SnT 2019
CTBT: SCIENCE AND TECHNOLOGY CONFERENCE

RNIAC: A cloud-based approach of the Radionuclide National Data Centre (NDC) in a Box software (RNIAB)

Author(s): Shaban LABAN, Freelance Consultant, E-mail: shaban.laban@gmail.com.

T3.5-P51

1. Introduction

The introduction of the Radionuclide NDC-In-A-Box (RNIAB) was a milestone in the treaty verification. The software is helping the Radionuclide community to get the latest and utmost of the Radionuclide data, analysis and review. However, maintaining and operating such complex software is a complex task and lacks the means of exchanging information and data analysis among the state parties. The aim of this research is to study the possibility of adopting, implementing and operating a cloud-based version of the RNIAB. The research benefits from the rapid and growing advances in communications, cloud solutions and services. Implementing such approach may increase the usage and operability of the Radionuclide software remotely and efficiently. Also, it can increase the exchange of information and Radionuclide data analysis among the National Data Centers as well as reduces efforts and time of maintainability of the software. The challenges to such approach will be presented. Also, the different cloud-based solutions and databases will be discussed. The methodology, models and techniques to achieve the proposed research will be introduced as well as future directions.

2.Challenges

The main challenges:

- The Radionuclide software
  - Lack Object Oriented Programming (OOP) Capabilities
  - Lack the support of Parallel/Asynchronous Spectrum Analysis and Revisions Capabilities
  - Requires Linux support and skills (Linux OS specifics)
  - Maintainability, Flexibility and Modular programming issues
- The Radionuclide Database
  - Complex structure
  - Lack of handling Parallel Spectrum Analysis and Review processes
- Data Exchange
  - Need to adopt/implement a standard simple and easy to use format for data exchange and processing
- Resilience
  - Need procedures/solutions for handling the interruption/loss of service or communication

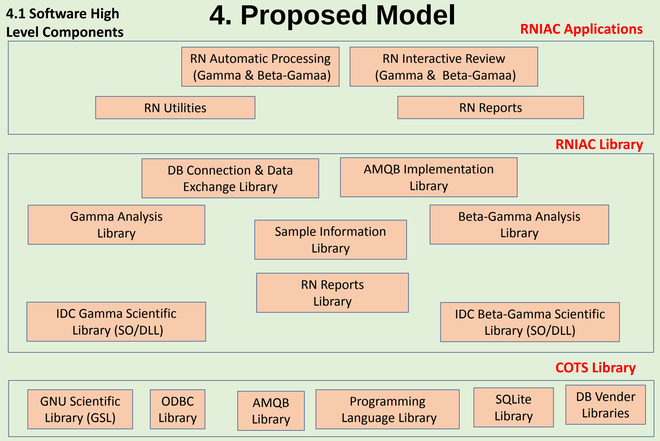
3. Proposed Approach

- Re-engineer Radionuclide software to support **Offline**, **Local**, **Remote** and **Cloud** analysis and review.
- Multiple Version Spectrum Analysis (**MVSA**) approach with new revision analysis tables. Introduction of **AnalysisID** column for processing tables in DB.
- Support any type of database vendors (Oracle, MySQL, Postgres or NoSQL(e.g. mongoDB).
- Introduce new RN binary product: Sample in Searchable File (**SSF**), SQLite format, to support data exchange and analysis reversions (SQLite is supported by almost all programming languages and free tools). All related configuration , raw data , processing and review results in **ONE BIG** Searchable Binary File.
- Exclude filesystem storage. All data and information are stored in ONE place (SSF, Database or Cloud.
- Support **Windows** for non-Linux users for simplicity and to increase usability.
- OOP implementation (**Python**, **C#** (.NET Core) and **Java**).
- Support flexible GUI interactive Review, step by step with decision, processing/analysis of the spectrum.
- Modularity: Possibility to plugin/addon other peaks, nuclides and/or any other scientific algorithms.
- Use messaging Queues (RabbitMQ) for communication and data exchange.

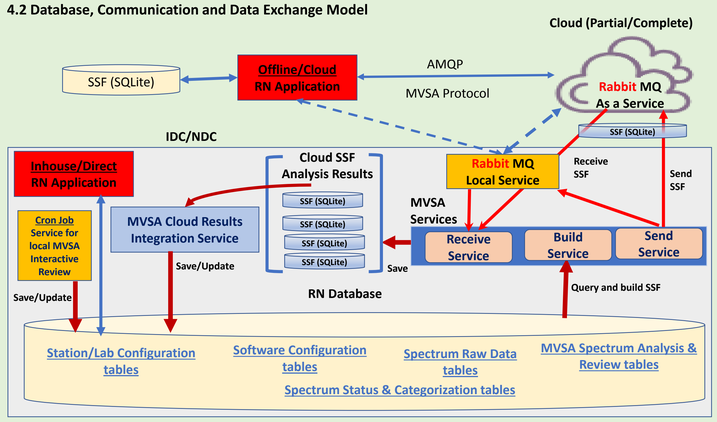
Disclaimer: The views expressed on this poster are those of the author and do not necessarily reflect the view of the CTBTO

4. Proposed Model

4.1 Software High Level Components



4.2 Database, Communication and Data Exchange Model



5.a Gamma Analysis Implementation Example

1. Analysis log diff between IDC(NIAB/Oracle) and Proposed Approach (Python/SSF, SQLite)

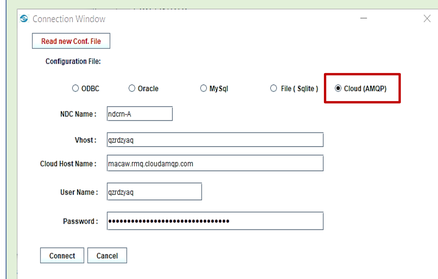
```
[ndcuser@localhost ~]$ diff -b /mnt/Share/3999002.py.txt
/mnt/Share/3999002.niab.txt
1c1
< Starting PySaint (PySaint version 1.0.0)
---
> Starting autoSaint (autoSaint version 2.1.3 build / from 2019-04-09 12:56:37)
40,41c40,41
<       ManualBaseline        : NULL
<       ManualScac            : NULL
---
>       ManualBaseline        :
>       ManualScac            :
91c91
< Reading from sqlite....
---
> Reading spectrum file
(/ops/products/rn/spectrum/2017/03/03//usp79_003_3999002g.s)
547c547
<         8  8.736e+01  2.654e+02 PB-212F, PB-214,
---
>         8  8.736e+01  2.654e+02  PB-214, PB-212F,
772a773,784
> Error executing the query ()
> Error getting average flow data ()
> Non-critical error while running QC routine (Flow 500)
> Error executing the query ()
> Error getting average flow data ()
> Non-critical error while running QC routine (Flow gaps)
> Error executing the query ()
> Error getting average flow data ()
> Non-critical error while running QC routine (Zero flow)
> Error executing the query ()
> Error getting average flow data ()
> Non-critical error while running QC routine (Flow)
776c788
< Finished successfully(xxxxxxx)
---
> Finished successfully (Tue Apr  9 15:42:37 2019)
```

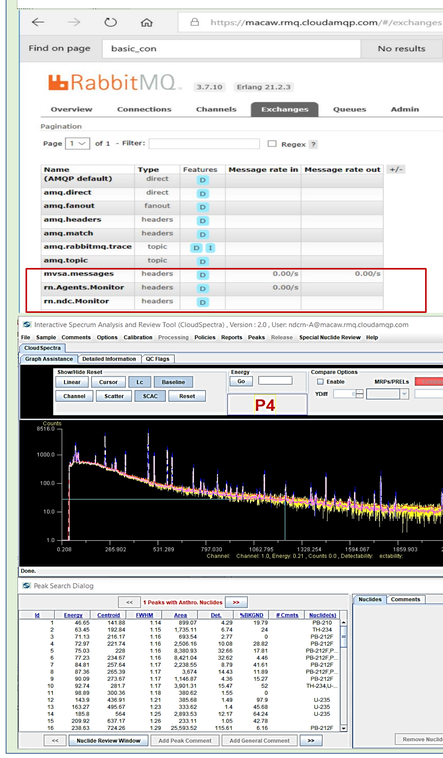
The Proposed approach has been implemented successfully for re-engineering the **autoSaint** program. The approach was implemented using OOP paradigm and in both **Python** and **C# programming** languages. The log comparison for a selected sample analysis between IDC/NIAB, **autoSaint**, and Python implementation , **PySaint**, is show in 1. While the log comparison for the same selected sample analysis between C# implementation, **NetSaint**, and Python implementation , **PySaint**, is show in 2 and shows consistency in analysis results .

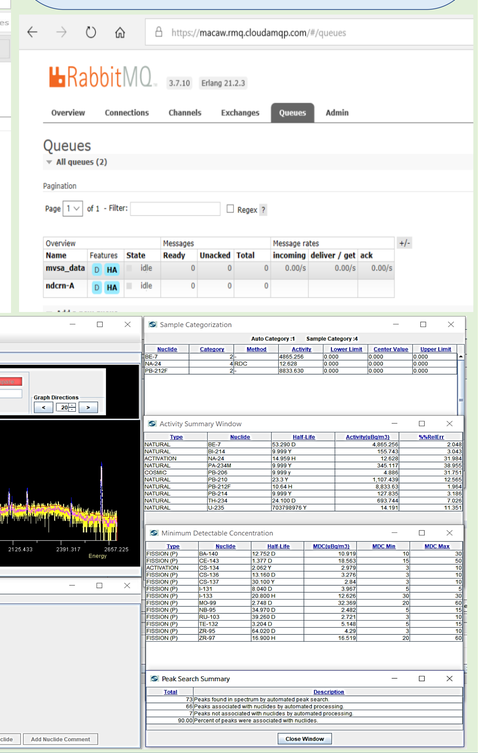
2. Analysis log diff between Proposed (Python/SSF, SQLite) and Proposed C#/ODBC (Oracle)

```
[ndcuser@localhost ~]$ diff -b /mnt/Share/3999002.py.txt
/mnt/Share/3999002.net.txt
1c1
< Starting PySaint (PySaint version 1.0.0)
---
> Starting NetSaint (NetSaint version 1.0.0)
91c91
< Reading from sqlite....
---
> Reading spectrum file
(/ops/products/rn/spectrum/2017/03/03//usp79_003_3999002g.s)
547c547
<         8  8.736e+01  2.654e+02 PB-212F, PB-214,
---
>         8  8.736e+01  2.654e+02  PB-214, PB-212F,
```

5.b Interactive & Cloud RabbitMQ Implementations Example







The SSF (SQLite database) was designed accordingly for supporting the MVSA approach. The SSF is a light-weight and self contained file-based and In-Memory database that will be used for both automatic and Interactive applications. The necessary tool/program was designed for collecting sample raw data, related configuration and analysis information as well as creating and populating the SSF database.
For Cloud testing purposes, CloudAMQP was used as a free version of RabbitMQ as a Service. The necessary prototype programs, exchanges and queues were designed and implemented as necessary. The **OpenSpectra** software that is included in the RNIAB was redesigned to support AMQP communication and SSF data analysis. Implementation Examples for interactive review, connecting and using RabbitMQ are shown in this section.

6. Conclusion and Future Work

- The re-engineering for **autoSaint** was successful and was implemented in both **Python** and **C#**.
- Implementing **autoSaint** and **bg_analyze** in Java to be done in future.
- Database update to support **MVSA** is done. Also, extracting all spectrum data and information into SSF is done.
- A free Cloud support is used for implementing **RabbitMQ** as a service (CloudAMQP).
- Necessary AMQP prototype programs, exchanges and queues were implemented successfully.
- The **OpenSpectra** software was updated accordingly to support both CloudAMQP and SSF.
- Re-engineering of **bg_analyze** and reports is **ongoing**.
- Build a **Windows** version for interactive review and/or updating INSPIRE will be done in future.
- Porting the RN database to **Postgres** was done as well, but need more tuning for some columns data types.
- Porting to NoSQL (**Mongodb**) is ongoing.
- More real-time testing and comparison will be conducted either with v-DEC, IDC operational database or RNIAB.

7. References

1. IDC NIAB Software, IDC Documentations and related RN Components Detailed Design Descriptions, http://ww.ctbto.org/.
2. IDC Database Schema and IMS Formats and Protocols, http://www.ctbto.org/.
3. RabbitMQ: An open-source lightweight, fast and reliable message-broke software, https://www.rabbitmq.com/.
4. AMQP: Advanced Message Queuing Protocol, https://www.amqp.org/.
5. SQLite: A small, fast, self-contained, high-reliability and full-featured SQL database engine, https://www.sqlite.org/.
6. GSL: The GNU Scientific Library for C , C++ programmers, https://www.gnu.org/software/gsl/.