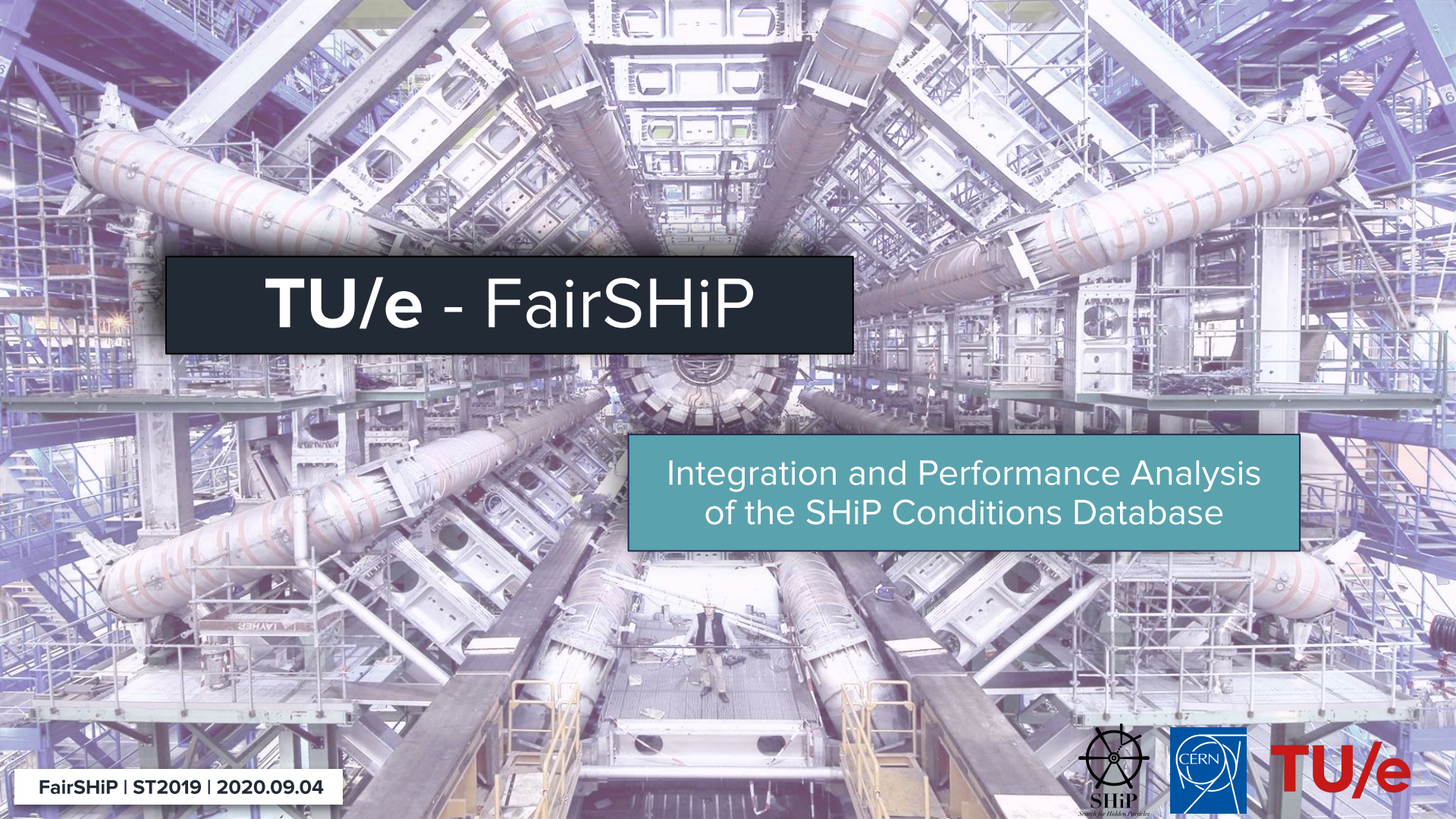


TU/e - FairSHiP

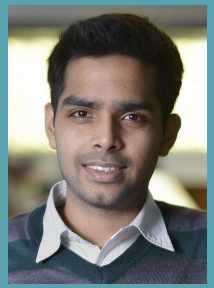
Integration and Performance Analysis of the SHiP Conditions Database

Agenda

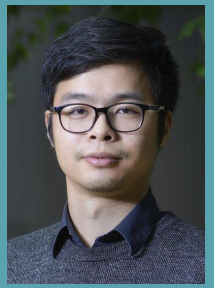
Agenda

Team Charter

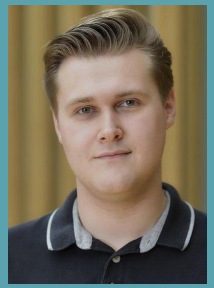
George Azis
Developer



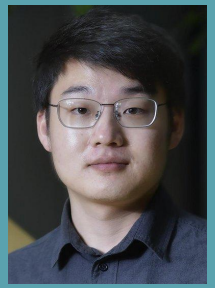
Nathan D'Penha
Developer



Han Gao
Developer



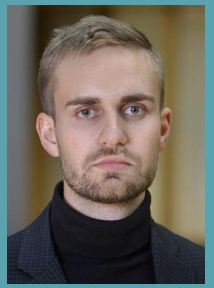
J.J.M.J. van der Heijden
Developer



Yitian Kong
Configuration Manager

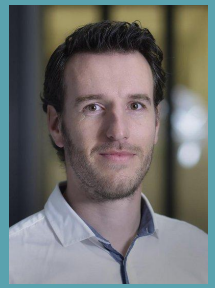
Christopher O'Hara
Project Manager

Yusril Raji
Scrum Master



Vladimir Romashov
Quality Manager

Raha Sadeghi
Test Manager



T.J.G.M. Vrancken
Architect

Introduction - What is FairSHiP?

Experiment

- FairSHiP is the simulation & reconstruction framework based on FairROOT and ROOT
- ***Search for Hidden Particles* (SHiP)**: Particle physics-based experiments for Hidden Sector
- Hopes to find evidence of Dark Matter and Lepton-Flavor Violations

Location

- Physical: Onsite at CERN SPS Facility in Switzerland (and part of France)
- Virtual (LHC@home): Research groups around the world (over 600 institutes and universities)
- Atlas@home led to discovery of Higgs-Boson and 300 publications

Mission

- “Unite people from all over the world to push the frontiers of science and technology, for the benefit of all” (CERN)

Executive Summary

POP
D&A
NFR

Purpose of Project

POP1	Integrate the SHiP Conditions Database into FairSHiP via aliBuild
POP2	Integrate the final product into an existing script (alignment data from testbeam)
POP3	Merge final product into the official FairSHiP repository

Design & Analysis

D&A1	Read & Write Conditions data into a database
D&A2	Analyze performance of the Conditions DB API (MongoDB)
D&A3	Check for edge cases during testing to ensure the API is problem-free

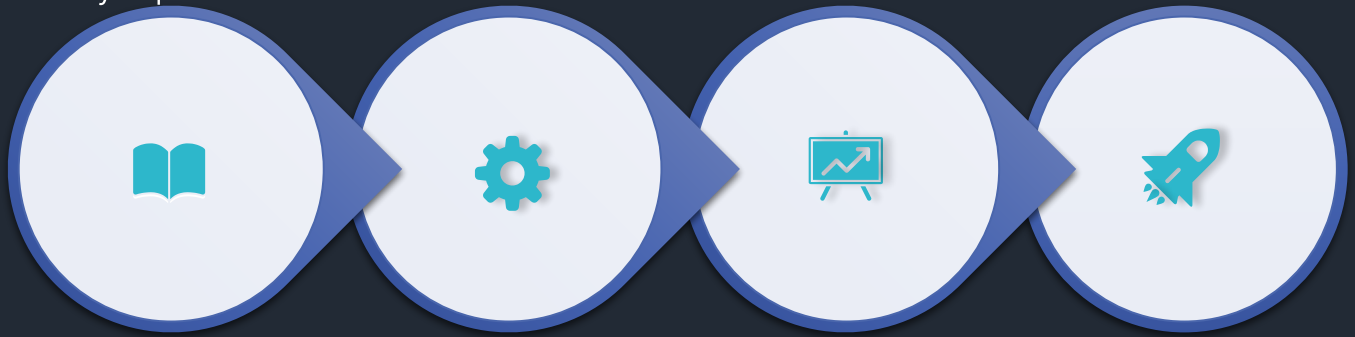
Non-Functional Requirements

NFR1	Ensure that the Conditions DB API is performant
NFR2	Create intelligible code and documentation for future useability
NFR3	Develop an extensible framework that is easily adaptable for new technologies

POP

STEPS TO SUCCESS

FIRST STEP
Conduct a detailed inspection of related works and consult directly with industry experts.



SECOND STEP
Setup up environments, databases, and containers. Derive requirements, functions, and implement continuously.

THIRD STEP
Test and benchmark API. Update architectures and merge results.

FOURTH STEP
Receive approval from CERN team. Merge Request to official repository.

TIMELINE

Our Brief Journey with Particle Physics

POP

TBD 2026
SHiP Experiment at SPS

04 FEB 2020
Project Kick-Off

Extract (non)-functional requirements

Ideate and prototype interface solution

EARLY APR 2020
Finalize Testing, Integration, and Benchmarking

Conduct Analysis and Write Documentation

Final Presentation and Merge

04 MAR 2020
Design Freeze!

Final Architecture

TBD 202X
Simulations for muon flux

Possible extension for other experiments and data

RISK MITIGATION

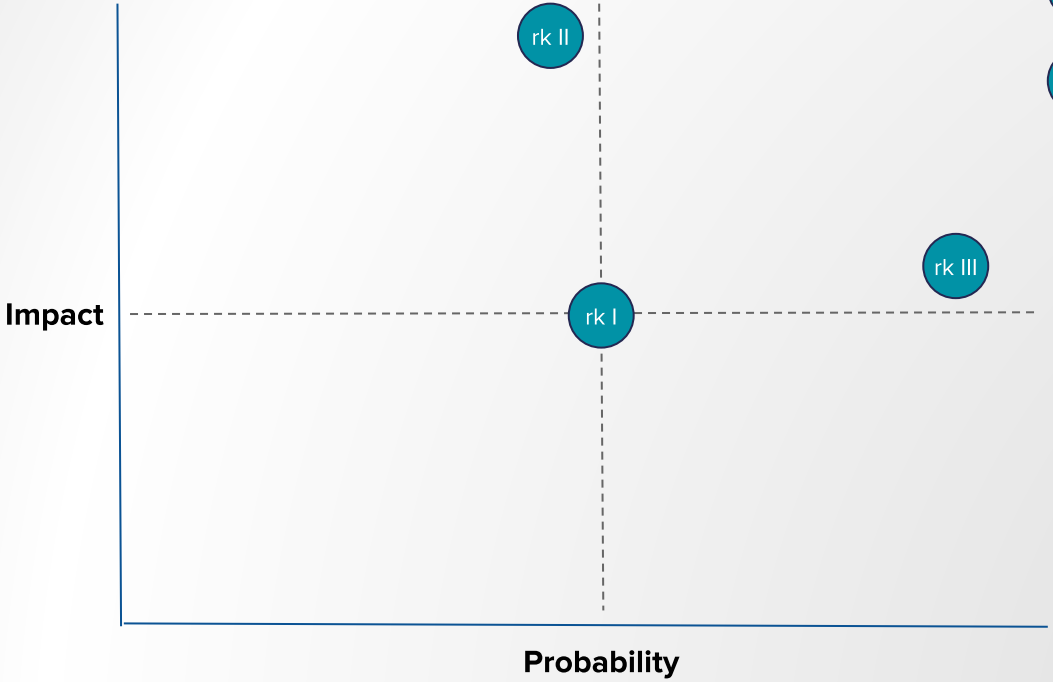
POP

Risk Avoidance and Mitigation Strategy

Risk (rk)	Details	Impact x Probability
I - Knowledge Deficiencies	Concerns: Lack of Technical Knowledge Plan: Consult industry experts and conduct research	Impact: Medium Probability: Medium
II - Deliver Wrong Product	Concerns: Mismatch in requirements Plan: Regular communication with CERN	Impact: High Probability: Low-Medium
III - COVID-19	Concerns: University shutdown, illness Plan: Create remote environments and contingencies	Impact: Medium Probability: High

Risk: Impact vs Probability

POP



- rk I — Knowledge Deficiencies
- rk II — Deliver Wrong Product
- rk III — COVID-19

Key criteria assessed

- Scope
- Deadline
- Quality
- Future Adjustments
- Technical Limitations

Agenda

Architecture

From Requirements to Model

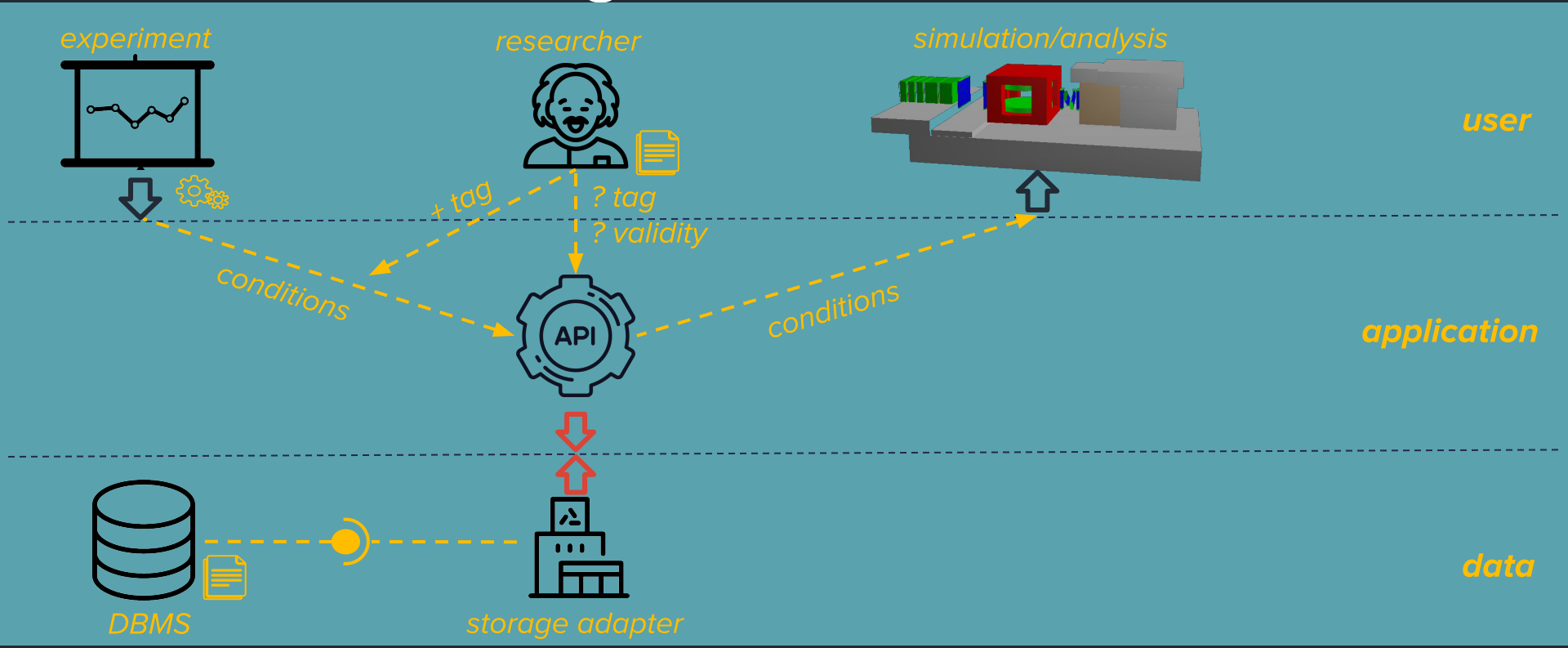
Architecture - Requirements

D&A
NFR

- The system shall be integrated into FairSHiP via aliBuild
- The system shall be able to access SHiP Conditions data (r/w)
- The system shall support MongoDB as DBMS
- The system shall be performant when running a local MongoDB-server instance

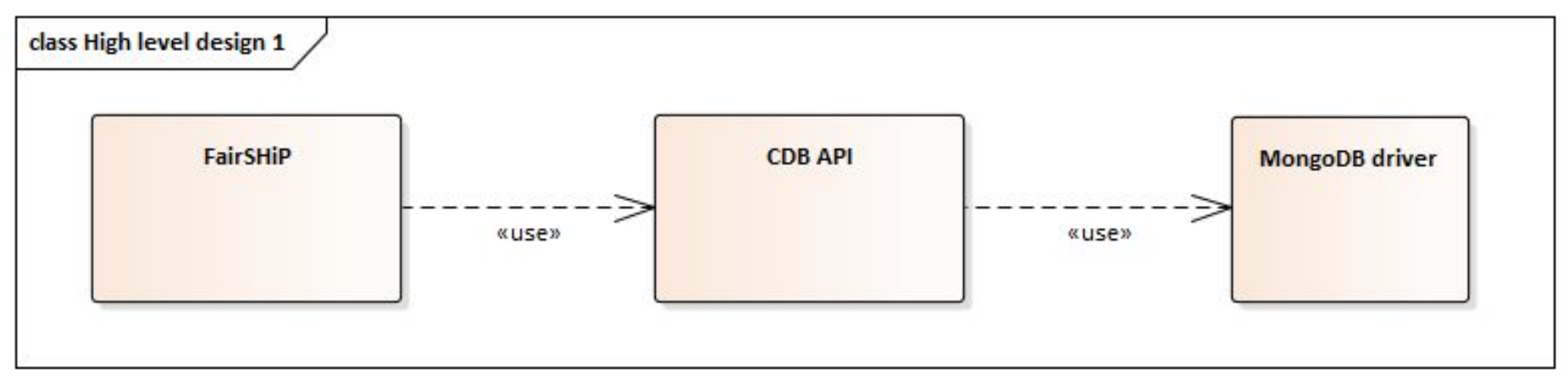
Architecture - High-Level

D&A
NFR



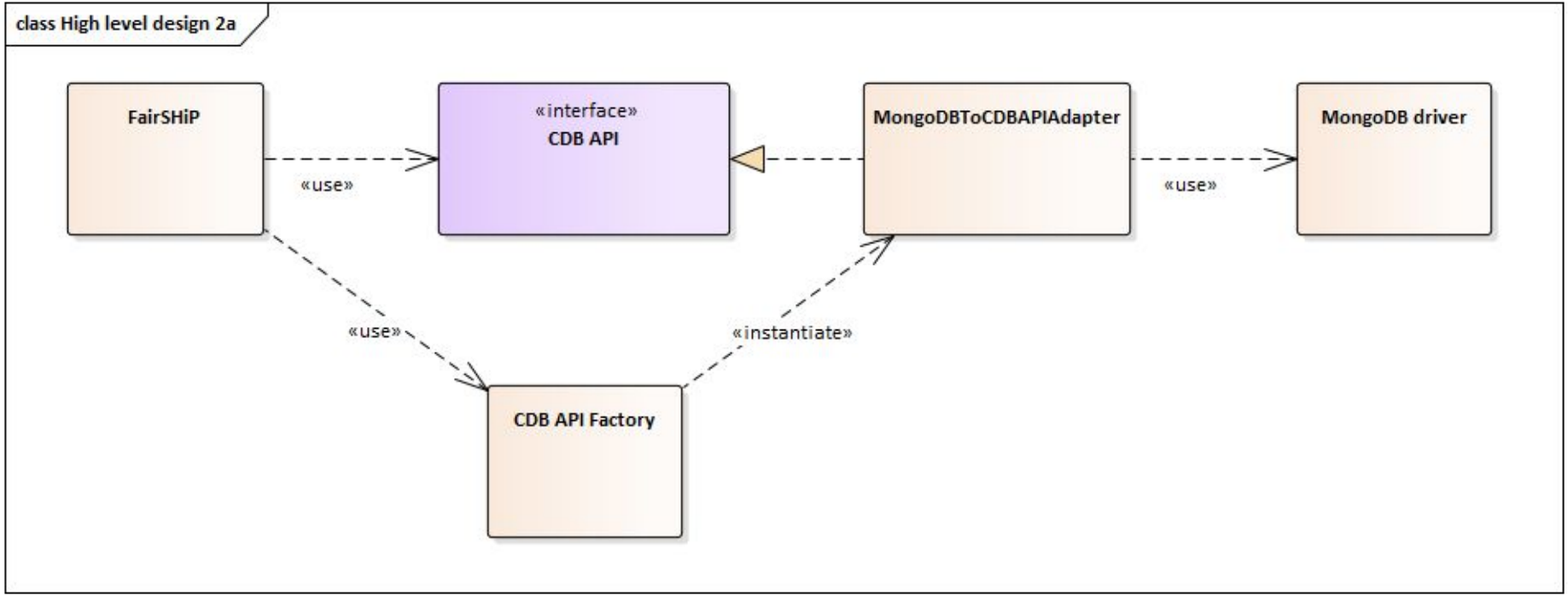
Architecture - Design 1

D&A



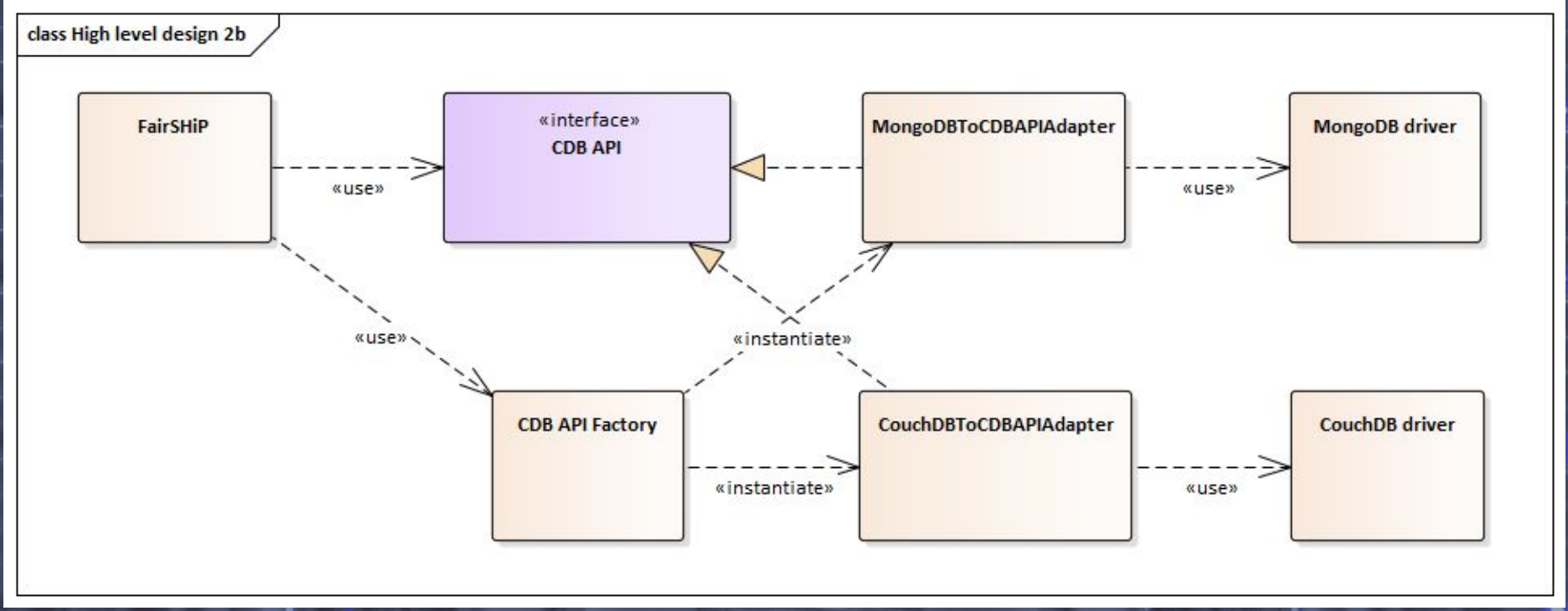
Architecture - Design 2a

D&A
NFR



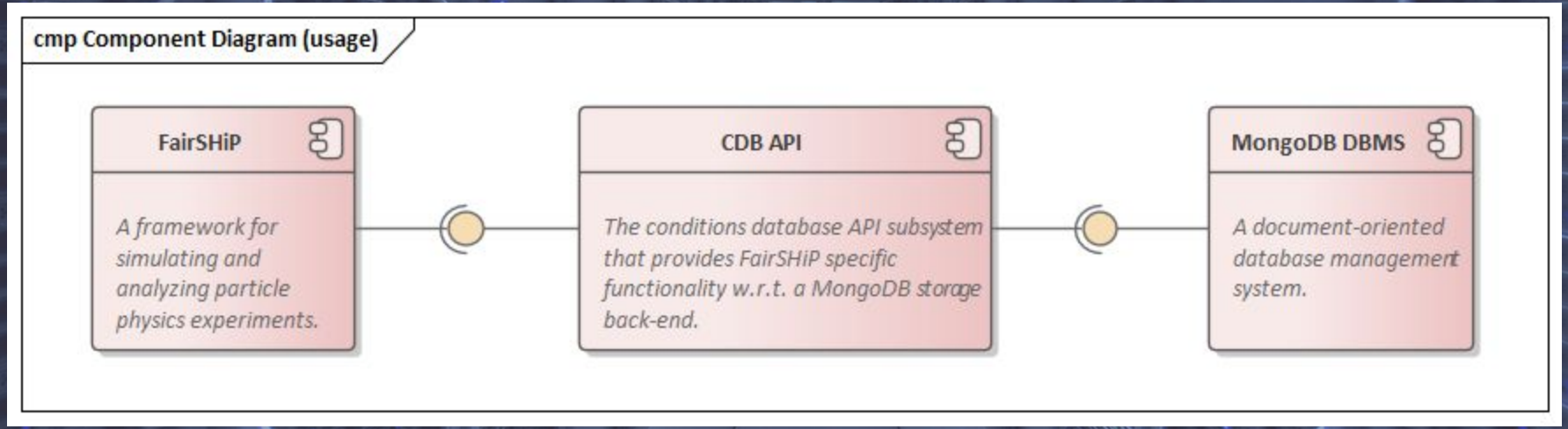
Architecture - Design 2b

D&A
NFR



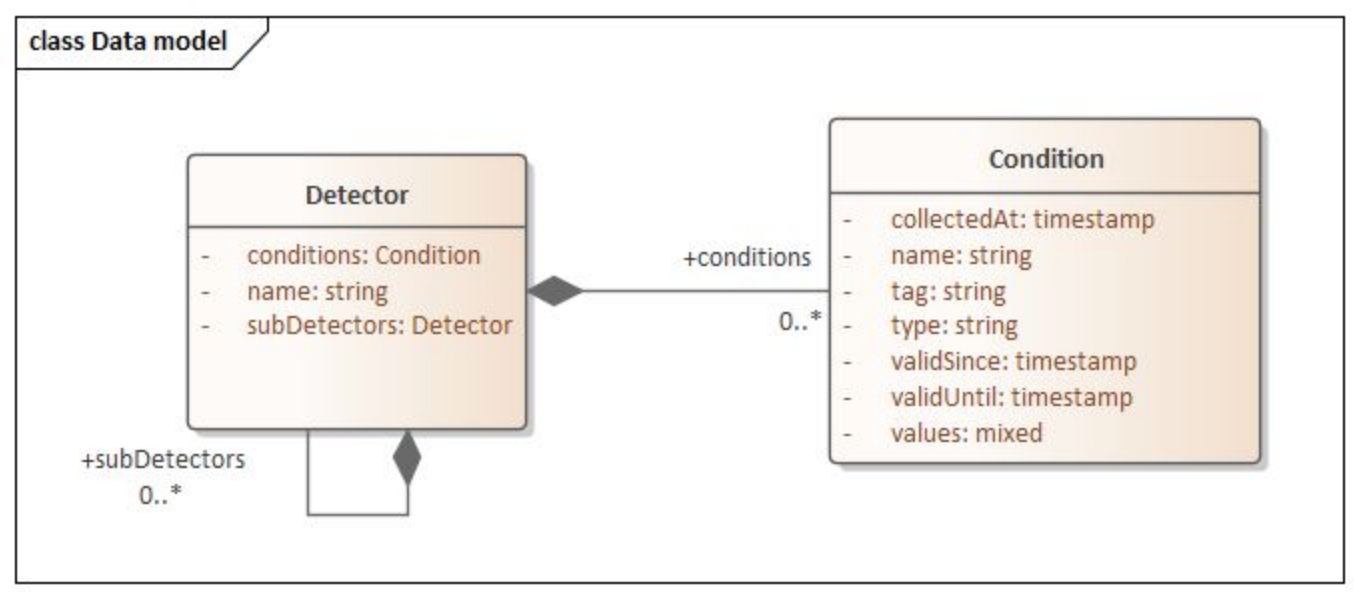
Architecture - Components

D&A



Architecture - Data Model

D&A
NFR



Methodologies - Technologies

D&A
NFR

Python	Interpreted, high-level, general-purpose language
MongoDB	Cross-platform document-oriented database
MongoEngine	Python object data mapper (structure)
PyTest	(Unit) Test framework for Python
PyLint	Source-code conformity checker for Python
Docker	Distributed runtime containers
GitLab CI	Continuous Integration Environment
aliBuild	Compiler/Build Tool for ALICE Experiments
Doxygen	Documentation Generator

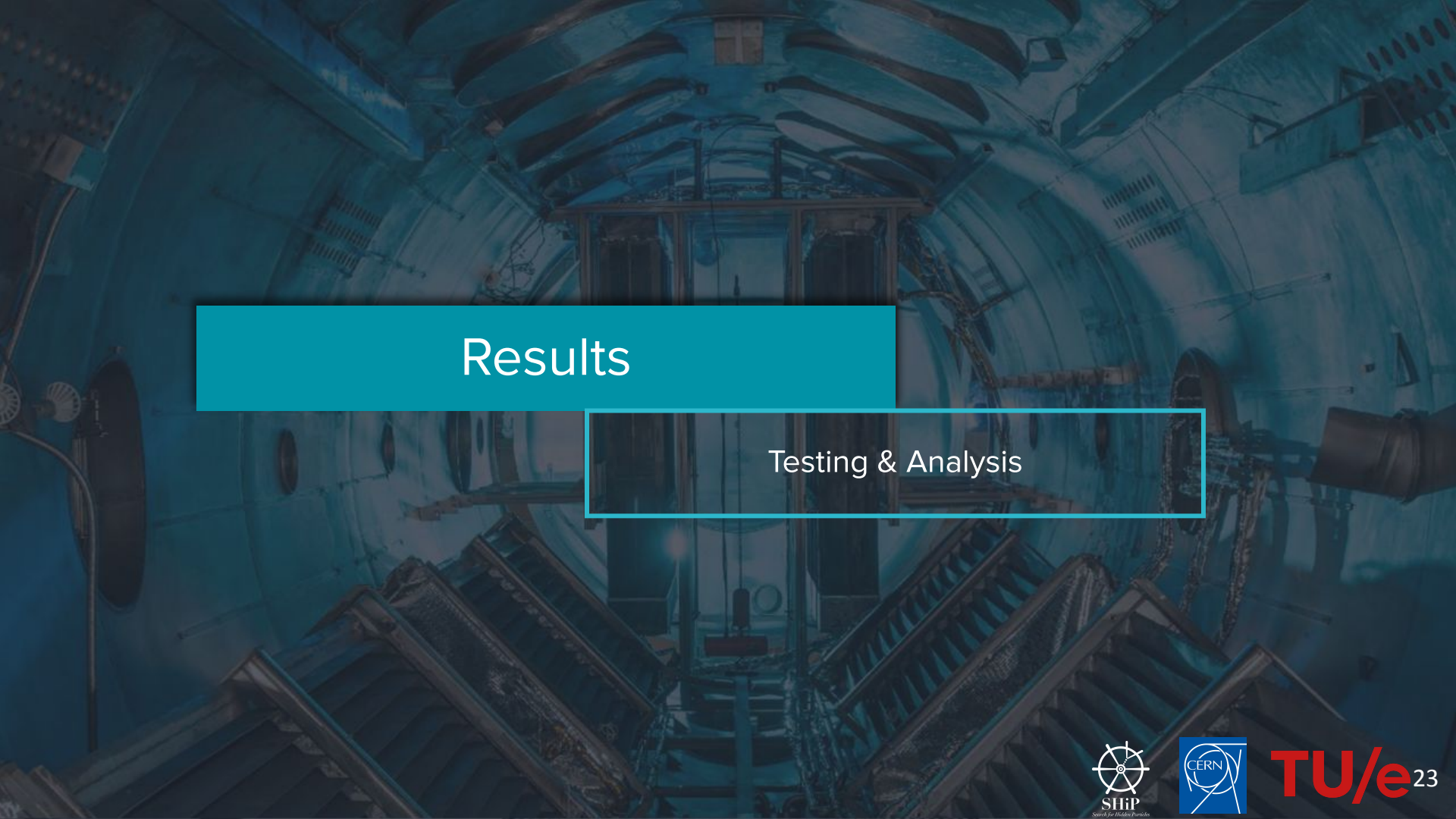
Examples of Functions

D&A
NFR

After any database has been constructed via a configuration file:

- list_detectors( )
- add_detector( ) ; remove_detector( )
- get_conditions( )
- get_conditions_by_tag( )
- get_conditions_by_name_and_validity( )
- get_condition_by_name_and_tag( )
- get_condition_by_name_and_collection_date( )
- update_condition_by_name_and_tag( )

Agenda



Results

Testing & Analysis

SHiP
Search for Hidden Particles

CERN

TU/e

Results - Testing/Benchmarking

D&A NFR

- Test cases derived from Business Requirements (Traceability)
- Automated testing for most cases using GitLab CI
  - Some items were manually tested
- Compared *features* and *steps* for *wall time* and *CPU time* (baseline vs implementation)
- Conducted Stress Test to evaluate use cases
- 147 Unit Tests (All Passed)

Results - Testing

D&A NFR

Coverage report: 98%

filter...

Module ↓	statements	missing	excluded	coverage
/home/yusril/Repository/CERN ST 2019/implementation/conditionsDatabase/databases/mongodb/__init__.py	0	0	0	100%
/home/yusril/Repository/CERN ST 2019/implementation/conditionsDatabase/databases/mongodb/models/__init__.py	0	0	0	100%
/home/yusril/Repository/CERN ST 2019/implementation/conditionsDatabase/databases/mongodb/models/condition.py	9	0	0	100%
/home/yusril/Repository/CERN ST 2019/implementation/conditionsDatabase/databases/mongodb/models/detector.py	6	0	0	100%
/home/yusril/Repository/CERN ST 2019/implementation/conditionsDatabase/databases/mongodb/models/detectorWrapper.py	5	0	0	100%
/home/yusril/Repository/CERN ST 2019/implementation/conditionsDatabase/databases/mongodb/mongodbadapter.py	424	8	0	98%
Total	**444**	**8**	**0**	**98%**

coverage.py v5.0.4, created at 2020-03-31 22:16

Results - Benchmarking (Features)

D&A NFR

	original [ms]		integrated [ms]		original - integrated [ms]	
features	wall time	cpu time	wall time	cpu time	wall time	cpu time
Daniel Retrieval	0.09	0.01	22.23	17.86	- 22.14	- 17.86
Persistent Alignment Constants	82.43	73.08	125.99	118.33	- 43.46	- 45.25
Import Alignment Constants	34.78	31.05	110.17	105.00	- 75.39	- 73.95

Results - Benchmarking (Steps)

D&A NFR

	original [s]		integrated [s]		original - integrated [s]	
step	wall time	cpu time	wall time	cpu time	wall time	cpu time
recoStep1	2359.26	2354.74	2433.88	2412.40	- 74.62	- 57.46
recoStep2	654.91	628.35	695.26	681.77	- 40.35	- 53.42

Results - Benchmarking (Stress)

D&A NFR

API Performance and reliability	wall time [s]	CPU time [s]
Add one detector	0.012	0.005
Add 100 detectors	0.190	0.170
Create *Complex Structure*	392.212	378.264
List Detectors *(> 100 detectors)*	0.389	0.370
Add one *massive* condition	0.372	0.355
Get Condition by name and tag	0.185	0.180
Update condition by name and tag	0.273	0.265

Agenda

Conclusion

Delivering the Product

Quality Assurance

NFR

- Strict review process
- Doxygen used for standardized code comments
- PEP8 style guide
- Traceability Matrix (Requirements to Test Case)

Conclusion

POP
D&A
NFR

- A production-level API was developed and integrated
- 98% coverage in testing
- Benchmarking revealed minimal additional overhead
- A demo video was created for knowledge transfer
- “Keeping it simple” can be challenging
- Communication is key

Acknowledgements

- Eric Van Herwijnen & Oliver Lantwin
- Yanja Dajsuren & Désirée van Oorschot
- Lecturers/Coaches
- Ad Aerts

Questions?

Thank you for your time!

References

- https://www.slac.stanford.edu/econf/C06092511/presents/TU002_PPT.PDF
- https://cds.cern.ch/
- ST-2017 Documentation
- https://ship.web.cern.ch/ship/FairShip/default.html
- https://github.com/ShipSoft/FairShip
- https://indico.cern.ch/event/482695/contributions/1159356/attachments/1226973/1796779/FairShip-Tutorial_Intro-Feb2016.pdf

Technologies - Doxygen

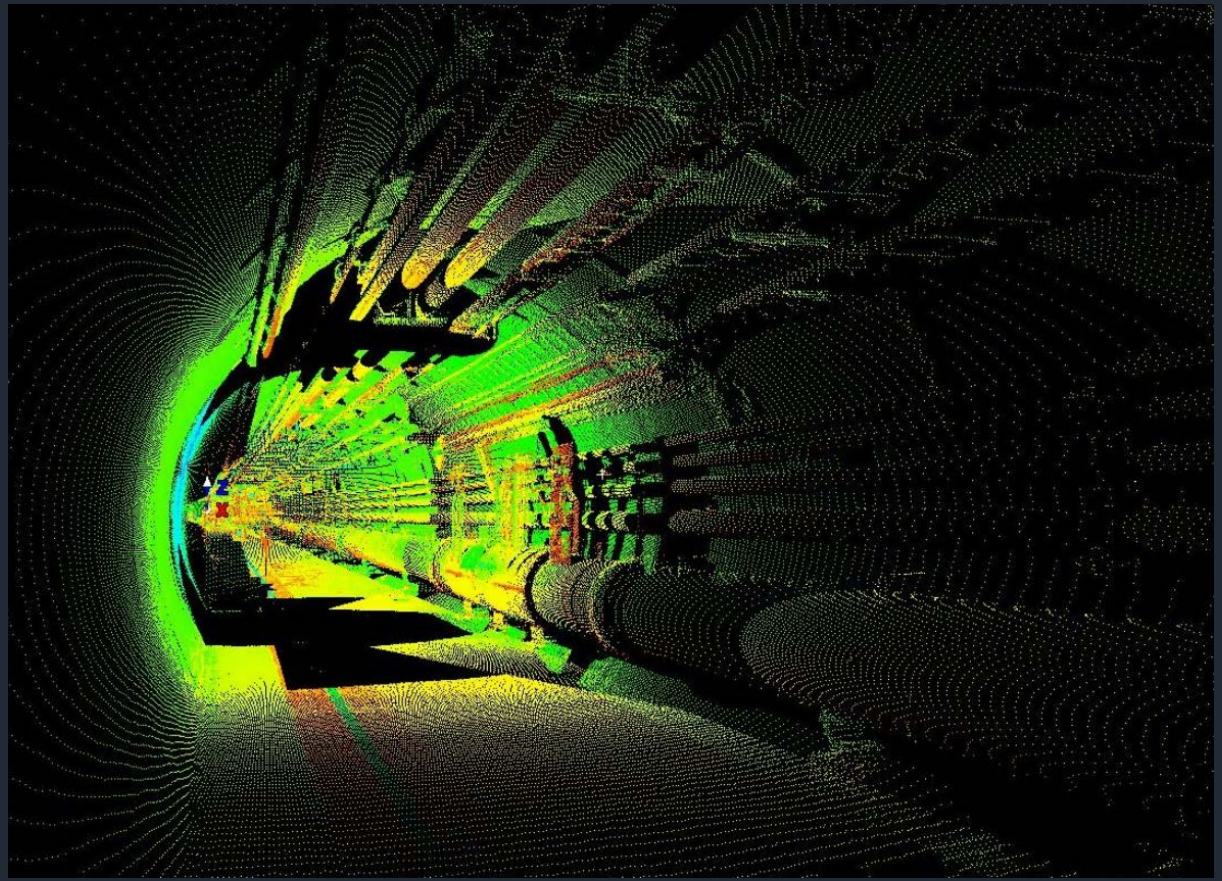
4.1.2.8 get_detector()

```
def interface.get_detector (
                self,
                detector_id )
```

Returns a detector dictionary.

Parameters

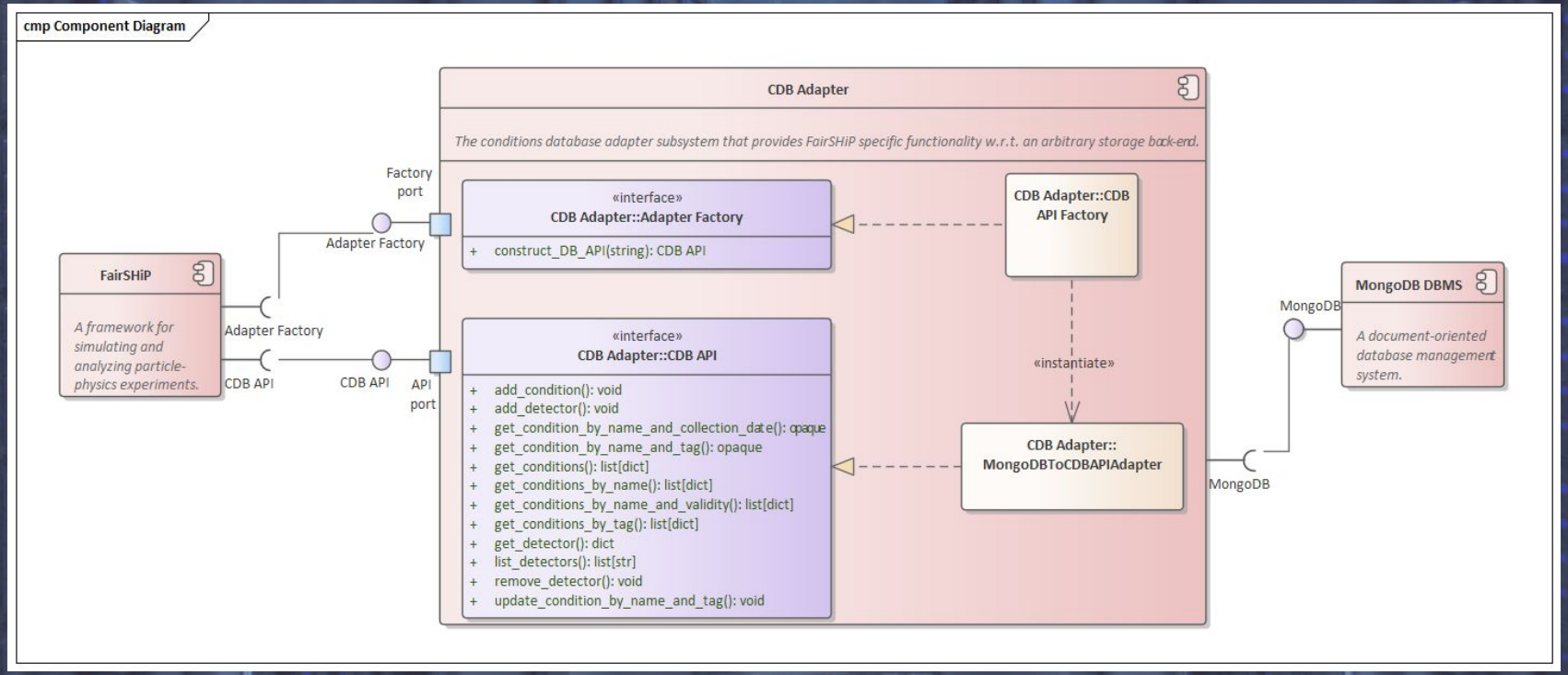
detector_id	String identifying the detector to retrieve (i.e. 'muonflux/straw_tubes').



www.slac.stanford.edu

Architecture - Components

D&A



Requirements

Number	Feature
1	The system shall store SHiP experiments' conditions data.
1-1	The system shall add a (sub)detector model.
1-2	The system shall add a condition model to a (sub)detector model.
2	The system shall retrieve SHiP experiments' conditions data.
2-1	The system shall retrieve a list with (sub)detector model names.
2-2	The system shall retrieve all condition models with a specific name for a given detector.
2-3	The system shall retrieve the condition model with a specific name and tag for a given detector.
2-4	The system shall retrieve the condition model with a specific name and collection date for a given detector.
2-5	The system shall retrieve all condition models with a specific tag.

Requirements

Number	Feature
3	The system shall remove a (sub)detector model and all associated conditions.
4	The system shall update a condition model validity interval and type.
4-1	The system shall update a condition model's validity interval.
4-2	The system shall update a condition model's type.
5	The system shall be integrated into FairSHiP such that it can be built with Alibuild.
6	When running a local MongoDB-server instance, the system shall return any set of condition models specified by a tag within 5s.
7	The system shall be extensible such that it can support multiple storage back-ends.
8	The performance deviation of the integrated system and the original one shall not be more than 20 percent.

Traceability

Req.	Prio.	Description	Related Test(s)
1	Must	The system shall store SHiP experiments' conditions data.	1, 5
1.1	Must	The system shall add a detector model.	1
1.2	Must	The system shall add a (sub)detector model to a detector model.	1
1.3	Should	The system shall remove a detector model and all associated conditions.	4
1.4	Should	The system shall remove a (sub)detector model and all associated conditions from a (sub)detector model.	4
1.5	Must	The system shall add a condition model to a (sub)detector model.	5
1.6	Must	The system shall update a condition model's validity interval.	11

Traceability

2	Must	The system shall retrieve SHiP experiments' conditions data.	3, 6, 7, 8, 9, 10, 12
2.1	Must	The system shall retrieve a list with (sub)detector model names.	2
2.2	Should	The system shall retrieve a (sub)detector model.	2
2.3	Must	The system shall retrieve all condition models for a given detector model.	12
2.4	Must	The system shall retrieve all condition models with a specific name for a given detector.	7
2.5	Must	The system shall retrieve all condition models with a specific name and validity for a given detector.	8
2.6	Must	The system shall retrieve the condition model with a specific name and tag for a given detector.	11
2.7	Must	The system shall retrieve the condition model with a specific name and collection date for a given detector.	10
2.8	Must	The system shall retrieve all condition models with a specific tag.	12

Traceability

3	Must	The system shall support MongoDB as DBMS.	19
4	Must	The system shall be integrated in FairSHiP such that it can be built with Alibuild.	15
5	Must	Integration of drifttubesmonitoring.py with the API	16
6	Should	When running a local MongoDB-server instance, the system shall return any set of condition models specified by a tag within 5s.	14, 17, 18
7	Could	The system shall be extensible such that it can support multiple storage back-ends.	19