

SVD RECONSTRUCTION DESIGN

PROPOSAL & PLANS

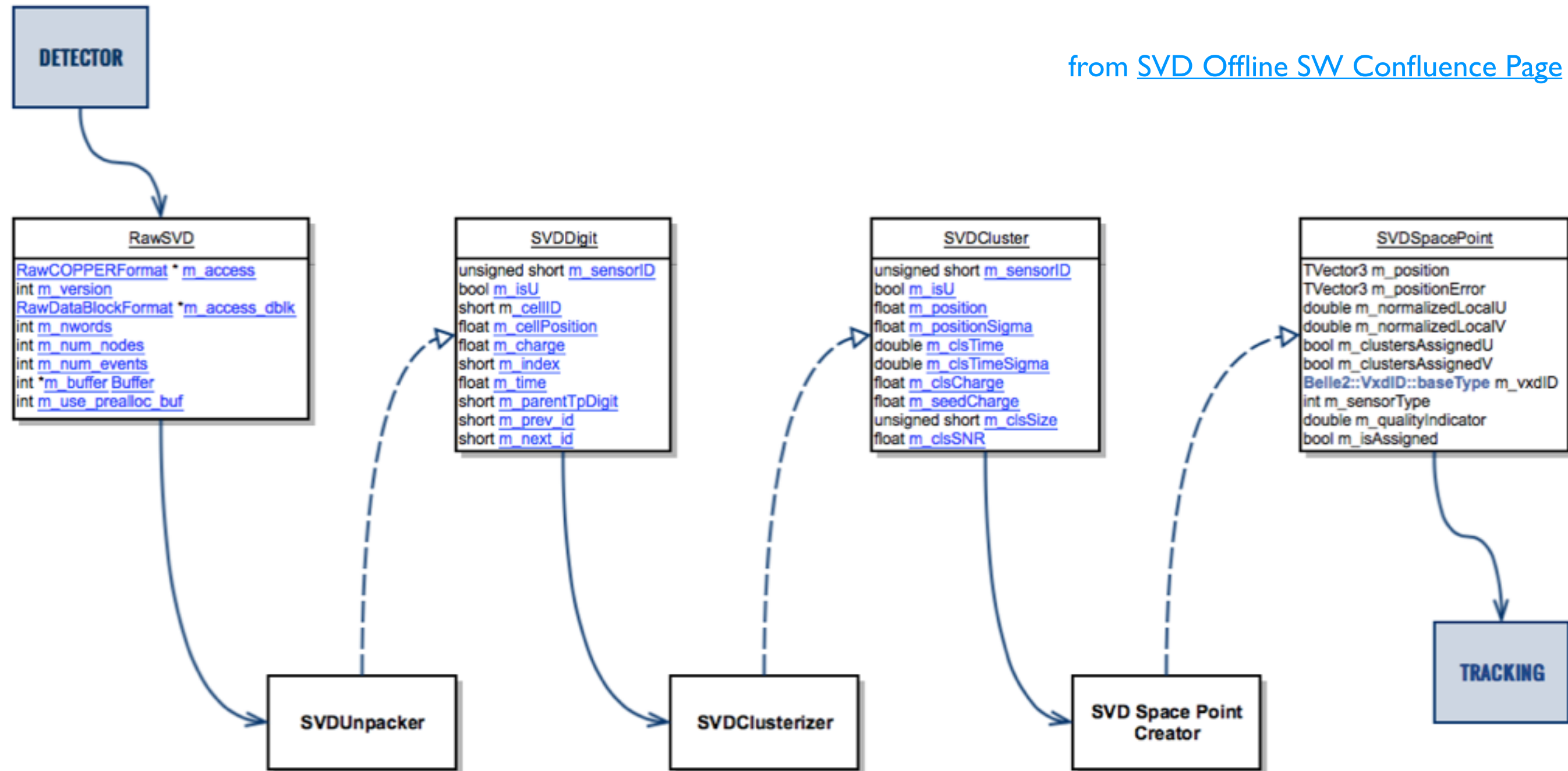
Giulia Casarosa



Tracking Meeting ~ July, 7th 2017

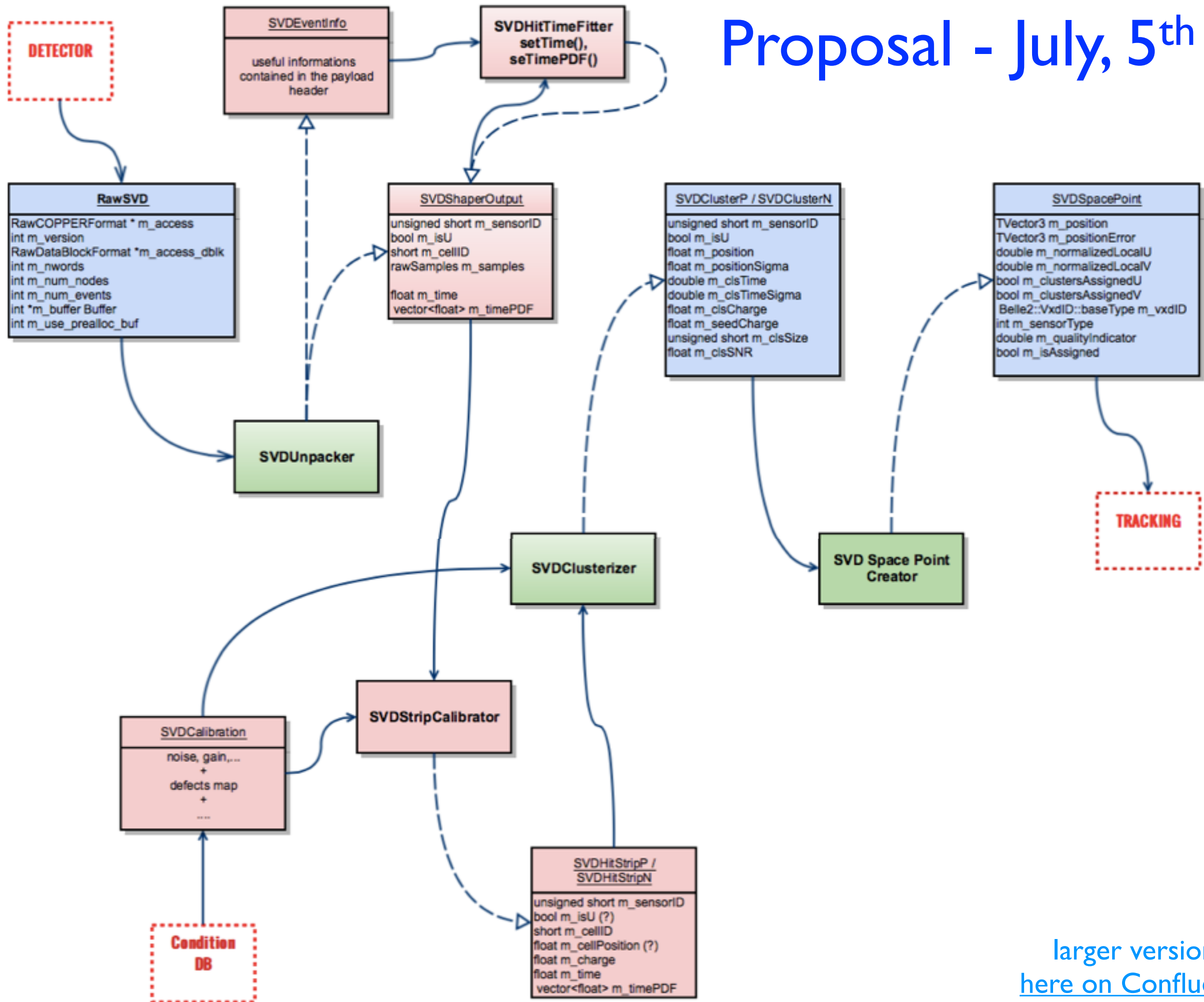
Current Design of the Reconstruction

from [SVD Offline SW Confluence Page](#)

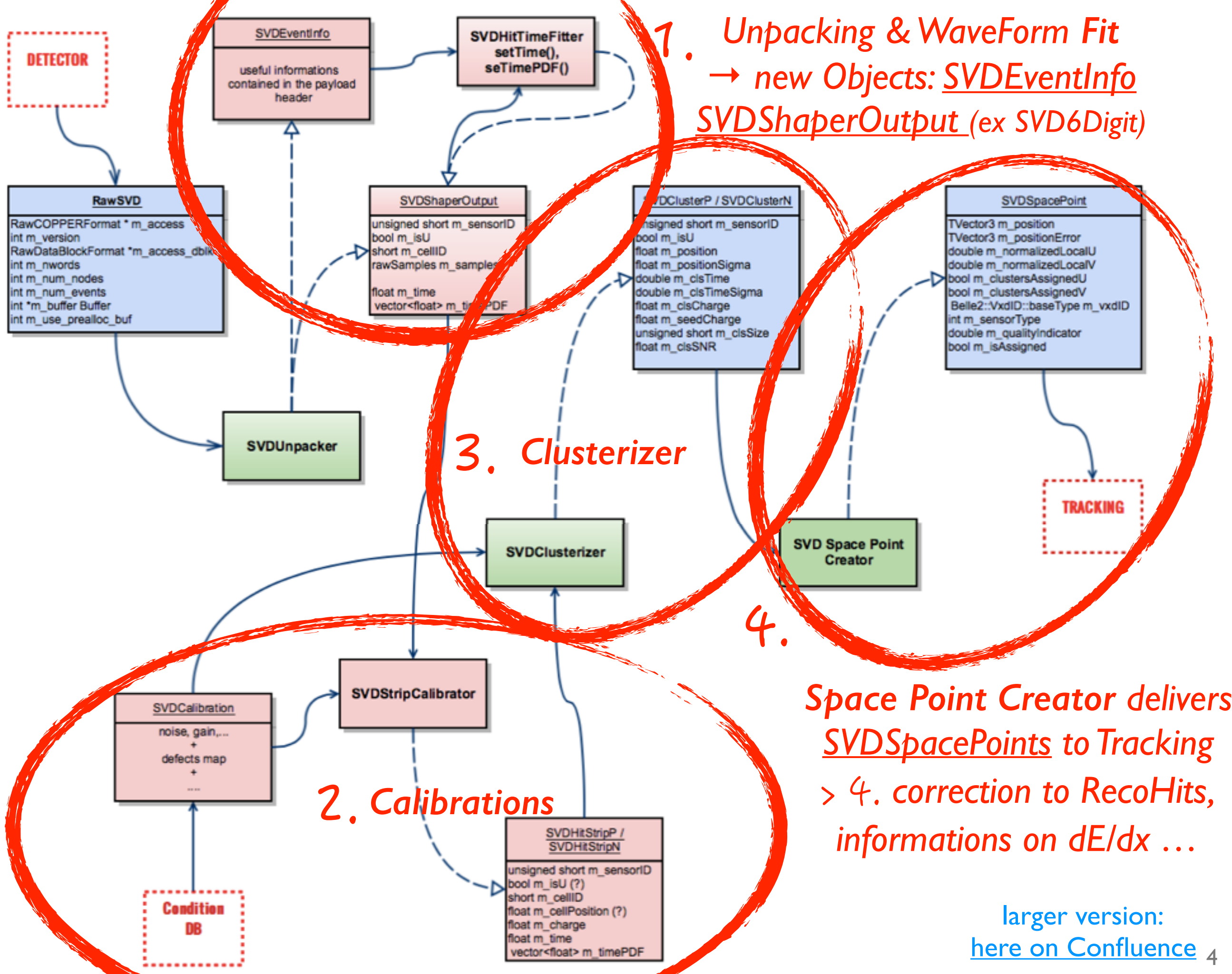


- ➔ We have implemented a basic design that has been working well up to now, it allows to reconstruct data from the test beams
- ➔ Peter's pull request includes improvements in the reconstruction, and more improvements are needed (e.g. calibration) → the design of the reconstruction needs a critical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

Proposal - July, 5th



larger version:
[here on Confluence](#)



Comments & Concerns

SVDShaperOutput
unsigned short m_sensorID
bool m_isU
short m_cellID
rawSamples m_samples
float m_time
vector<float> m_timePDF


- ➔ 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, ...)
 2. agreement on the fact that it allows to test different algorithms for the same tasks (e.g. time determination algorithm)
 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
 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”

Goals and Tentative plan

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

week month	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

➔ This will allow to:

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

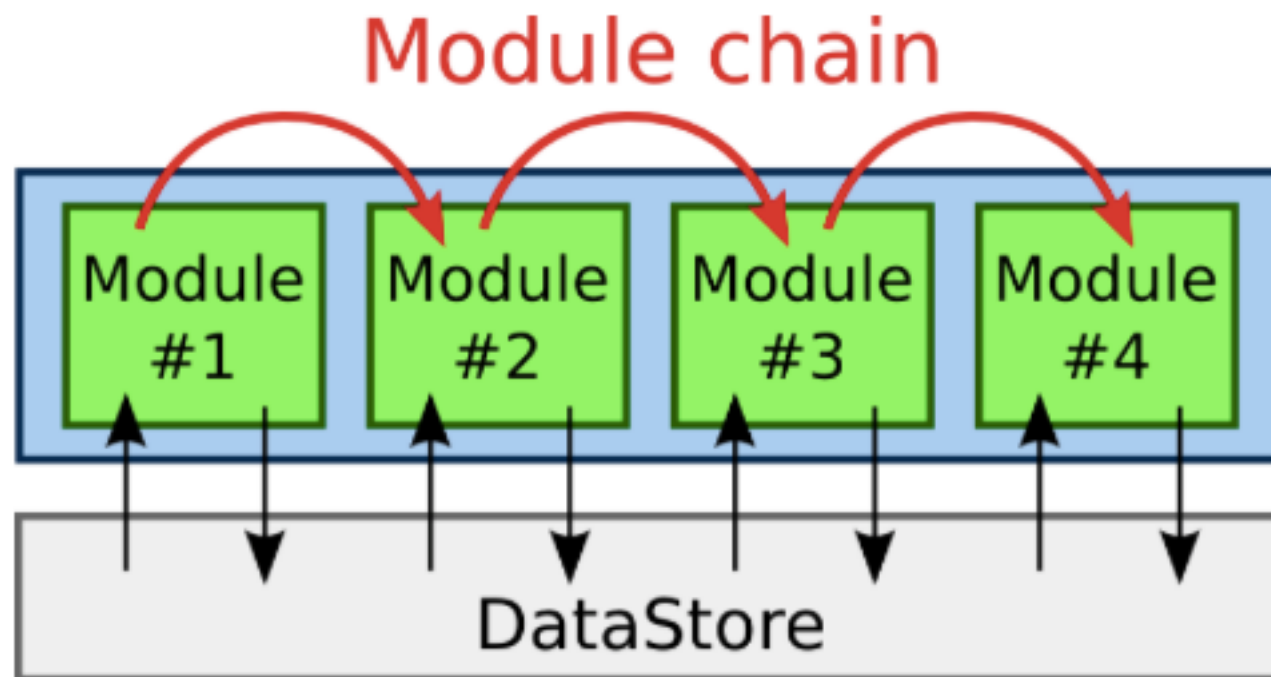
Why Redesigning the Code

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

basf2 design

The basf2 **module** is the key component of the basf2 framework. It is the "worker" of the framework, doing all the work related to event processing.



Path

steering scripts:

- for users: prepared by the svd group, checked and updated when needed
- for developers: the developer knows at least the basics of each module and each data object.

The **data store** is basf2's interface for storing data that needs to be exchanged between modules or read from/saved to file. It can be used to store single objects as well as arrays, provided they follow some simple rules.