Development Project Management

Jim Kowalkowski

Outline

- Planning and managing software development
  - Definitions
  - Organizing schedule and work (overall structure)
  - Relationship with requirements
  - Tools experience
- Using redmine
  - Is it the right choice? (strengths and weaknesses)
  - Working with redmine for development
  - Using attributes

Definitions

- Milestone – “Milestones are tools used in project management to mark specific points along a project timeline. These points may signal anchors such as a project start and end date, a need for external review or input and budget checks, among others. In many instances, milestones do not impact project duration.”
- Release – A named set of features / functionality with a deadline. Deadline can be periodic, based on an external event, or total labor estimate for a grouping of needed features.
- Stakeholder – An individual, team, or organization having an interest in the completed system meeting there requirements.

Overall WBS structure

- What is the project?
- Who are the critical stakeholders?
  - How will they judge that it is done?
  - What do they want out of the project?
- Define project phases
  - Each can deliver more complete functionality (than previous)
  - Milestones, significant events or parts of the project
  - Can be time dependent or follow one from another
- Define major deliverables (set of features)
- Divide deliverables into the phases
- Define the tasks to complete each deliverable
- Determine the relationships between the tasks
- Determine the time required for each task
  - Coarse guess for our purposes here

How have we applied this?

- Identify the major components or subsystems
  - the architecture of the system
- Phases become the milestones (significant events)
  - Identify what will be available or complete across all components (effectively stating what is required at this time)
  - Associate each with a named release
- Deliverables (set of features) are defined within phase and component
- Tasks define what needs to be done to complete a deliverable
- A deliverable is assigned an owner to coordinate the development
- Does not work like building a car or constructing a building
  - Each release completes some functionality of each component

Relationship with Requirements

- Answers the question: Are we working on the right stuff?
- Requirements from key stakeholders are most important
- In this context, tie directly to milestones
- The milestone should list the key requirements that will be met
- Avoid diversions (things without direct link to a milestone)

Tools experience

- Difficult finding the right tool
- Problem: Three different areas are involved
  - Planning
  - Development
  - System architecture and requirements
- Planning tools (MSProject) only cover one area.
  - Not geared for collaborative use at the level we can need them.
- Managing development can benefit from some of the agile development practices.
  - These practices can collide with standard PM methods and tools.
- Enterprise tools, such as IBM Jazz/RTC and TeamCenter cover everything, but the learning curve, cost, and development constraints are very high.

What about redmine?

- Strengths: it works, known by many users, useful for issue tracking, connected with repositories, can tie issues to releases.
- Weaknesses: planning tools tied to issue tracking, scheduling tools (Gantt chart) based simply on issue start / end dates and fraction complete. Limited relationship support.
- Conclusion: Mediocre solution for project planning and tracking. Useable for development management. It is a practical solution given the circumstances. Best thing that we have access to.

Using redmine: observations

- Roadmap is important: It summarizes releases.
  - Use symbolic names for releases.
  - Use these as milestones.
  - Release numbers are not generated until the work unit is complete and ready for tag and build.
- The issue tracker hierarchy (projects and subprojects) is useful for separating the work.
  - Developers and managers must directly use the issue tracker.
- Assign deliverables within components to development groups / developers.
- Weekly development group meetings for moving tasks through the development workflow or updating the completion status are important. Also split or move tasks to different releases (reprioritization process) at these meetings.
- Use custom queries during meeting to see active and resolved issues.
- Communication between redmine subproject needs to be handled by the developers of that component.
- When development is underway, tasks where time is reported directly with estimates over a couple of weeks are difficult to believe

Use of redmine attributes in art

- Target version:
  - Used if the issue is accepted
- Category:
  - Describe what type of work the issue pertains to
  - Examples: I/O, Event loop, Infrastructure, Integration and review, Metadata
- Component or Package:
  - Well-defined areas that the request applies to
  - Examples: infrastructure, Fhicl, Art, cetlib
- Status:
  - state in the issue workflow
  - New, assigned, resolved, feedback, rejected, accepted, closed

Attributes continued

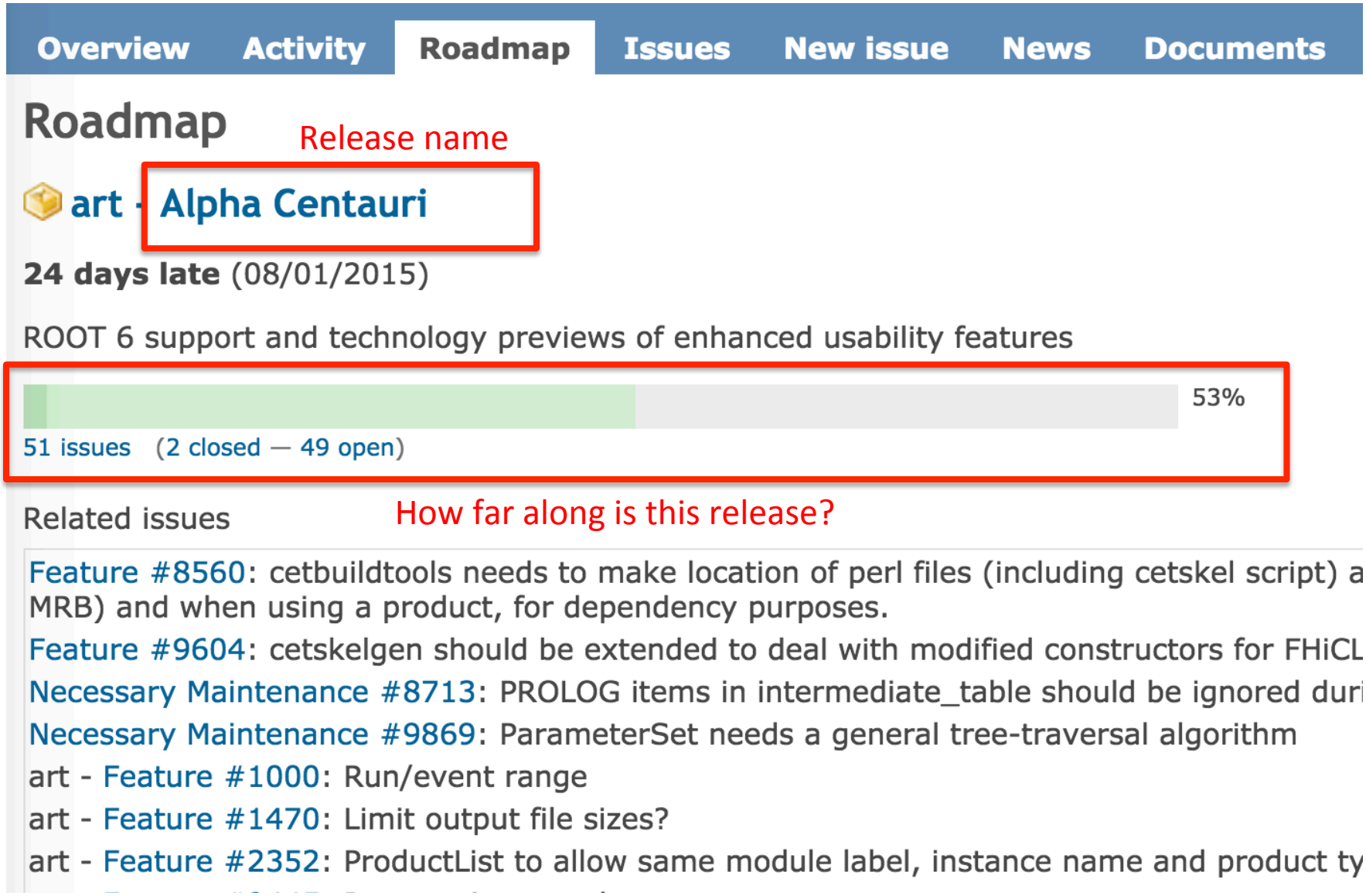
- Start date:
  - Date when we would start this work if we had infinite manpower
  - The ideal start date.
  - Accepted status means that a start date should be filled in.
- Due date:
  - Stakeholder provides for feature, otherwise filled in when it is accepted.
  - Aligned with the release that it will be tied to.
- Estimated time:
  - The time we expect to spend on the issue.
  - Put in when the issue is accepted.
- Spent time:
  - record of how much time was actually spent on this.
  - Used incrementally. This is used to help improve estimates.

A few questions

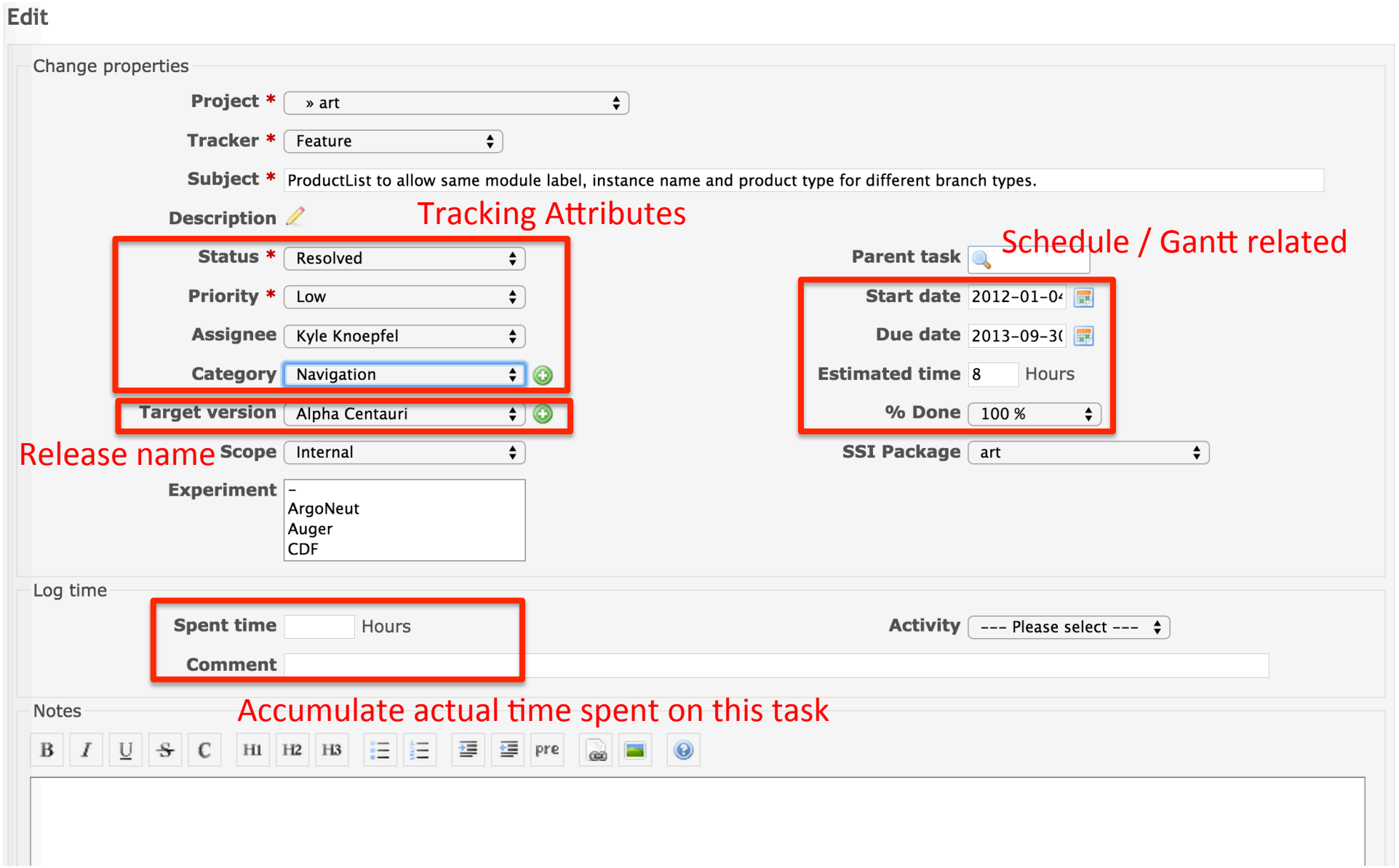
- Do you want to define a WBS?
- What is the overall strategy for tying tasks that identify deliverables to deadlines?
- How will development be managed?
- How will reporting and status be handled as things progress?
- How many people will be carrying out the development tasks under the task that identify deliverables?

Backup slides

Example Roadmap screen



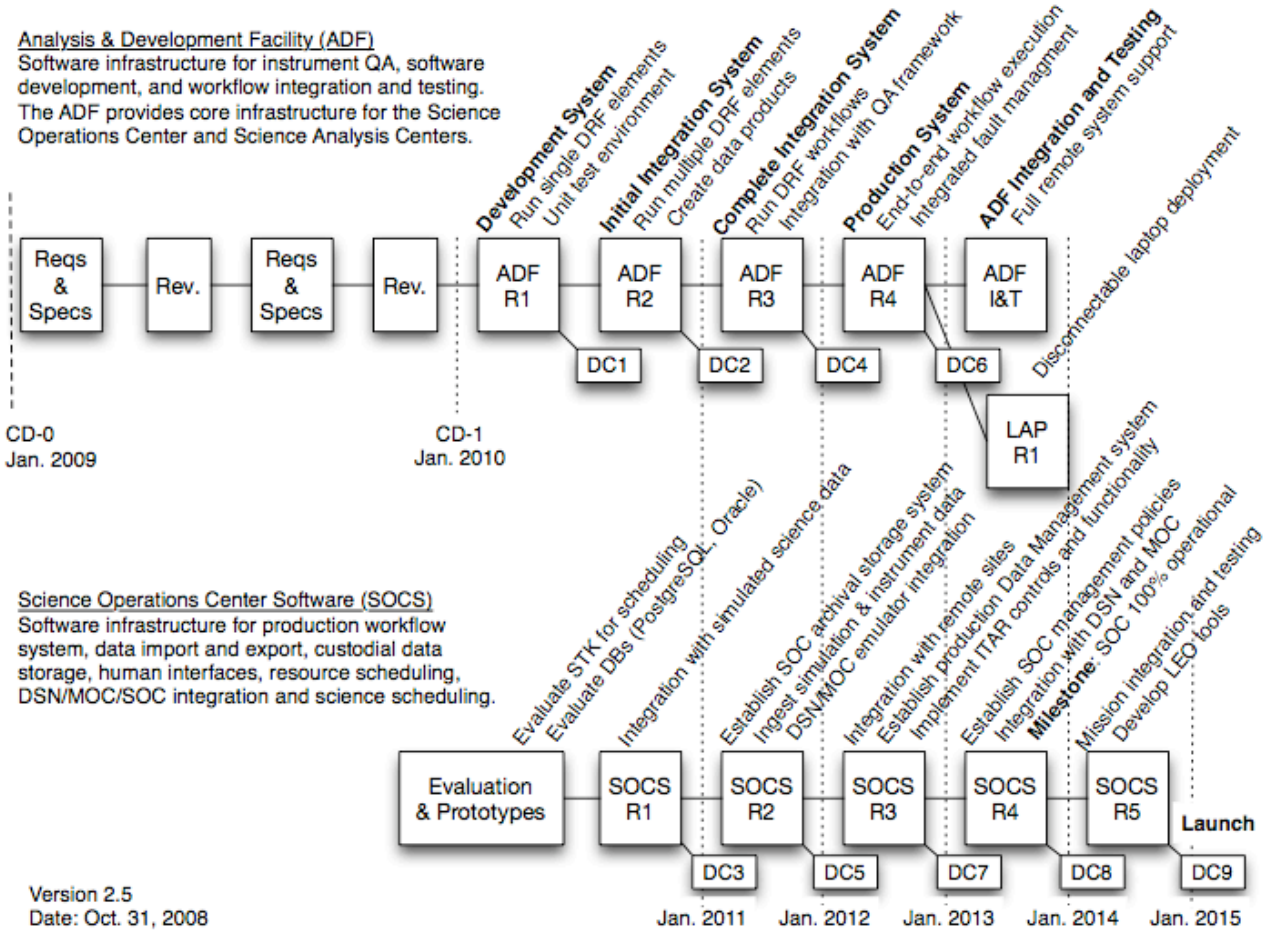
Art redmine issue example



Major Phases (From Stan)

- Information gathering
- Conceptual Design Phase
- Requirements
- Sub-System Development Infrastructure
- Evaluation and Simulation
- Development
- Data Challenge(s)
- Hardware specifications and acquisition
- Deployment
- Maintenance
- Close

Example release schedule from JDEM



Example Breakdown – JDEM SOC

Sheet1

	January, 2009		January, 2010		January, 2011	January, 2012
	CD-0		CD-1		CD-2/3a	
		Reqs &Spec	**Code Repository**	**ADF R1/SOC R0 (~6 mo)**	**ADF R2/SOC R1 (~6 mo)**	**ADF R3/SOC R2 (~1 yr)**
DEV	**Development**	Define Code standards	Evaluate CVS/SVN/etc; checkout/commit packages; validation of coding standards	Build/configure private release; unit test data created	Build/configure release	
CODE	**Code**			Simplified test environment	[good test environment]	Runtime environment; run integration tests; performance report generator; define workflow; interact with DB
CONF	**Configuration**			Exercise API for unit development [text file mode]	Functionality report generator; [define chain of elements]	Human interface to CONF
DRF	**Data Reduction Framework**			Run single element [logging API operational]	Run multiple elements	Checkpointing; integrate with QAF
DRM	**Data Reduction Module**	Define API		Code module using API		
DR	**Data Reduction System**	Define Data Dictionary		Define data product types	Create data products	Specify production workflow (one or more pipelines); workflow provenance tracking [use CONF to do DR]; save data products
DRFS	**Data Reduction Framework Sensors**					Write DRFS modules
QAF	**Quality Analysis Framework**				Run single element	Map status codes to actions; run multiple elements; supports publish/subscribe
QAM	**Quality Analysis Modules**	Define API		API library implementation; define QAF products	Code module using API; create QAF products; global logging system	Write resource monitoring modules
(D/M)QM	**Data/Mission Quality Monitor**				DQM modules R1	DQM modules R2; web based DQM & MQM

Structure example – JDEM SOC

		January, 2009		January, 2010	January, 2011		January, 2012		January, 2013	January, 2014	January, 2015 launch
		CD-0		CD-1	CD-2/3a						
			Reqs &Spec	**Code Repository**	**ADF R1/SOC R0 (~6 mo)**	**ADF R2/SOC R1 (~6 mo)**	**ADF R3/SOC R2 (~1 yr)**	**ADF R4/SOC R3 (~1 yr)**	**ADF I&T**	**SOC R4 "100%" (~1 yr)**	**SOC R5 "50%" +JDEm I&T**
	DEV	**Development**	Define Code standards	Evaluate CVS/SVN/etc; checkout/commit packages; validation of coding standards	Build/configure private release; unit test data created	Build/configure release		Release management/distribution			
WRITING	**CODE**	**Code**			Simplified test environment	[good test environment]	Runtime environment; run integration tests; performance report generator; define workflow; interact with DB	Debugging environment			
UI	**CONF**	**Configuration**			Exercise API for unit development [text file mode]	Functionality report generator; [define chain of elements]	Human interface to CONF			Workflow template management; assign pipeline priorities; reprocessing request mechanism; establish priority policy	
	DRF	**Data Reduction Framework**			Run single element [logging API operational]	Run multiple elements	Checkpointing; integrate with QAF	full workflow scheduling/multiple; workflow coordination; workflow restart capabilities		Pipeline load balancing; pipeline optimization	
	DRM	**Data Reduction Module**	Define API		Code module using API			Science product provenance in DM; science product provenance in header			
CORE: IMAGE DATA STREAM	**DR**	**Data Reduction System**	Define Data Dictionary		Define data product types	Create data products	Specify production workflow (one or more pipelines); workflow provenance tracking [use CONF to do DR]; save data products	DR: apply fault management policy (ADF)			
	DRFS	**Data Reduction Framework Sensors**					Write DRFS modules				
	QAF	**Quality Analysis Framework**				Run single element	Map status codes to actions; run multiple elements; supports publish/subscribe	Historical tracking of workflow progress (DB); automated error handling; [event-driven system with scheduling]; alarm handling			
CORE: EXTERNAL DATA	**QAM**	**Quality Analysis Modules**	Define API		API library implementation; define QAF products	Code module using API; create QAF products; global logging system	Write resource monitoring modules				
UI	**(D/M)QM**	**Data/Mission Quality Monitor**				DQM modules R1	DQM modules R2; web based DQM & MQM	DQM modules R3; MQM modules R1		MQM: physics validation; MQM modules R2; MQM: human interaction	MQM modules R3
	PS	**Production System**					PS: schedule DF for newly arrived data; integrate with GRS	PS: applies fault management policy (SOC)		PS: human interaction with job queue; applies Production Data Management Policy	
ONLINE	**ST**	**Supernova Trigger**				ST: define ORL API	ST: human interface for ORL	ST: human interface for selection/prioritization; generate prioritized ORL			
	DM/L	**Data Manager/Local**			Initial DB API/Schema [storage capability: local FS]	Local ID assignment; [configuration and provenance API/schema]; [storage of user-defined products]	Storage of configuration/provenance using API	DM/L: Integrate with proxy system			
DM	**DM/C**	**Data Manager/Central**				Global ID assignment	Instantiate cataloging system; instantiate archival storage (heirarchical dCache/Enstore)	DM/C: instantiate production DM			
	PIT	**Pipeline Integration Tester**					PIT: definition of environment and tests; integrate with mock (external) components	PIT: recording of test outcome	PIT: integrate with release system to automate running	PIT: human interacts with results	
	CC	**Calibration Creation**					CC: generate calibration observation requests	CC: generate optimized schedule		CC: human interface to select workflows; human interface to modify and execute calibration workflows	
	OS	**Observation Scheduler**			Evaluate STK	Install/configure STK	Generate ORL schedule	Generate optimized ORL schedule; generate telescope commands		Receive instrument availability; send telescope commands to instrument; send observation schedule to instrument; sanity check schedule	Human interface for science scheduling
ONLINE	**(D/M)IM**	**DSN/MOC Interface Manager**				Simulated DSN data	"Real" DSN data written to DM/C; Mock MOC integration; verify validity of dadta (gaps)	ITAR ICD implemented; implement schedule integrity		Real MOC integraton; MIM: verify receipt of data; MIM: QoS features available	
	PROXY/S	**Proxy Server**					Instantiate proxy system	Proxy: caching capability		Proxy: apply DM/C write restrictions;	
DM	**PROXY/C**	**Proxy Client**					Direct access to DM/C; integrate with DM/L				
	IM	**Ingest Manager**				Define ingest API	Ingest instrument/simulation data; ingest external catalogs	Automated ingest of data		Ingest: apply DM/C access restrictions	
	DE	**Data Exporter**				Define export API	Web interface to export data	SQL interface to export data		Export: apply DM/C access restrictions; Laptop: populate cache	
EXTERNAL	**PAI**	**Public Archive Interface**					Define PAI API with "STScI"	Initial implementation of PAI		Instantiate PAI; transfer data to PAI	Data release capability and distribution validation
	OD	**Operations Display**					Delivery of QA data to clients; web based access to QA data	OD: display QAF and DRF status; computing cluster display; alarm display; flexible visualization of data;		Human interface for fault management policy	
UI	**SOCM**	**SOC Manager**					Electronic logbook; restricted virtual operations display	Automated logbook entries	RBAC; open virtual operations display		
	GRS	**Global Resource Manager**	Evaluate batch systems			Batch system	Manage console hardware and software		Respect pipeline priorities		
		Database	Evaluate DB (scalability)	Development DB (Postgres)	Instaniate Integration DB		SOC instantiated	Instantiate Production DB (scalability tests)			
		Physical Plant	Prototype system	Prototype system	Development system at FNAL	Integration System at FNAL	Add to Development & Integration systems	Production system (scalability tests)	Ability to reproduce an integration system elsewhere		