SVD Reconstruction Design

PROPOSAL & PLANS

Notes P. Kvasnicka, 07/07/2017

Giulia Casarosa

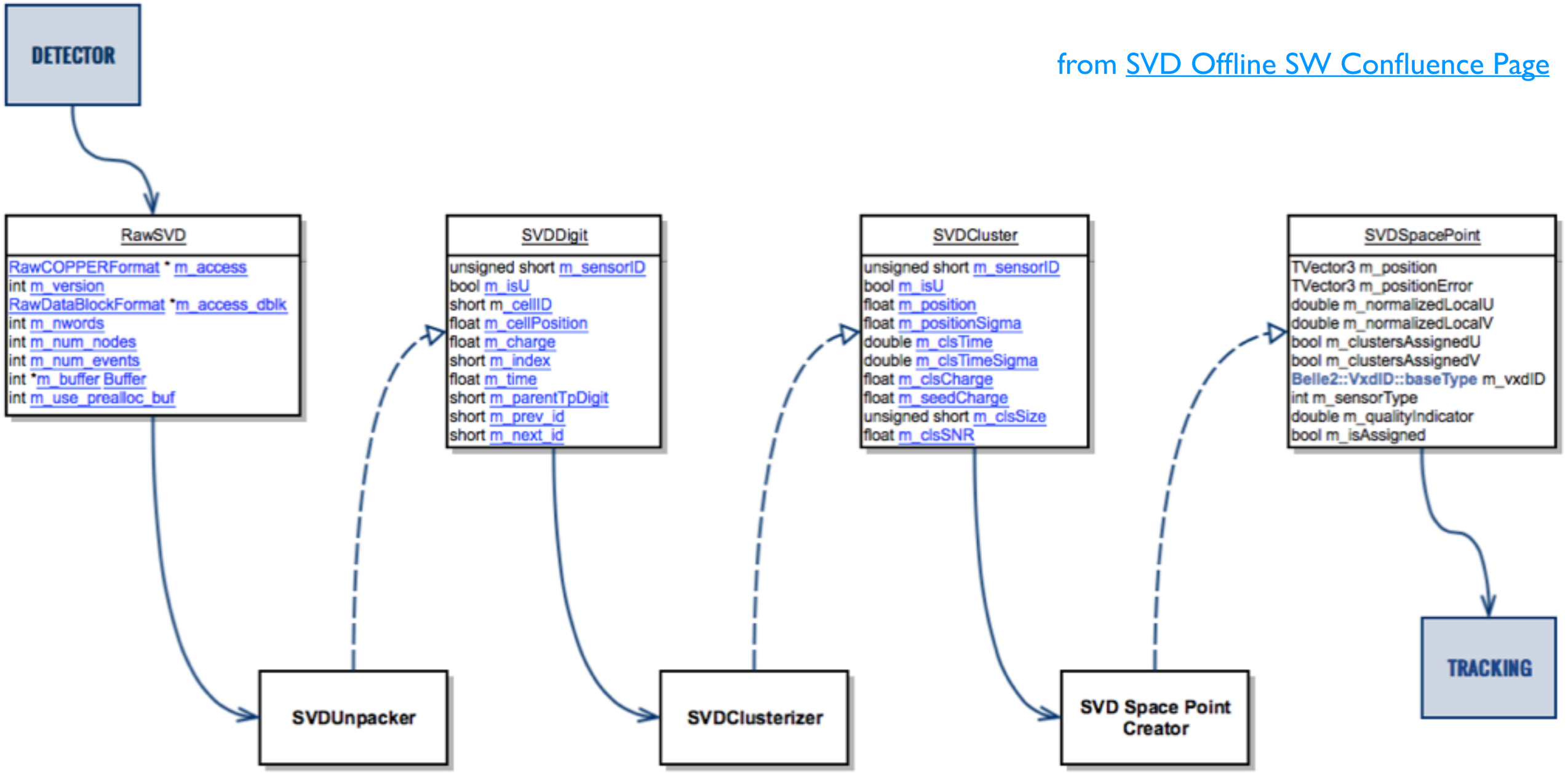
JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Alexander von Humboldt Stiftung/Foundation

Tracking Meeting ~ July, 7th 2017

Current Design of the Reconstruction

from SVD Offline SW Confluence Page



- We have implemented a basic design that has been working well up to now, it allows to reconstruct data from the test beams
- Peter's pull request includes improvements in the reconstruction, and more improvements are needed (e.g. calibration) → the design of the reconstruction needs a critical revision

Proposal - July, 5th

DETECTOR

SVDEventInfo

useful informations contained in the payload header

SVDHitTimeFitter
setTime(),
seTimePDF()

We actually don't need this to be a module - it may be a dbObject! providing a time fitting service.

short m_eventTimeBin
daqMode m_mode

Can't we just have a RelationArray<SVDCluster, SVDCluster>
for this puroose? Or a similar structure where we can ask which are possible v-partners to my u-cluster?

RawSVD

RawCOPPERFormat * m_access
int m_version
RawDataBlockFormat *m_access_dblk
int m_nwords
int m_num_nodes
int m_num_events
int *m_buffer Buffer
int m_use_prealloc_buf

SVDShaperOutput

unsigned short m_sensorID
bool m_isU
short m_cellID
rawSamples m_samples

float m_time
vector<float> m_timePDF

separate object, must not depend on DAQ mode!!!

SVDClusterP / SVDClusterN

unsigned short m_sensorID
bool m_isU
float m_position
float m_positionSigma
double m_clsTime
double m_clsTimeSigma
float m_clsCharge
float m_seedCharge
unsigned short m_clsSize
float m_clsSNR

SVDSpacePoint

TVector3 m_position
TVector3 m_positionError
double m_normalizedLocalU
double m_normalizedLocalV
bool m_clustersAssignedU
bool m_clustersAssignedV
Belle2::VxdID::baseType m_vxdID
int m_sensorType
double m_qualityIndicator
bool m_isAssigned

SVDUnpacker

SVDClusterizer

SVD Space Point Creator

TRACKING

- select fitter object based on DAQ mode (zero-sup 3 or 6 samples, transparent)

SVDCalibration

noise, gain,...
+
defects map
+
...

SVDStripCalibrator

Create functional dbObjects

SVDHitStripP / SVDHitStripN

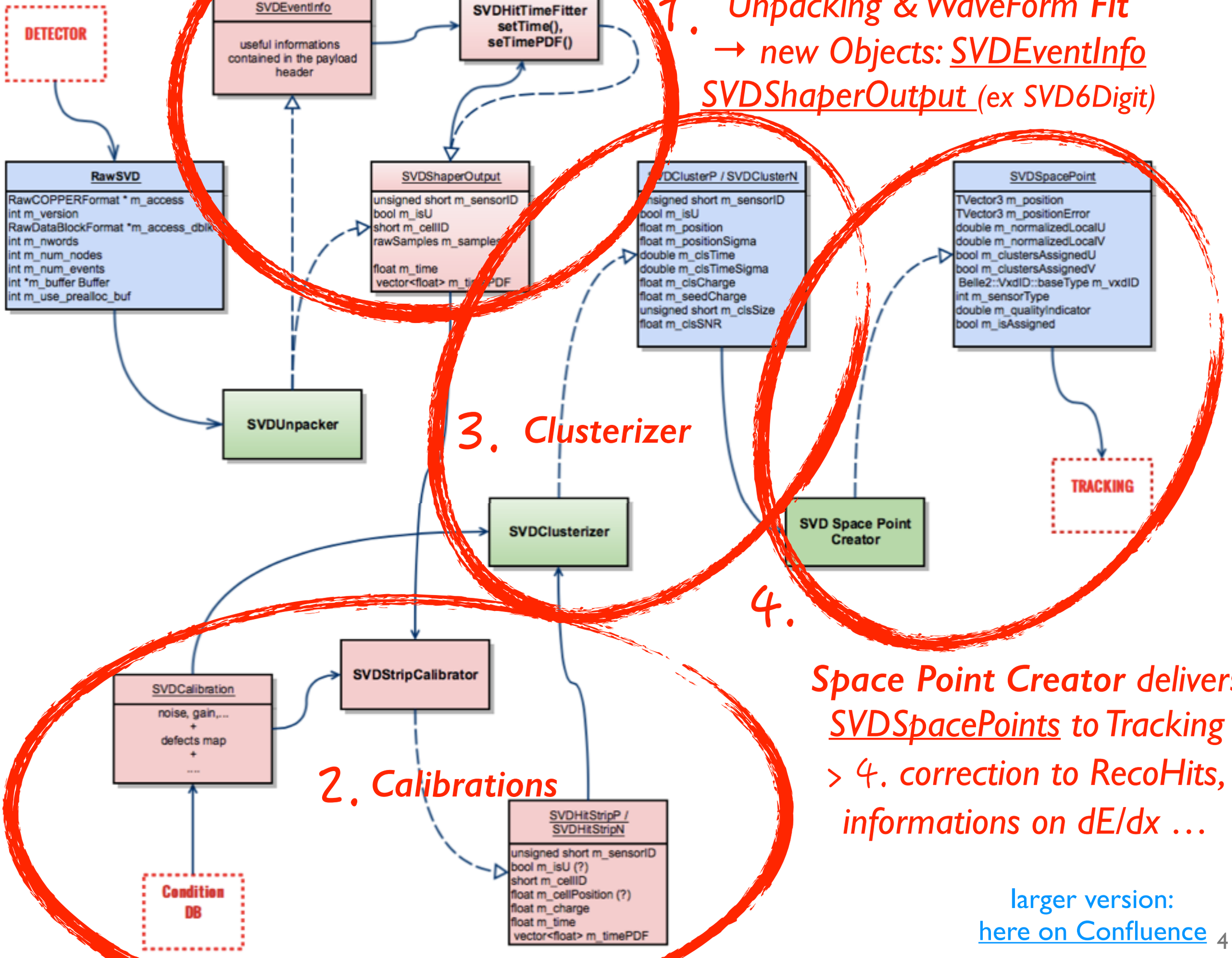
unsigned short m_sensorID
bool m_isU (?)
short m_cellID
float m_cellPosition (?)
float m_charge
float m_time
vector<float> m_timePDF

This is no-no:
such separation breeds code duplication; we don't need a separate object at all here.

Condition DB

Services can be tested more easily than modules - even outside of basf2.

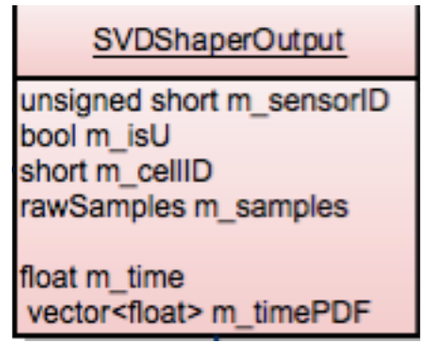
larger version:
here on Confluence



Space Point Creator delivers *SVDSpacePoints* to Tracking
> 4. correction to RecoHits, informations on dE/dx …

larger version:
here on Confluence

Comments & Concerns



- Regarding Peter's pull request we agreed on splitting it into smaller pull requests

- Regarding the proposed design:

1. agreement on the fact that having one module to perform one well-defined task improves software development (parallel work, smaller pull requests, easier debugging, …) With reservations: splitting modules is NOT the only and not the most efficient way to achieve this.
2. agreement on the fact that it allows to test different algorithms for the same tasks (e.g. time determination algorithm) Algorithms can be switched using module parameters. No need of a separate module.
3. Peter suggests to change the name of the proposed object SVDShaperOutput into SVD<something>Digit (ideas: SVDWakeDigit, SVDSampledDigit, …?)
4. Eugenio suggests to create another object containing the result of the time determination (time and PDF) and use relations to connect it to the SVD<something>Digit OK
5. Eugenio pointed out that the split of SVDCluster StoreArray into two StoreArrays, one for P and one for N clusters, would require to double the relations. Moreover Peter is not in favour of this change, therefore I would cancel this proposed change from the proposal

Comments and Concerns Cont.'d

6. Andrzej expressed the following concerns, quoted from his email:
   - "the proposal introduces objects which would have different contented in function of previously called modules (unpacker + time fitter), even if this is allowed in Belle2 (I don't think so), it is a bad practice and should be avoided,"
   - "it increase the **data size** few times (estimate depends on how many bins we would need for time PDFs),"
   - "In my opinion, unfortunately it would add additional **delays**. I remind everyone that if we want to be ready for rev 10, we would need both time fitter and calibration in main branch by August/September. The Rev. 10 would be the one used on the Beast Phase 2 "
   - his idea is that the new objects should be "treated as internal **temporary objects** created at the stage of clusterizing and should be deleted afterward and should not replace SVDDigit used in many place in software also outside of the tracking group scope"

SVDDigits are not that important or widely used, and can be modified without much pain, but it has to be done carefully. But it won't be a good idea to have an additional dataobject, or make SVDDigits bulky.

Goals and Tentative plan

The goal is to have the hit time determination (and the calibration) in release-01-00-00

month \ week	I	II	III	IV	V
July		first pull request	Unpacker 6-samples	validation…	
			SVDxxxDigit Digitizer	validation…	
			SVD Space Point Creator	validation…	
Aug	validation…	Unpacker, 3-samples	validation…	filtered clusters	validation…
	validation…	validation…	hit time determin. outside cluserizer	validation…	validation…
	calib. interface implemented	how do we apply the calibration? first we have to converge on the design			
Sept	simulation VS testbeam data	simulation VS testbeam data	simulation VS testbeam data	calibration of the simulation	
	validation…	impact on tracking	impact on tracking		

First Pull Request

➡ I propose the following:

- Rename of the SVD6Digit into SVDxxxDigit, keep Peter's design of this object except for the addition of the trigger timing information
- Include the Splitter (and Merger?) module to convert: SVDDigit ↔ SVDxxxDigit
- Unpacker and Digitizer should produce SVD<something>Digit *in addition* to the SVDDigit
- Unpacker should add the information of the trigger timing into the SVD<something>Digit; this information is stored in the header of the payload

  No. SVDEventInfo is the logical place to store information about event time.

➡ This will allow to:

- Validate the SVDxxxDigits against the SVDDigits
- Keep exactly the current steering files with no modifications

  I don't see how this is true. In dozens of basf2 steering scripts, we will have to insert a converter module between the digitizer and clusterizer.
- Immediately start the validation of the simulation of the time structure of the APV response against TB data. Since the NN is trained on simulation, this is critical.

Why Redesigning the Code

each module performs one single and well defined task

- ➡ true for SVD reconstruction in release-00-09-00
- ➡ ok, but why?

1. it's good practice
2. maintainability of the code in the future is easier (remember VXDTF…)
3. debugging of the code is easier
4. smaller and easier-to-review pull requests
5. the split of tasks in different modules, allows parallel developments (but the interfaces must be decided and agree *before* starting to write code!!)
6. documentation is easier to write and to read
7. implementation of alternative reconstruction methods in the future will be a non-traumatic (for the code) and much easier to manage and configure