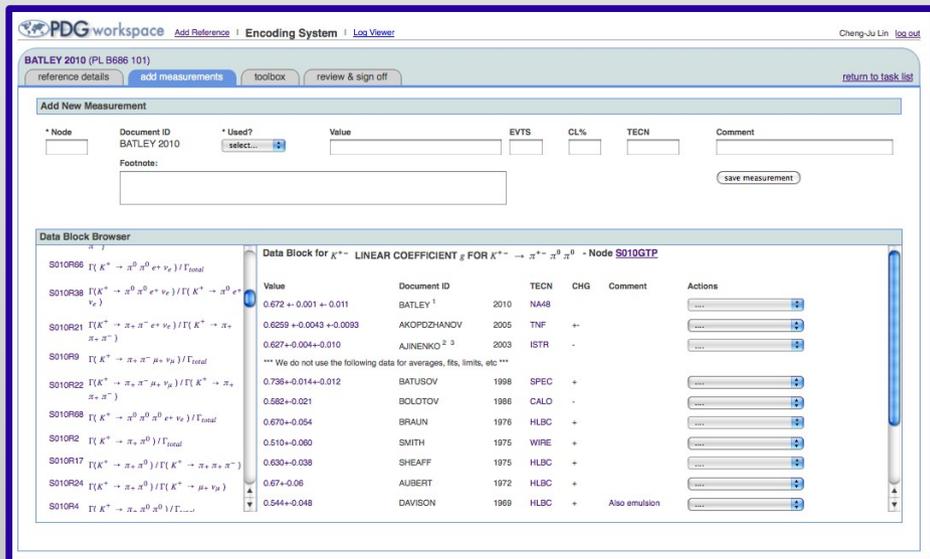


Overview of the PDG Computing Upgrade

Juerg Beringer
 Physics Division
 Lawrence Berkeley National Laboratory



The screenshot shows the PDG workspace interface. At the top, there are navigation links: "Add Reference", "Encoding System", and "Log Viewer". Below this, the "BATLEY 2010 (PL B686 101)" section is visible, with tabs for "reference details", "add measurements", "toolbox", and "review & sign off".

The "Add New Measurement" section contains a form with fields for "Node", "Document ID" (set to "BATLEY 2010"), "Used?" (a dropdown menu), "Value", "EVTS", "CL%", "TECN", and "Comment". There is also a "Footnote:" field and a "save measurement" button.

The "Data Block Browser" section displays a table of data blocks. The table has columns for "Value", "Document ID", "TECN", "CHG", "Comment", and "Actions". The data is filtered for "Data Block for $K^+ \rightarrow \pi^0 \pi^0 e^+ \nu_e$ LINEAR COEFFICIENT g FOR $K^+ \rightarrow \pi^+ \pi^0 \pi^0$ - Node S010GTP".

Value	Document ID	TECN	CHG	Comment	Actions
0.672 ± 0.001 ± 0.011	BATLEY ¹	2010	NA48		
0.6259 ± 0.0043 ± 0.0093	AKOPDZHANOV	2005	TNF	+	
0.627 ± 0.004 ± 0.010	AJINENKO ^{2,3}	2003	ISTR	-	
*** We do not use the following data for averages, fits, limits, etc ***					
0.736 ± 0.014 ± 0.012	BATUSOV	1998	SPEC	+	
0.582 ± 0.021	SOLOTOV	1986	CALO	-	
0.670 ± 0.054	BRAUN	1976	HLBC	+	
0.510 ± 0.060	SMITH	1975	WIPE	+	
0.630 ± 0.038	SHEAFF	1975	HLBC	+	
0.67 ± 0.06	AUBERT	1972	HLBC	+	
0.544 ± 0.048	DAVISON	1969	HLBC	+	Also emulsion

Outline:

- Introduction
- Challenges and project strategy
- **Major success: V0 Release**
- Development, documentation, ...
- Status and plans

- **Obviously:**
 - Efficiently **managing hundreds of people** and
 - **producing a book of 1,400+ pages**
 - summarizing >30,000 measurements from >7,000 papers
 - every 2 years (with intermediate web update),
 - supporting different print and online editions**requires an adequate computing system**
- **Yet presently used PDG system dates back to late eighties and can no longer handle requirements without great risk**
- **Urgency of a computing upgrade and need for additional resources to carry it out were widely recognized by reviewers**
- **Developed plan for PDG computing upgrade and asked DOE (and NSF) for funding**

Written in 2006

High-Level Requirements and Roadmap for PDG Computing

Juerg Beringer
Particle Data Group
Lawrence Berkeley National Laboratory

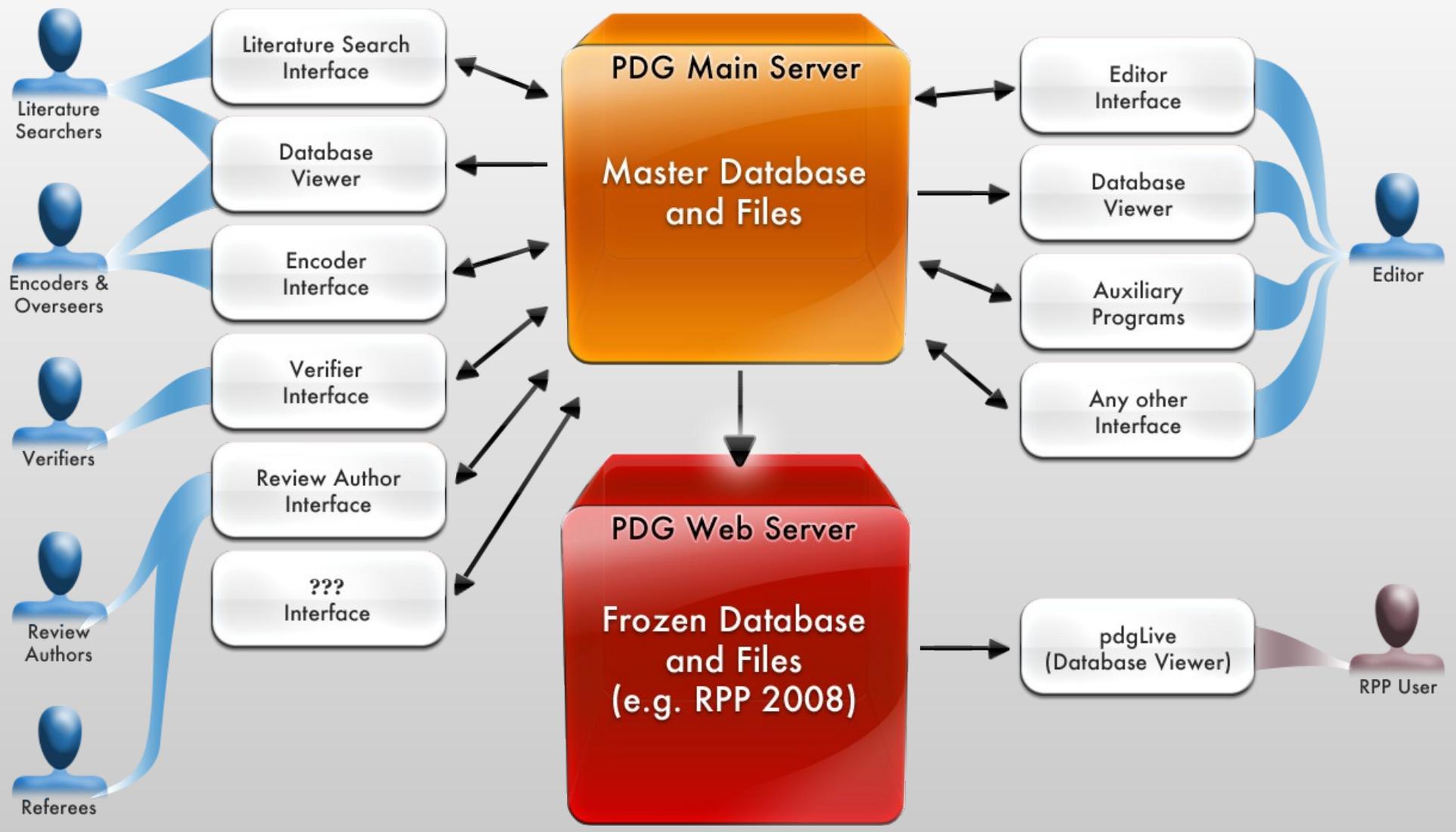
This document summarizes the high-level requirements for the upgraded PDG computing system and proposes a roadmap for completing the upgrade. It is intended to serve as a starting point for a cost estimate for the completion of the upgrade project.

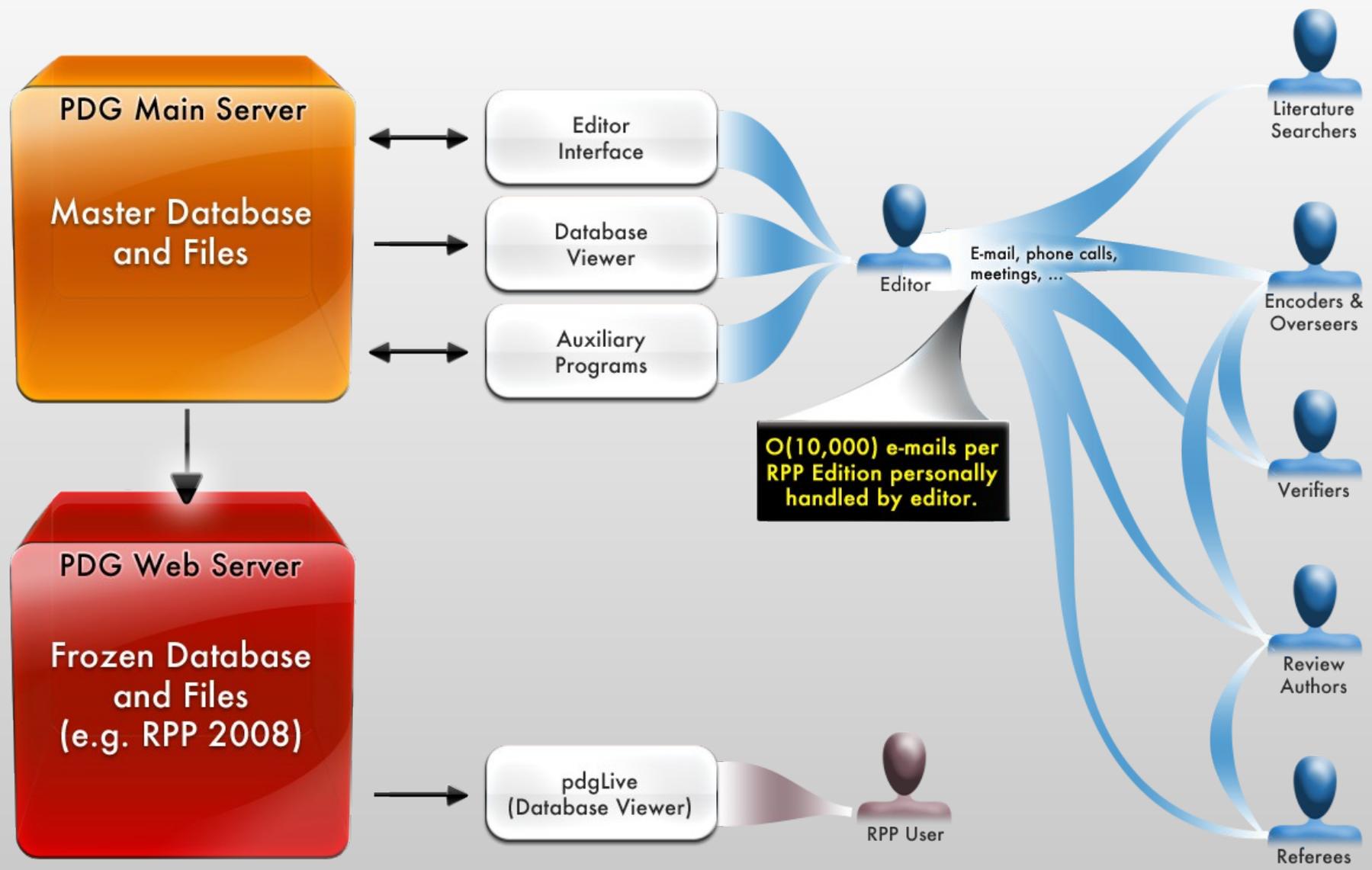
- **Comprehensive DOE review of PDG in September 2008**
(<http://pdg.lbl.gov/doereview/agenda.html>)
 - **Vital role of PDG is reaffirmed**
 - “The PDG publications are crucial to the field ...” (DOE reviewer)
 - **DOE asked us to increase our request for resources for the computing upgrade to ensure we will succeed**
 - Now 2 FTE for 3 years (until end of FY11)
 - 0.5 FTE for ongoing support after initial development
- **NSF agreed to contribute to the computing upgrade according to its overall share of PDG funding**
 - Grants PHY-0652989 and PHY-0966691
- **Development in full swing by end of 2008**

Today we will discuss what we have achieved during the first ~half of the computing upgrade project

- **A modern, modular, extendable, easy-to-use, maintainable and well-documented **computing infrastructure for PDG****
- **Production quality system – PDG data must be correct**
 - Extensive error-checking and cross-checking built into system
- **Support all areas of our work, including in particular:**
 - Decentralized, web-based data entry and verification for Listings
 - Interaction with over 100 review authors
 - Monitoring of progress in RPP production
 - Programs for evaluation of data (fits, averages, plots, ...)
 - Expert tools for editor, including creation of book manuscript and static web pages (PDF files)
 - Interactive browsing of PDG database similar to pdgLive

Details and status of system components will be discussed in the subsequent talks





- **PDG has special requirements that cannot be addressed by “commodity software”**

Solution:

- Identified challenging areas posing potential risk to project
- Carefully addressed these areas **first** (through **design, technology choices, and project planning**)

- **Computing upgrade must proceed in parallel to PDG work**
 - Legacy system must continue to run during development
 - Severely limits opportunities for system deployment (once per year)
 - Workload on PDG experts from having to work with two systems

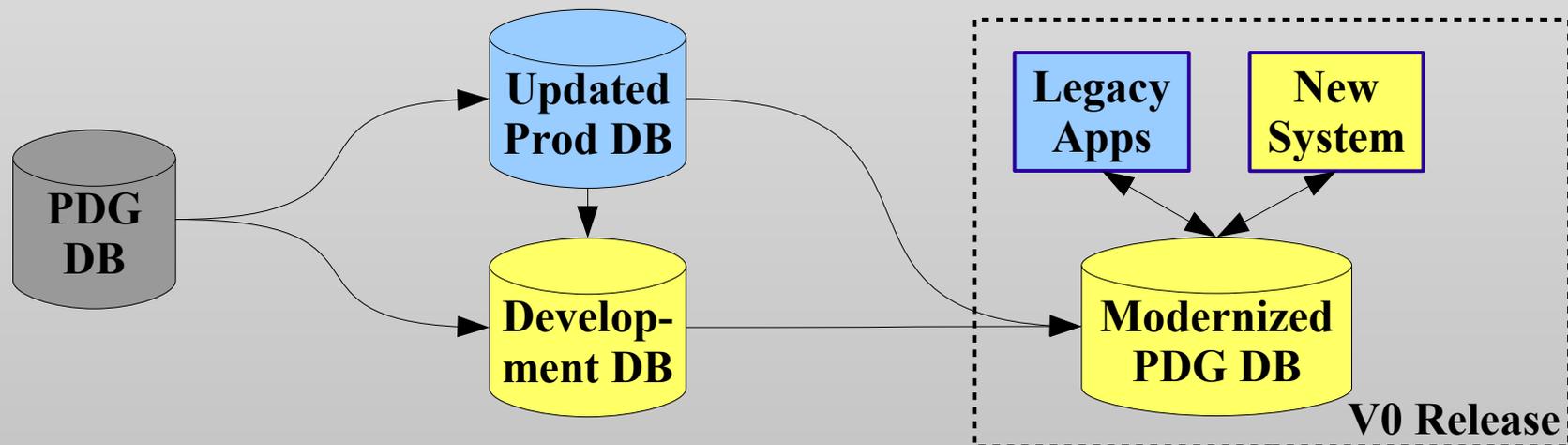
Solution:

- Must carefully plan new system deployment
- Release as early as possible with legacy applications running within new system (“V0 Release”, see later)
- Allows incremental deployment of new components

- Existing scientific data must be migrated to new system
 - Complete redesign of PDG database from scratch impractical from many points of view
 - Changes to PDG database must be made incrementally
 - Small database changes mandated by ongoing PDG work
 - Conventions on how data is stored in the database (macros, flags, etc)
 - Occasionally need new columns in tables

Solution:

- Modernized PDG database used by both (updated) legacy applications and the new system



- **Scientific output from old and new system must be identical; PDG data must be correct**
 - Inherently difficult to validate tens of thousands of numbers

Solution:



- Nightly builds with unit tests
- Careful and detailed validation before use for PDG production
- Detailed logging of changes at database level
- Version control of database contents by dumping to CVS
- **System validation by producing TeX manuscript of full Review in old and new system, then making sure all changes (“diff”) are expected and desired**

- **Distributed data entry**
 - System must take care of complicated distributed work flow
 - Detailed logging of changes (“Why did this number change?”)

Solution:



- Careful design
- Suitable industry-standard technology choices (J2EE)
- Innovative logging scheme using database triggers that keeps track of logical operations and enforces logging at database level for any application (doesn't need any application specific logging support)

- **Use of TeX and display of math on the web**

Solution:



- Evaluate existing solutions (MathML, jsMath, mimeTeX, TeX-to-MathML translators, ...)
- Found solution that addresses our needs (see Sarah's talk)

- **Browser and platform diversity among large user base**

Solution:

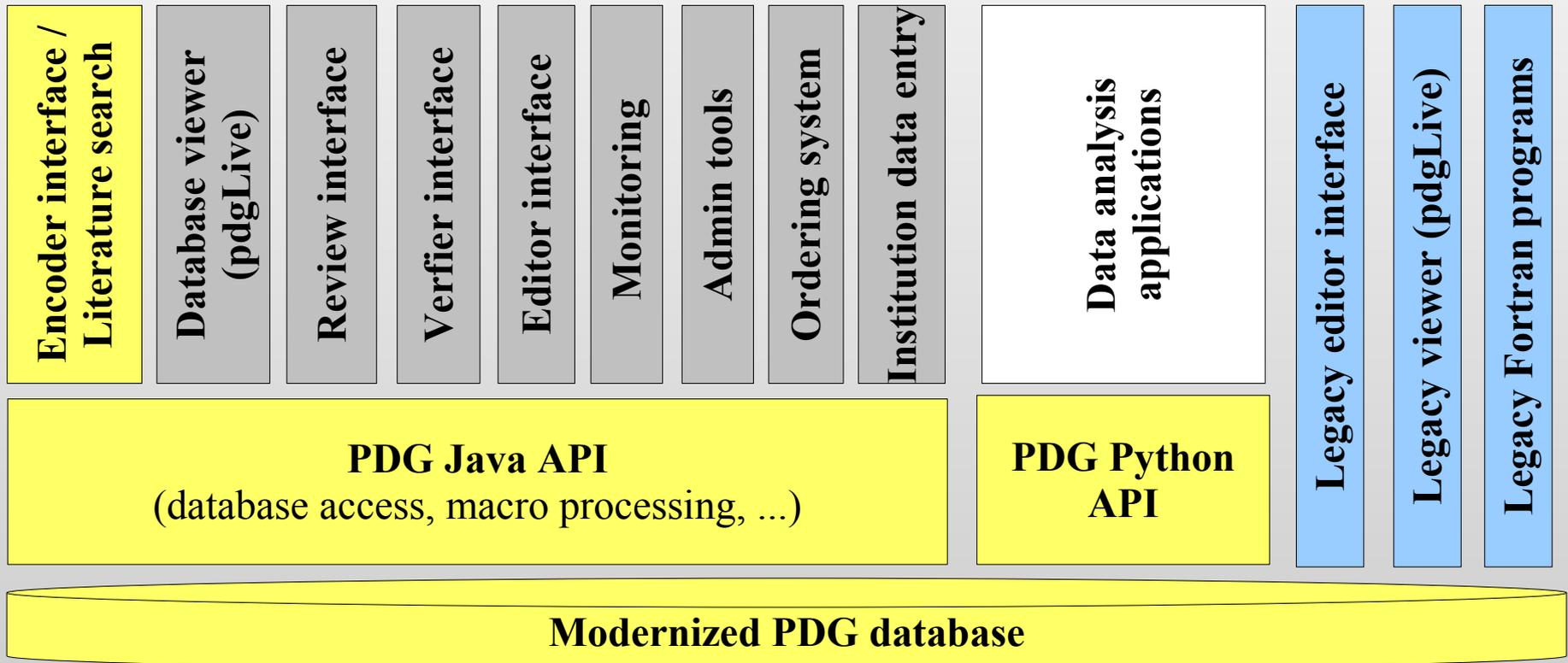


- Use existing extensive JavaScript library where this problem is already solved (see Sarah's talk)

- **The V0 Release is the backbone of the upgraded system**
 - It's key ingredient is the **modernized PDG database**
 - All technologies of new system included & working (full vertical slice)
 - **All challenging areas addressed**
- **All (updated) legacy applications run in V0 Release system**
 - Thus it is a complete and fully functional **production release**
 - Validated and has become **current PDG production system**
- **Provides a modular framework into which applications can be easily and incrementally included (during ongoing PDG work)**
- **Includes alpha release of the encoder interface**
 - **By far most difficult and complex application**
 - Includes the main building blocks required by the other applications
 - Supports complete standard encoding cycle plus advanced tools

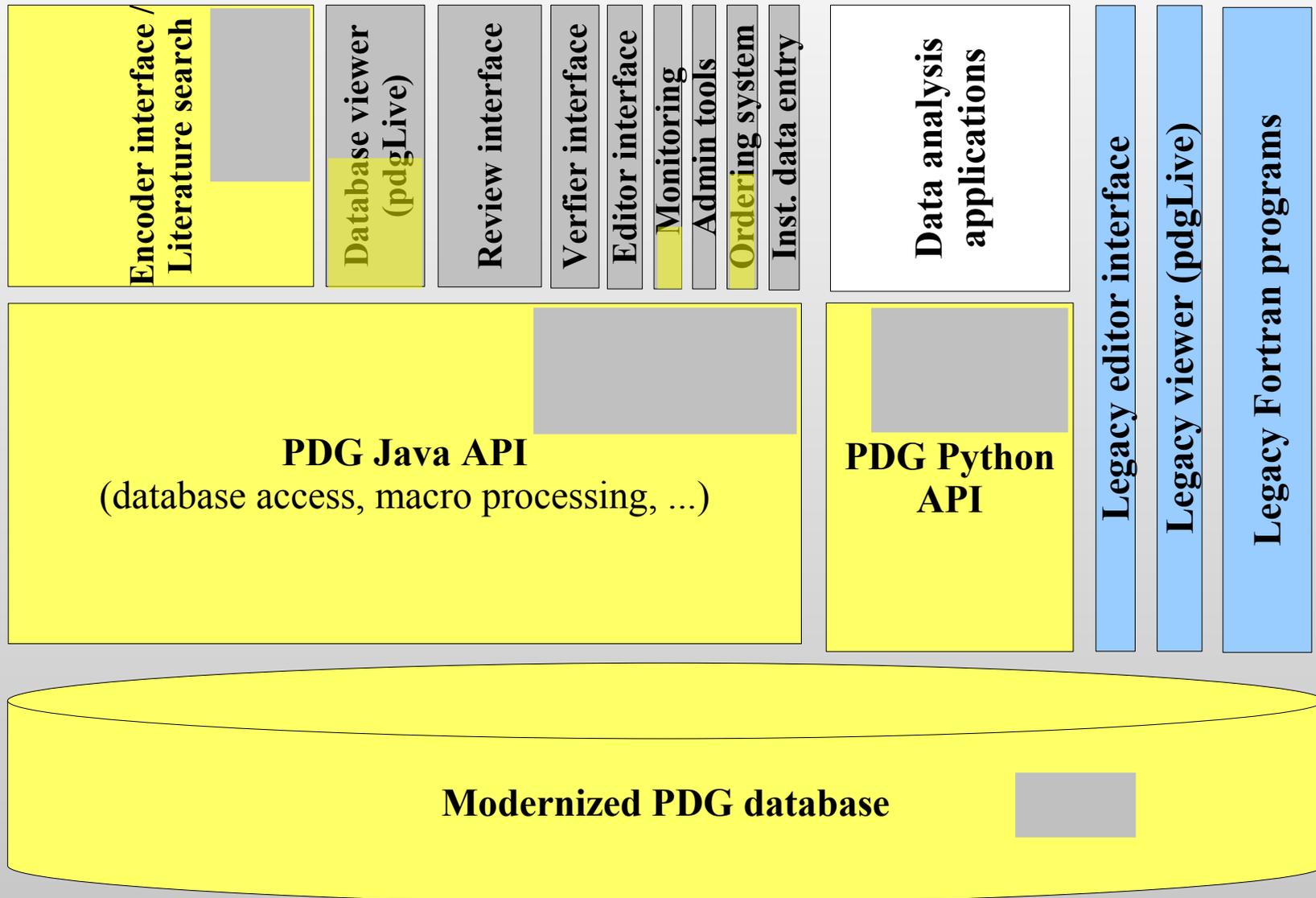
Successfully deployed August 11, 2010

- = updated legacy applications (in V0 release)
- = new components included in V0 release
- = still to be implemented as part of upgrade (some partly done)



- **Encoder interface includes building blocks for remaining applications**
- **Python-based API for data analysis also included**

- Rescaled diagram to reflect approximate development effort



- Entering a measurement through the encoder interface
 - Note: the encoder interface includes the building blocks needed for putting together the remaining applications!

PDG
Workspace

The screenshot shows the PDG workspace interface for 'BATLEY 2010 (PL B686 101)'. It features a navigation bar with 'reference details', 'add measurements', 'toolbox', and 'review & sign off'. Below this is the 'Add New Measurement' form with fields for Node, Document ID, Used?, Value, EVTS, CL%, TECN, and Comment. The 'Data Block Browser' section displays a table of data blocks for the linear coefficient g for $K^{*+} \rightarrow \pi^+ \pi^0 \pi^0$. The table includes columns for Value, Document ID, TECN, CHG, Comment, and Actions.

Node	Value	Document ID	TECN	CHG	Comment	Actions
S010R66	$\Gamma(K^+ \rightarrow \pi^0 \pi^0 e^+ \nu_e) / \Gamma_{total}$					
S010R38	$\Gamma(K^+ \rightarrow \pi^0 \pi^0 e^+ \nu_e) / \Gamma(K^+ \rightarrow \pi^0 e^+ \nu_e)$	BATLEY ¹	2010	NA48		
S010R21	$\Gamma(K^+ \rightarrow \pi^+ \pi^- e^+ \nu_e) / \Gamma(K^+ \rightarrow \pi^+ \pi^-)$	AKOPDZHANOV	2005	TNF	+	
S010R9	$\Gamma(K^+ \rightarrow \pi^+ \pi^- \mu^+ \nu_\mu) / \Gamma_{total}$	AJINENKO ^{2 3}	2003	ISTR	-	
*** We do not use the following data for averages, fits, limits, etc ***						
S010R22	$\Gamma(K^+ \rightarrow \pi^+ \pi^- \mu^+ \nu_\mu) / \Gamma(K^+ \rightarrow \pi^+ \pi^-)$	BATUSOV	1998	SPEC	+	
S010R68	$\Gamma(K^+ \rightarrow \pi^0 \pi^0 \pi^0 e^+ \nu_e) / \Gamma_{total}$	BOLOTOV	1986	CALO	-	
S010R2	$\Gamma(K^+ \rightarrow \pi^+ \pi^0) / \Gamma_{total}$	BRAUN	1976	HLBC	+	
S010R17	$\Gamma(K^+ \rightarrow \pi^+ \pi^0) / \Gamma(K^+ \rightarrow \pi^+ \pi^+ \pi^-)$	SMITH	1975	WIRE	+	
S010R24	$\Gamma(K^+ \rightarrow \pi^+ \pi^0) / \Gamma(K^+ \rightarrow \mu^+ \nu_\mu)$	SHEAFF	1975	HLBC	+	
S010R4	$\Gamma(K^+ \rightarrow \pi^+ \pi^0 \pi^0) / \Gamma_{total}$	AUBERT	1972	HLBC	+	
		DAVISON	1969	HLBC	+	Also emulsion

Math
display

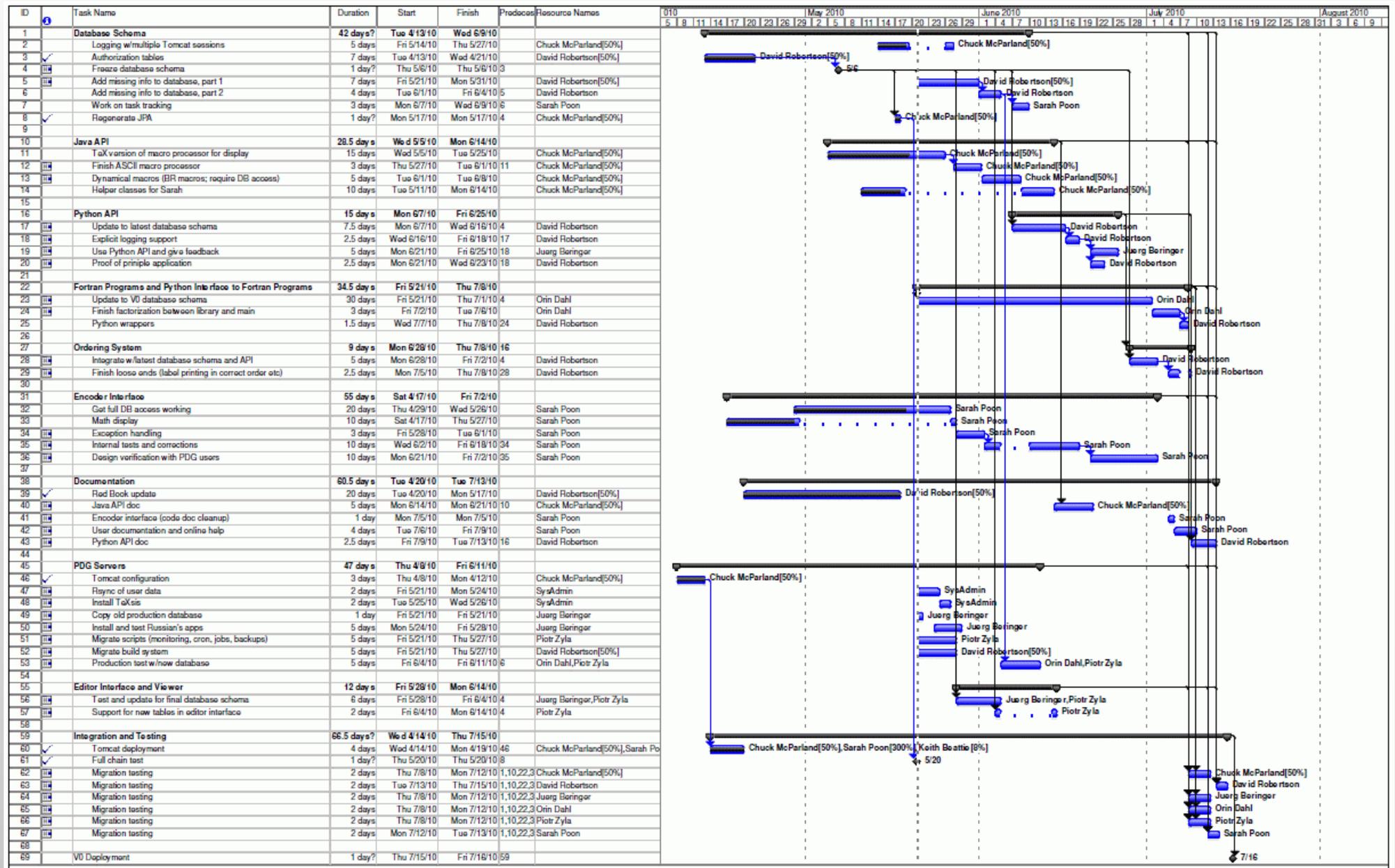
Display of data
block (→pdgLive)

- **Juerg Beringer (PDG physicist)**
 - Project leader, requirements, system architecture
- **Chuck McParland (computer scientist)**
 - Java API
- **Sarah Poon (computer systems engineer)**
 - Web design, user interfaces, JavaScript
- **David Robertson (computer systems engineer)**
 - Database, Python API, scripts
- **Orin Dahl (PDG physicist, retired)**
 - Legacy Fortran programs
- **Piotr Zyla (PDG editor)**
- **Contributions from Jacob Andreas, Cecilia Aragon, Keith Beattie, Igor Gaponenko, Keith Jackson, Kirill Lugovsky, Slava Lugovsky**

Each member of the team has many years of software development experience

- **Follows widely-adopted practices, including**
 - Iterative design process with close interaction with users
 - Ongoing documentation (Wiki, within code, formal manuals)
 - Nightly builds and nightly unit tests
 - Using existing tools, components and libraries to maximize efficiency
- **Frequent communication**
 - Weekly general meetings
 - Weekly individual meetings of developers with project leader
 - Additional meetings as needed
 - Mailing list
- **Close involvement of PDG members**
 - So far through Orin, Piotr and myself (plus occasionally Cheng-Ju Lin and Weiming Yao)
 - As user testing ramps up, will increasingly involve other members of LBNL PDG group plus selected members from PDG collaboration

Detailed Project Planning



- **Computing TWiki**



- **Manuals (in particular RedBook)**

14

ENCODING			
ID ^k	INT		Unique identifier used for logging changes to the database.
REFERENCE_ID ^f	INT	5	The identifier of the reference to be encoded. Foreign key to REFERENCE table.
PAR_CODE ^f	CHAR	4	The code of a particle studied in the reference. NULL means all particles. Foreign key to PARTICLE table.
PAR_PROPERTY ^k	CHAR	10	The property of the particle studied (measured) in the reference. NULL means all properties.
CONTENTS	CHAR	1	Whether or not the paper contains any encodable information. NULL Papers whose status is unknown. D Papers that contain data that is in the database. E Papers that have no encodable information.
PUBLISHED	CHAR	1	Whether or not the paper was in the database as of the last publication. NULL Paper was not in the database as of the last publication. P Paper was in the database as of the last publication.
STATUS	CHAR	1	Foreign key to STATUS table. Code for status of the task.
ASSIGN_STATUS	CHAR	2	Indicates several states that a paper can be in, in the case of overseeing and encoding U Paper is unassigned. UE There is an unassigned encoder but an assigned overseer. UO There is an unassigned overseer but an assigned encoder. A Both the encoder and overseer have been assigned.
FINDER	CHAR	10	The name of the physicist who performed the first literature search.
ENCODER	CHAR	10	The name of the team associated with the encoder. There can be single person teams.
OVERSEER	CHAR	10	The name of the team associated with the overseer.
COORDINATOR	CHAR	10	The name of the team associated with the coordinator.
DATE_ENTERED	DATE		The date and time when this encoding was originally entered (which is the default).
STATUS_DATE	DATE		The date and time on which STATUS was last changed.
NOTE	CHAR	50	A comment.
ENCODER_ID ^f	INT		It was determined that this could not be used for v0.
OVERSEER_ID ^f	INT		It was determined that this could not be used for v0.
COORDINATOR_ID	INT		It was determined that this could not be used for v0.

August 4, 2010

- Initial design and planning ✓
- System architecture ✓
- Database abstraction layer ✓
- Encoder interface and literature search interface **mostly** ✓
- Database viewer **(main building blocks available)**
- Data analysis environment **partly** ✓
- Review interface
- Other system tasks
 - Refactor existing auxiliary programs ✓
 - Status monitoring
 - System monitoring **partly** ✓
 - Verifier interface
 - Editor interface
 - Ordering system **partly** ✓
 - Institution data entry
- Final acceptance test

- Initial design and planning ✓
- System architecture ✓
- Database abstraction layer ✓
- Encoder interface and literature search interface **mostly** ✓
- Databases
- Data analysis
- Review interface
- Other systems
 - Refactoring
 - Status
 - System
 - Verification
 - Editor
 - Ordering
 - Institution data entry
- Final acceptance test

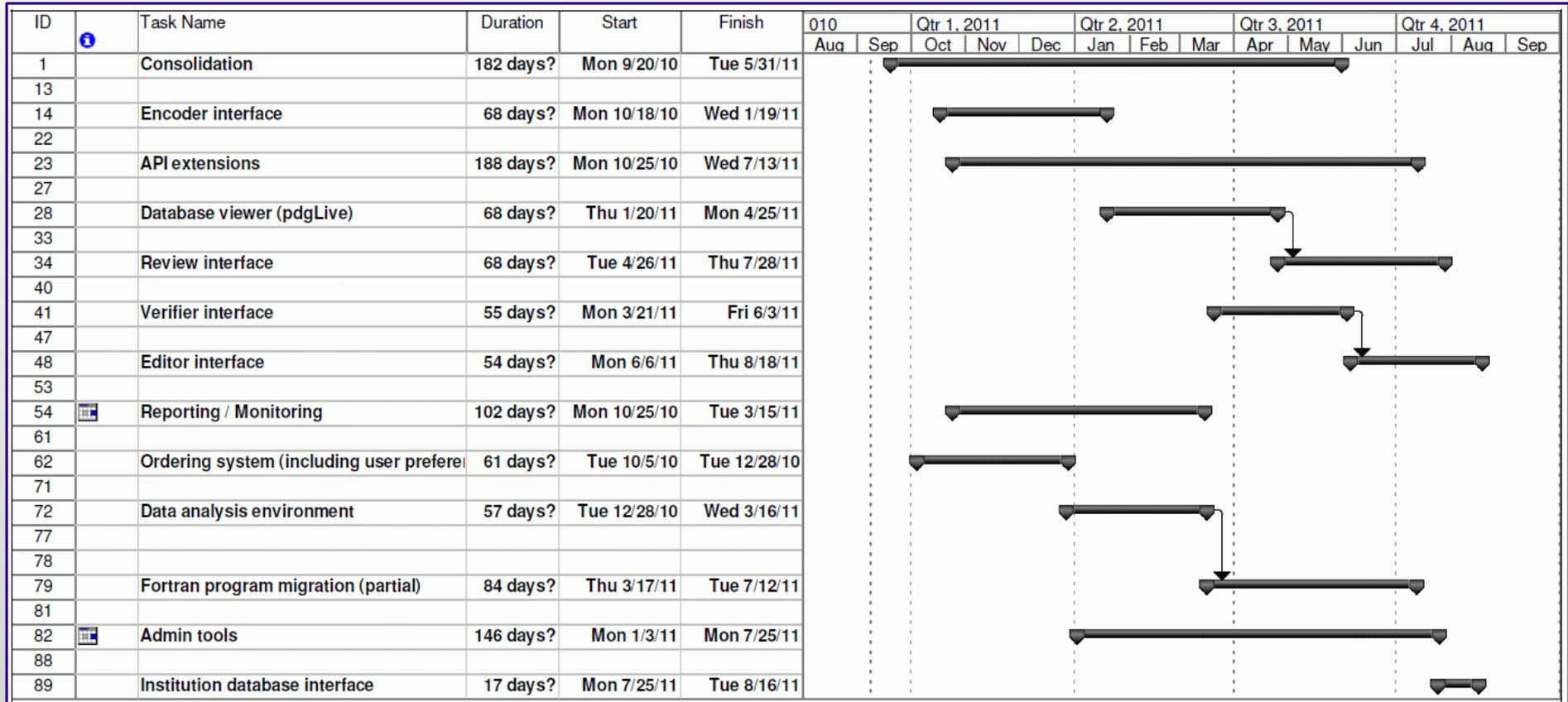
• All difficult parts posing potential risk to the project are implemented

• The encoder interface is by far the most complex and difficult application to implement

• The encoder interface includes the building blocks needed for the other applications (e.g. macro processing, math display, etc)

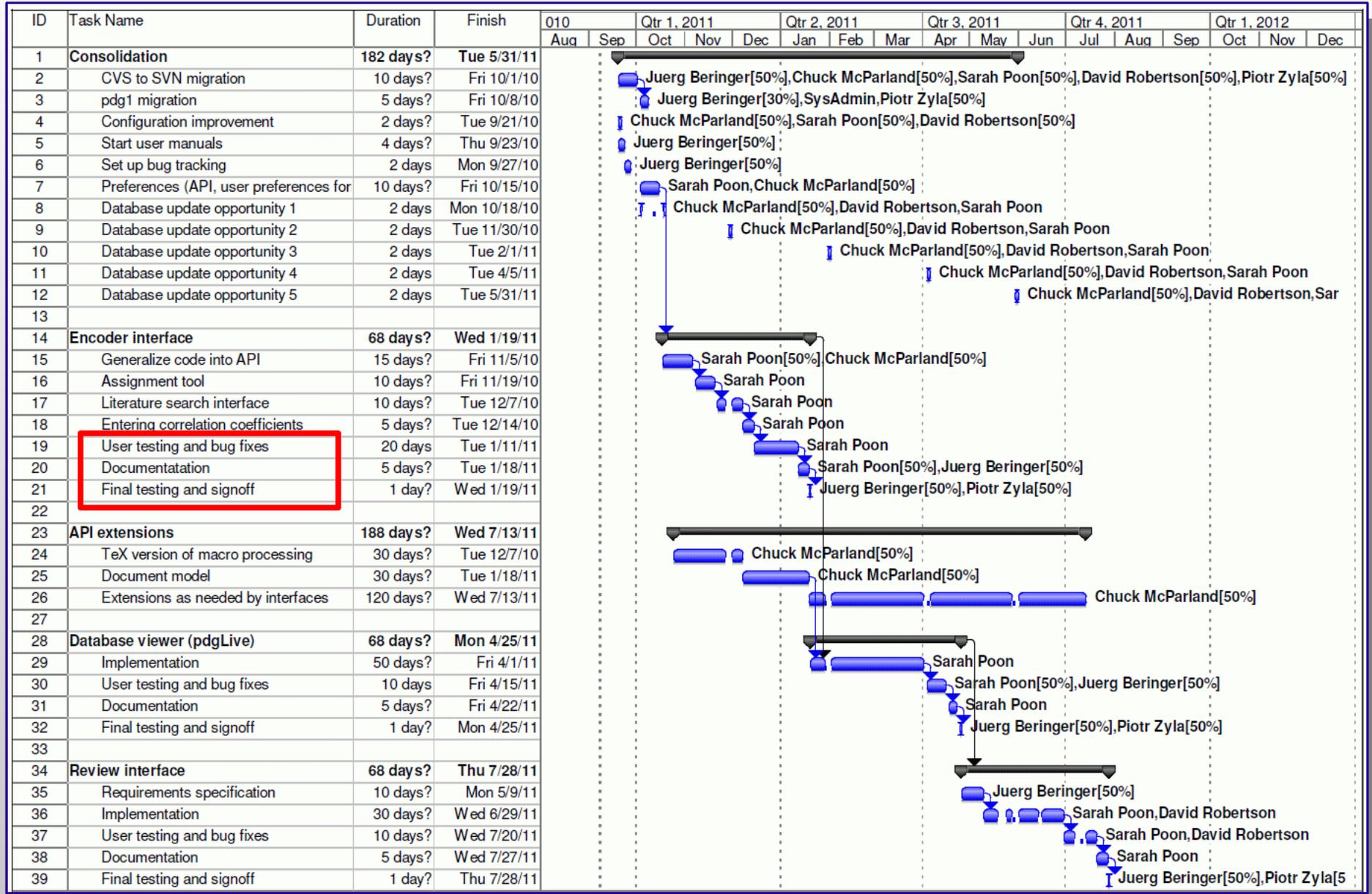
• Therefore, building the remaining applications will be relatively fast

Future Plan - Summary

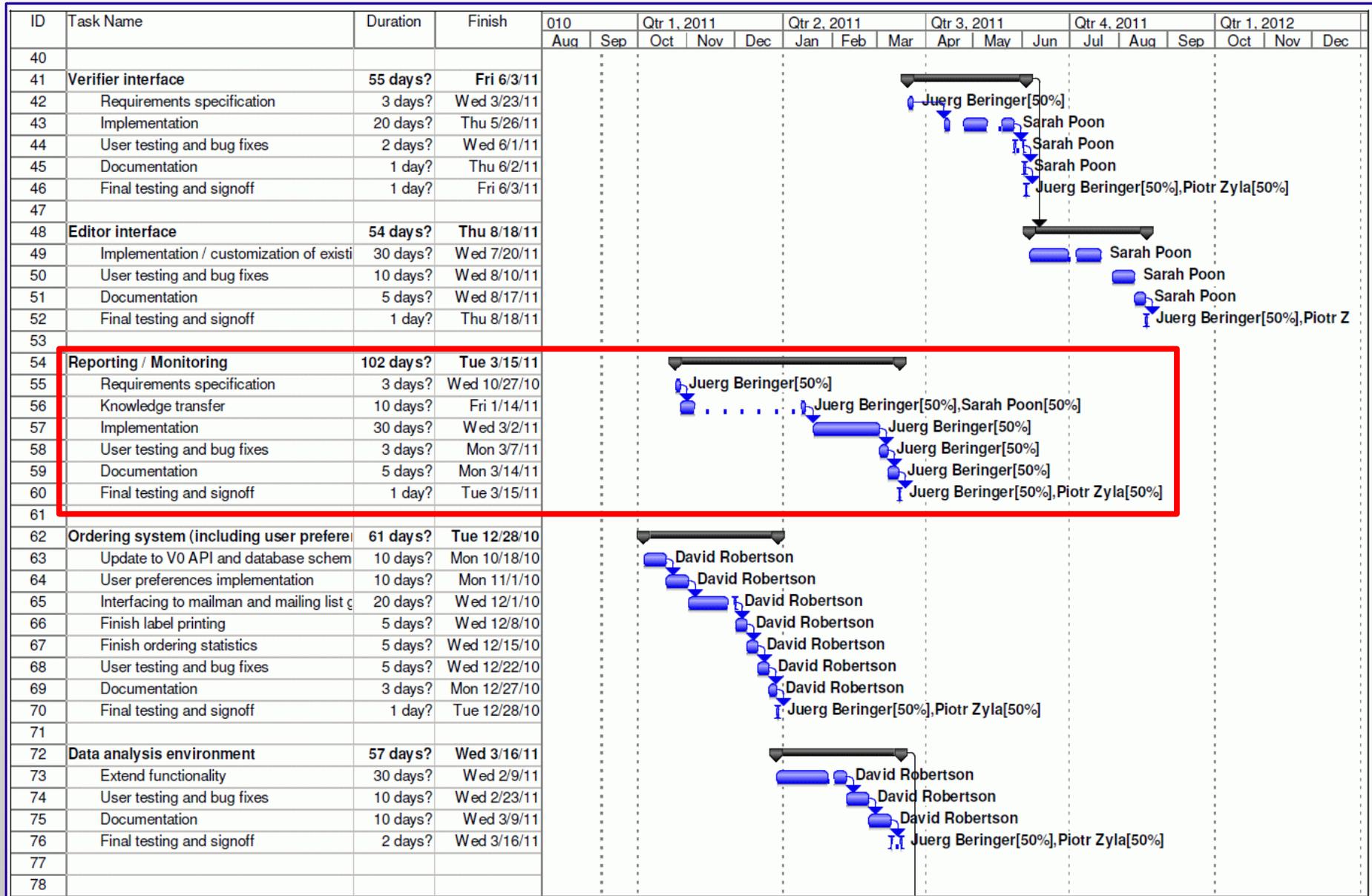


Project completion expected mid August 2011
• Leaves 1.5 months of contingency until end of FY11

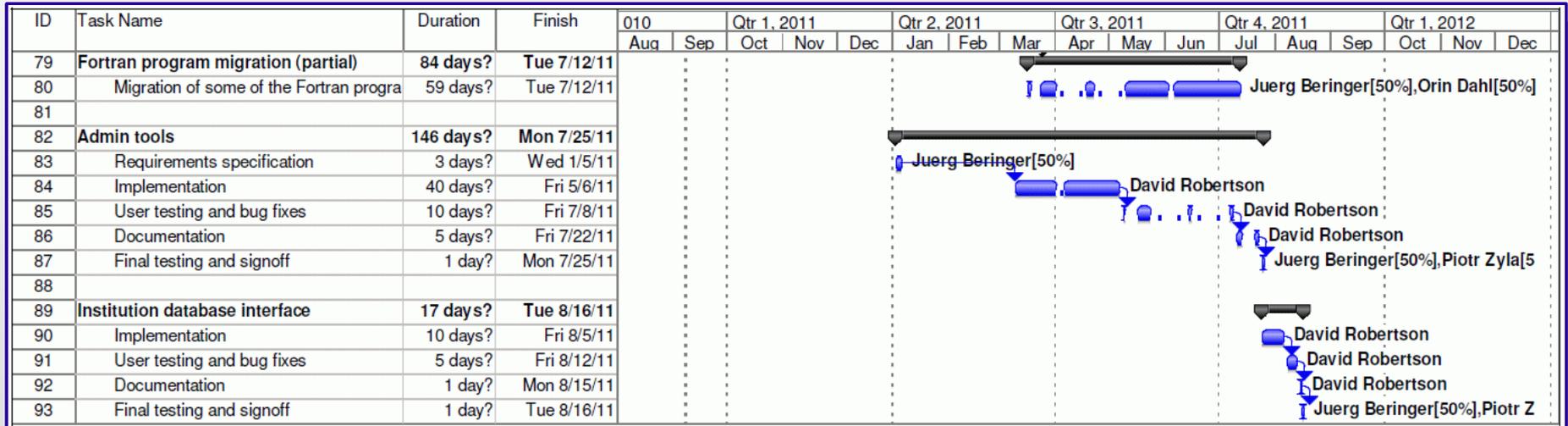
Future Plan - Details



Future Plan - Details



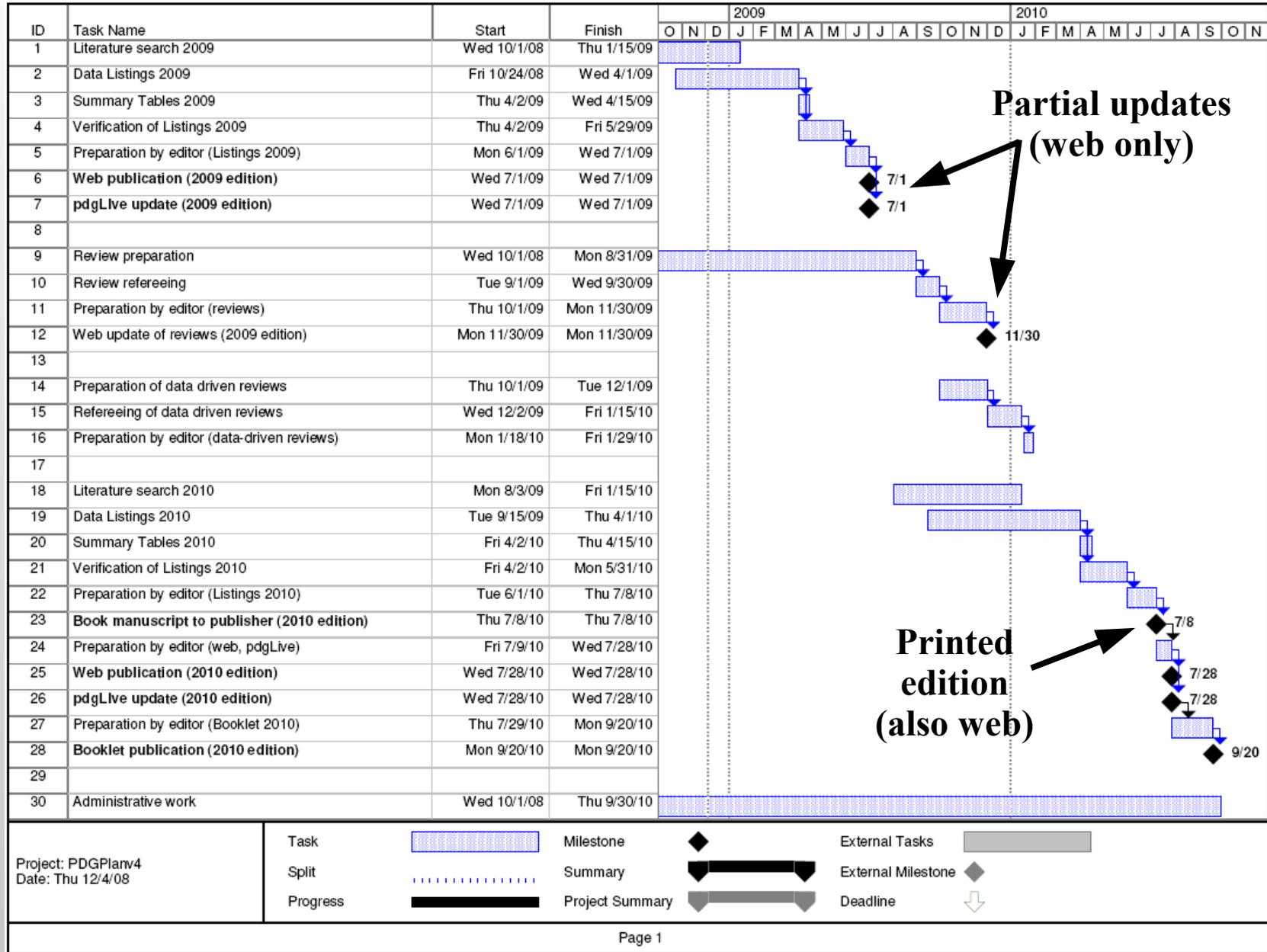
Future Plan - Details



- **Immediate and primary goal of the PDG computing upgrade is to ensure PDG can continue to function well**
 - This has absolute priority over any fancy extensions
- **New computing system is also providing platform where innovative new features can be implemented**
- **Several activities started in this context**
 - Collaboration with **INSPIRE** on cross-linking using **PDG Identifiers**
 - Participation in **HEP Information Resource Summits**
 - Accepted **oral presentation at CHEP'2010**
 - Will be an important forum to get user input
 - Brain-storming about new features (pdgLive on smart phones, opening PDG platform to support averaging groups, user tagging, programmatic user access to PDG database, ...)

- **With the V0 Release the backbone of the upgraded PDG computing system has been completed and successfully deployed into PDG production**
 - The primary challenges of the project have all been successfully addressed
 - First release of a modern, **extendable** and **maintainable** PDG system
- **All technologies for the remaining parts of the system are already working in the V0 system, and the main building blocks needed for the remaining applications are available and working in the encoder interface**
- **The remainder of the project will be primarily devoted to the implementation of the remaining user interfaces**
- **We foresee a successful completion of the project on time and on budget around mid-August 2011**
 - 1.5 months of contingency until end of FY11

Backup Slides



To give an approximate measure of the size of the source code developed, here are some numbers of lines of source code:

- **Java API** **75k**
 - Related to database (of which 38k generated) 44k
 - Related to macro processing 22k
 - Related to unit tests 9k

- **Encoder interface** **16k**
 - Java 8k
 - CSS 2k
 - HTML, JSP, JavaScript 6k

- **Python API** **1k**

- **Migration scripts (SQL, some Python)** **3k**

- **Legacy Fortran programs (incl. 45K comment lines)** **110k**