



LHCb Production System

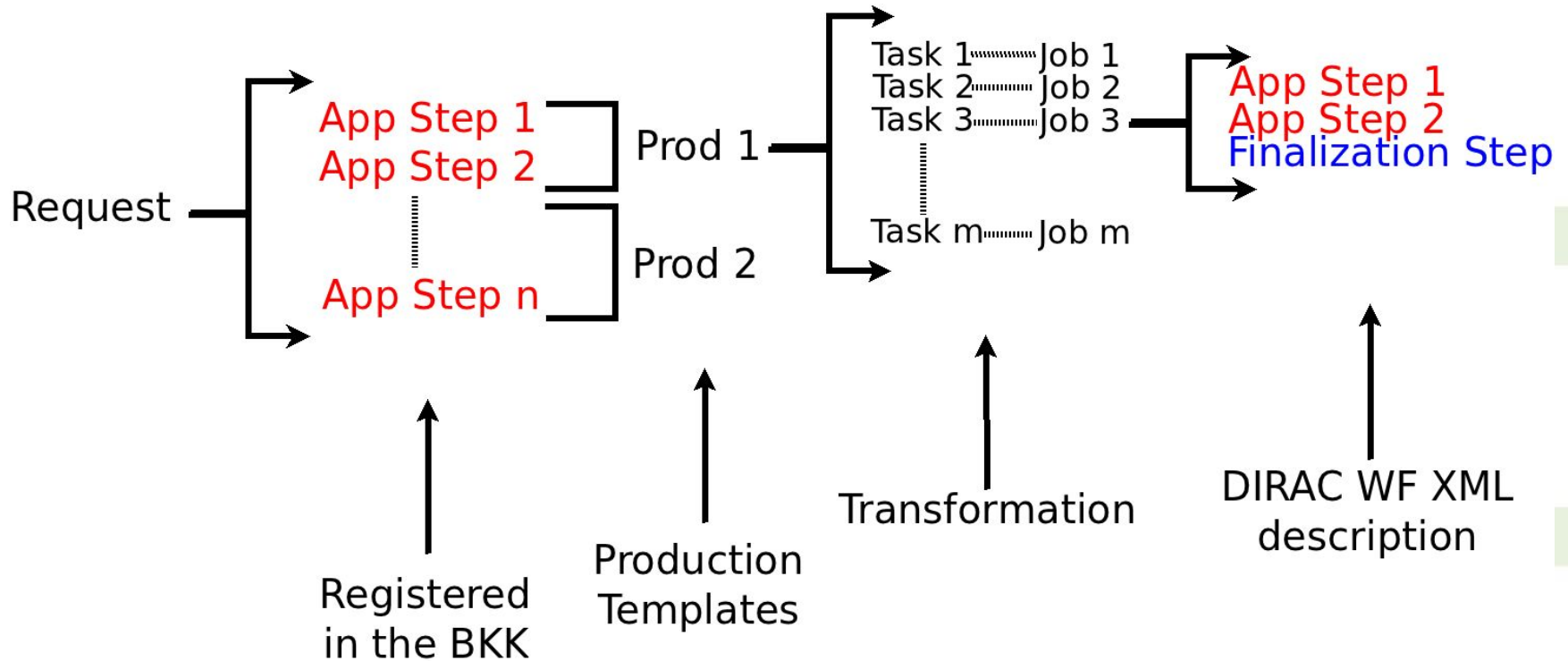
Federico Stagni, on behalf of the LHCbDirac team



**I prepared these slides in 2012!
for the 3rd DUW**



Putting concepts together



[DIRAC user meeting 2011](#) (click!)

- ▶ Xml ↔ python dict
- ▶ A workflow connects steps together
- ▶ `dirac-jobexec aWorkflow.xml`

Jdl:

- ▶ `Executable = "$DIRACROOT/scripts/dirac-jobexec";`
- ▶ `Arguments = "jobDescription.xml -o`
`LogLevel=verbose";`





Production Request System

System Jobs Production Data View Web Selected setup: LHCb-Development

Registered Steps Edit step 14378 Edit step 14278

Name: Validation-Stripping17-Stripping-CondDB20111111

Processing pass: Stripping17

Application: DaVinci v29r1

Option files: \$APPCONFIGOPTS/DaVinci/DV-Stripping17-Stripping.py

Options format:

Extra packages: AppConfig.v3r111,SQLDDDB.v6r20

Runtime project: Select Runtime Project if desired

CondDB: head-20111111

DDDB: head-20110914

DQTag:

Visible: Y

Usable: Yes

Input File Types

File type	Visible
SOST	Y

Output File Types

File type	Visible
BHADRON.DST	N
CALIBRATION.DST	N
CHARM.MDST	N
CHARMCOMPLETEEVENT.DST	N
DIMUON.DST	N

Save Cancel

production > Step manager fstagni@ lhc_b_tech (DC=chDC=cern/OU=Organic Units/OU=Users/CN=fstagni/CN=693025/CN=Federico Stagni)

- ▶ Application Managers defines application steps
- ▶ “What to run” to go from X to Y
- ▶ LHCb application
- ▶ A step “translates” in a workflow application step

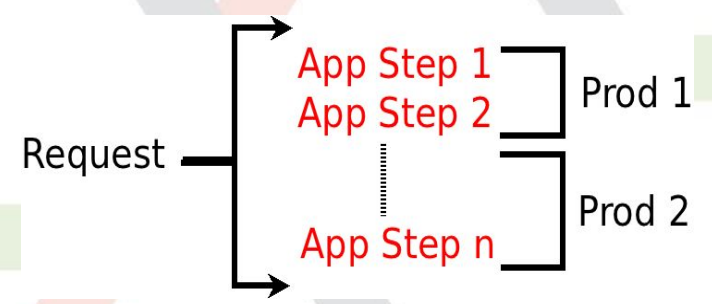


Production Request System /2

The screenshot shows the 'Production Request System' interface. The main window displays details for 'Request 22', including its name, type (Reconstruction), priority (2b), and author (fstagni). It also shows input data and processing pass information. A 'Replace Step' dialog box is open, showing a table of steps and their details.

ID	Name	Processing...	App.	Step details
12618	CaloFemtoDat...	CaloFemto...	DaVinci	DataQuality-FULL-01/12278/DataQuality-FULL-01 - DaVinci-v29r0 Options: \$APPCONFIGOPTS/DaVinci/DVMonitor-RealData.py,\$APPCONFIGOPTS/2011.py,\$APPCONFIGOPTS/DaVinci/DaVinci-InputType-SDST.py Options: DDB: head-20110914 Condition DB: head-20110914 DQTag: null Extra: AppConfig.v3r110 Runtime projects: Visible: Y Usable:Yes Input file types: RAW(Y) Output file types: BRUNELHST(Y),SDST(Y)
12658	CaloFemtoDat...	CaloFemto...	DaVinci	
13618	CaloFemtoDat...	CaloFemto...	DaVinci	
12338	CaloFemtoDat 02	CaloFemto...	DaVinci	
12178	CaloFemtoDat01	CaloFemto...	DaVinci	
12738	DataQuality-01...	DataQualit...	DaVinci	
12278	DataQuality-FU...	DataQualit...	DaVinci	
12418	DataQuality-FU...	DataQualit...	DaVinci	
12038	DataQuality-FU...	DataQualit...	DaVinci	
12518	DataQuality-FU...	DataQualit...	DaVinci	
13118	DataQuality-FU...	DataQuality	DaVinci	
13098	DataQuality-FU...	DataQuality	DaVinci	
13539	DataQuality-FU...	DataQuality	DaVinci	
13759	DataQuality-FU...	DataQuality	DaVinci	

► Steps are combined in production requests (e.g. MC, or Reconstruction)



Production Request System /3

Generate production script

Please specify Production parameters

Parameter ▲	Value
GENERAL: Set True for EXPRESS (Run at C...	False
GENERAL: Set True for certification test	False
GENERAL: Set True for local test	False
GENERAL: Set True to create validation pro...	False
GENERAL: Use Oracle	True
GENERAL: Workflow string to append to pr...	1
GENERAL: Workflow system config e.g. x8...	ANY
PROD-RECO: DataReconstruction or DataRe...	DataReconstruction
PROD-RECO: Group size or number of files ...	1
PROD-RECO: Max CPU time in secs	1000000
PROD-RECO: Number of Files	-1
PROD-RECO: Output Data Storage Element	Tier1-RDST
PROD-RECO: ancestor production if any	0
PROD-RECO: dicrete list of run numbers (do...	
PROD-RECO: distribute output data True/Fal...	False
PROD-RECO: priority	7
PROD-RECO: production plugin name	AtomicRun
PROD-RECO: run end, to set the end of the ...	0
PROD-RECO: run start, to set the start run	0

« Previous Next » Generate Preview ScriptPreview Cancel

- ▶ Production requests are submitted using production templates
- ▶ e.g.: priority, which plugin, where the outputs are stored, DIRAC CPU, etc.
- ▶ Each production is created using the Production API

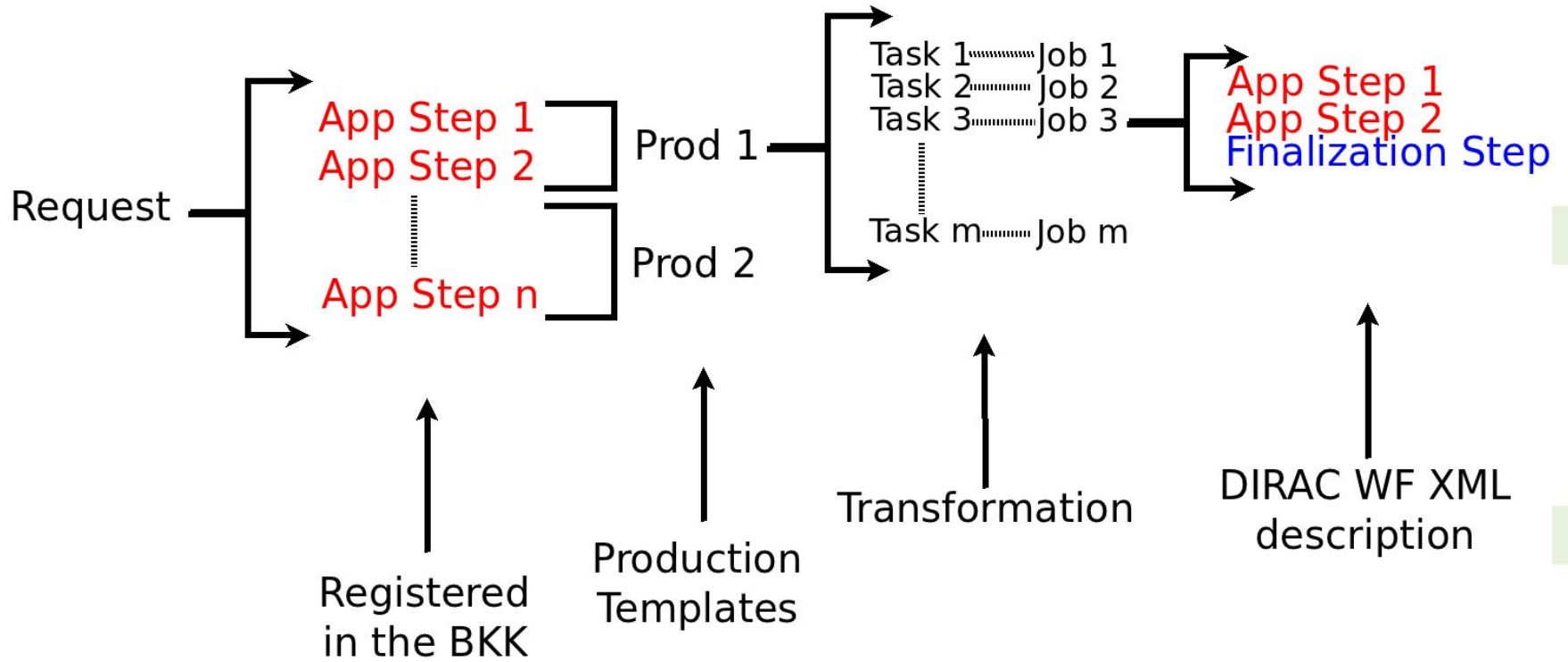
- ▶ Extension of the DIRAC TS, mostly for interacting with the BKK
 - ▶ DB:
 - ▶ Physics RUNs information
 - ▶ BKK queries (supersedes TransformationInputDataQuery)
 - ▶ Service and clients extended for the DB extension

- ▶ Agents
 - ▶ BookkeepingWatchAgent
 - ▶ Looks for BKK queries, and fills the TransformationFiles table
 - ▶ Threaded, uses pickle file for caching
 - ▶ DataRecoveryAgent
 - ▶ Resets input files in “Unused” status, in case the jobs failed
 - ▶ A counter is kept, with a maximum of re-trials
 - ▶ Extensions for cleaning, and closing productions

- ▶ Plugins
(LHCbDIRAC.TransformationSystem.Agent.TransformationPlugins)
 - ▶ Many LHCb plugins coded
 - ▶ e.g. ByRun, with flushing...
 - ▶ This is where you want to extend

- ▶ Expose functionalities to connect together TS, BKK and Production Request System
- ▶ Use LHCbJob.py (extension of DIRAC.Interfaces.API.Job.py) to create a DIRAC workflow, whose xml is uploaded to the Transformation DB
- ▶ python modules are run within the workflow, grouped within steps. Application steps and Finalization steps are present

Putting concepts together



WRT to the proposed implementation

- We have in ProductionRequestDB a table for ProductionRequest description
- The association ProductionID -> TransformationIDs is in the “TransformationFamily” field in TransformationDB/Transformations
- We have an agent for updating the transformations and the productions status, and a state machine