Iterative and incremental development of the ATLAS Publication Tracking System

Ana Clara Loureiro Cruz[1,***], *Carolina* Niklaus Moreira Da Rocha Rodrigues[1,], *Gabriel* de Aragão Aleksandravicius[1,], *Gabriela* Lemos Lúcidi Pinhão[2,], *Pedro Henrique* Goes Afonso[1,], *Rodrigo* Coura Torres[1,], and *José Manoel* Seixas[1,]

[1]Laboratório de Processamento de Sinais, COPPE/EE/IF
[2]Laboratório de Instrumentação e Física Experimental de Partículas - LIP, Lisbon

Abstract. The ATLAS experiment is a particle physics experiment situated at the Large Hadron Collider (LHC) at CERN. It involves almost 6000 members from approximately 300 institutes spread all over the globe and more than 100 papers published every year. This dynamic environment brings some challenges such as how to ensure publication deadlines, communication between the groups involved, and the continuity of workflows. The solution found for those challenges was automation, which was achieved through the Glance project, more specifically through the Glance Analysis systems, developed to support the analysis and publications life cycle in 2011. Now, after twelve years, in order to satisfy the experiments' most recent needs, the systems need code refactoring and database remodelling. The goal is to have only one system to accommodate all the analysis and publications workflows, the so-called ATLAS Publication Tracking system, an evolution of the current Analysis systems. This project includes a database remodelling that reflects the hierarchical relation between analyses and publications; a code base that supports non-standard workflows; the expansion of the current API so all the authorized ATLAS members can access ATLAS publication data programmatically; a service-oriented architecture for integration with external software, such as GitLab; the creation of an automatic test environment, which assures the quality of the systems on each update. The ATLAS Publication Tracking system is a long-term project being developed with an iterative and incremental approach, which ensures that the most valuable tools are implemented with priority while allowing a smooth transition between the old systems and the new one.

1 Introduction

The ATLAS experiment [1] is the largest experiment running at the Large Hadron Collider (LHC) at CERN. It has approximately 6000 active members around the globe, working in different institutes and different time zones. Coordinating the efforts of so many people across different time zones and cultures can be challenging, and the volume of data that needs to be managed can be overwhelming.

[*]e-mail: ana.clara.loureiro.cruz@cern.ch
[**]Copyright [2023] CERN for the benefit of the ATLAS Collaboration. CC-BY-4.0 license

The Glance team's [2] primary goal is to ensure that the experiments' data is managed efficiently and effectively. Currently, the Glance team is present in three experiments: ATLAS, ALICE and LHCb. In the ATLAS experiment, the team provides interfaces to manage various entities, such as member information, employment records, appointment assignments, paper submissions, and speaker selection.

In the context of managing the ATLAS publications, Glance offers the Analysis System to automate the processes related to the elaboration, revision, and publication of papers, public notes, conference notes and plots. This system aims to ensure the publications' deadlines, the communication between groups, and the continuity of the workflows. It has been operating for 12 years, and has around 200 users accessing it every day, publishing more than 100 papers per year.

Since the development of the Analysis System in 2011, some issues have been identified:

- The system was designed for simpler publication workflows, but the requirements have evolved and become more complex over time. The existing code base in the Analysis System is not prepared for non-standard workflows and it struggles to adapt to the increasing complexity.
- The database modelling of the Analysis System includes redundant data storage, so the users need to manually synchronize data, resulting in duplicated information and introducing potential problems with data consistency.
- The outdated code base is challenging when implementing and improving features. This results in compatibility issues with newer technologies, potentially compromising long-term code maintenance.

These problems in the Analysis system can not be solved easily without refactoring the code base and the database. The challenge in this scope is to make the necessary refactoring to meet the users' and the developers' needs in a system that is already being largely used.

2 Project

The ATLAS Publication Tracking System aims to become a newer and updated version of the Glance Analysis System. This project was designed to solve the problems observed in the previous system by employing an incremental development method. The goals of this project are:

- Refactor the code base in order to facilitate the systems' maintenance and implementation of new features.
- Remodel the database in order to avoid duplication of data.

Implementing core changes on a system that is widely used can be very challenging. The approach used to make this refactoring was the Iterative and Incremental Development (IID) [3]. This method was applied along with an agile project management [4], allowing the developer to focus on smaller and more manageable implementations that can be delivered in iterations.

The process is illustrated in Figure 1. After identifying the changes needed to be made to the system on the initial planning, it is possible to divide the deliveries into packages and develop each one separately and in order of importance. One crucial step in this method is the testing and feedback phase. This step allows the developer to catch bugs and revise the implementation with the user. The ATLAS Publication Tracking System was planned following this approach, and it is currently in the deployment phase of the first iteration.

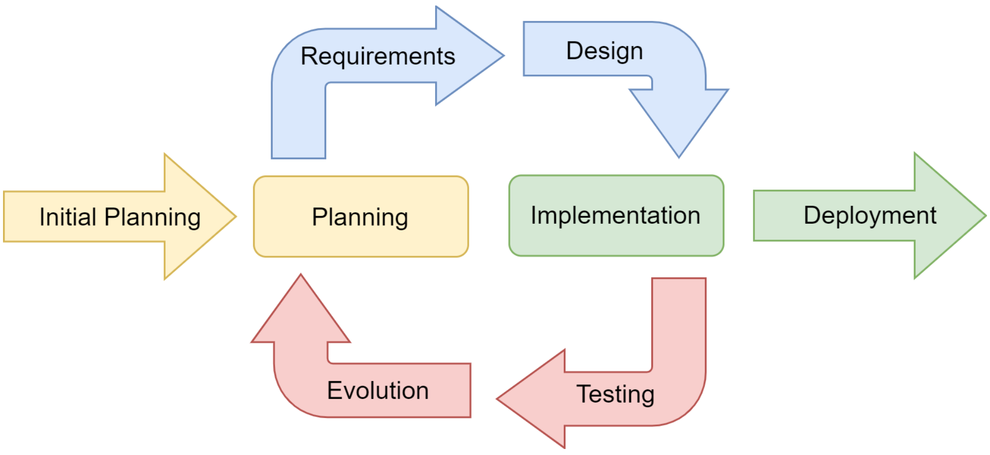

Figure 1. Diagram of Iterative and Incremental Development approach.

3 First Iteration

For the first iteration, the main goal was to remodel the database in order to avoid data duplication.

3.1 Database Changes

The new modelling of the database was designed to be normalized and avoid data duplication [5]. The normalization of the database results in easier queries, more comprehensive structure and reduction of redundancies.

One of the changes made was implementing a single source in the database for data that is common to the different kinds of documents. According to the system requirements, the data from a document flows from phase 0 (physics analysis) to phase 1 (paper, conference note or public note).

In the Analysis System database modelling (Figure 2), the data referring to a document would be saved twice in the database (both on the phase 0 and the phase 1). This replication of data doesn't happen in the normalized Publication Tracking database (Figure 3). The new modelling included the creation of a entity named 'Task' that would store all the data shared by the different kinds of documents (such as short title and full title). All of the data related to the Task entity was stored in a new database schema, allowing the system to be modular, preventing unintended interactions between different systems.

Besides removing data duplication, the normalized design provides consistency within the database, better executed and easier to build queries.

3.1.1 History Implementation

The database modelling also adds tracking capabilities of all data changes performed by users, resulting in a fully auditable database. The result is valuable both for the developer and for the user, facilitating the tracking of bugs and also the recovery of data if needed to retrieve information from a specific date.

This tracking was achieved by utilizing database triggers that write in the history schema every time an action (inserting, updating or deleting) is executed on the main schema.

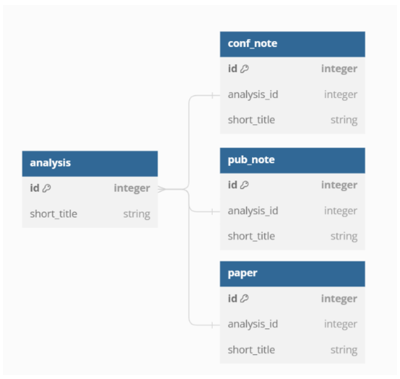

Figure 2. Model of denormalized schema

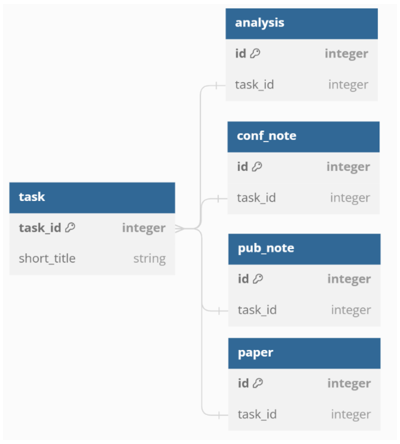

Figure 3. Model of the refactored normalized schema

3.1.2 Database Versioning

In order to keep track of all the changes done on the remodelling of the database and the data migrations, Liquibase [6] was used as a database versioning tool. This tool facilitates the establishment of a timeline for the changes, storing different states of the database.

The usage of this tool resulted in the ability to easily navigate between different states of the database. This capability ensured the flexibility to roll back and re-run the migrations when necessary during the development phase, both for data transactions or structural changes, such as adding a column to a table. Liquibase also provides additional advantages, such as the ability to specify contexts and labels to changesets [6].

3.2 Code Changes

In response to the database changes, it was raised a need to re-write some parts of the code that had been written more than 5 years ago. The principles of Domain Driven Design (DDD) [7] were adopted to provide a foundation for improving code testability, making it easier for the system to have automated tests. This approach automated the testing phase in the IID method. This step not only improved code testability, but also facilitated code readability and made code maintenance more efficient.

3.3 Automated Tests

In this iteration, a significant effort was made to implement automated testing. This emphasis was due to the fact that crucial changes were being done to a stable system, so the risk of introducing bugs to the deployed version is mitigated.

The incorporation of automated tests brings the following advantages:

- Quality control: early catching of bugs;
- Confidence in deployment: shields the working features from changes that can break its functionality;
- Documentation of use cases: the characterization tests documents the behaviour of specific pieces of code [8], this makes the code clear for the next generations of developers.

The implementation of the tests was done using the testing framework PhpUnit [9] and integrating it with Gitlab CI [10]. With that, every time a code change is performed, the tests run and, if any test fails, the change is not approved. The results of the implementation of automated testing were positive for the team, as dozens of bugs were easily found in the development life cycle, where fixes are usually easy to be implemented.

4 Conclusion and next steps

The ATLAS experiment conducted at the LHC generates a substantial volume of scientific documents. In this context, an automated system to manage publications is indispensable to ensure the workflow that promote the quality of the documents produced. The Analysis System, responsible for this automation, has a code base and database that have become outdated, presenting challenges of data synchronization and rigidity of workflow. To address these limitations, the ATLAS Publication Tracking System was developed using an IID approach.

This project is currently in the deployment phase of the first iteration. In order to achieve the final version of the system, other iterations are already mapped to be done, such as a broader code refactoring, flexibilization of the workflow and expansion of the API endpoints.

The IID approach allows the developer to constantly revisit the requirements with the users. The first iteration focusing on database remodelling was very tricky and involved work both in the code and in the migration of data. For this iteration, the results were significant in multiple fronts. The database remodelling effort has implemented a single source of truth for data flowing from phase 0 to phase 1. The integration of history tracking is important to the system's reliability and maintenance. The utilization of a database versioning tool has offered flexibility in managing database changes. The code refactoring performed so far resulted in better code testability and maintainability. At last, the implementation of automated testing has allowed the team to spot bugs prior to the project deployment.

Finally, the development of this new version is a long term project, that should be delivered with multiple iterations of the Iterative and Incremental Development method.

The authors would like to thank CNPq, CAPES, FAPERJ, RENAFAE (Brazil), as well as CERN and the ATLAS collaboration for providing support for this work. We thank Mr. Joel Closier for supporting the presentations and revising the work.

References

[1] ATLAS Collaboration, JINST **3**, S08003 (2008)

[2] C. Maidantchik, F.F. Grael, K.K. Galvão, K. Pommès, J. Phys.: Conf. Ser. **119**, 042020 (2008). DOI: 10.1088/1742-6596/119/4/042020

[3] C. Larman, V.R. Basili, Computer **36**, 47-56 (2003). DOI: 10.1109/mc.2003.1204375

[4] K. Schwaber, *Agile Project Management with Scrum* (Microsoft Press, Redmond, 2004).

[5] T.M. Connolly, *Database Systems: A Practical Approach to Design, Implementation and Management* (Addison-Wesley Longman, Boston, 2004)

[6] Liquibase, URL https://www.liquibase.com/ [accessed 12-Jul-2023]

[7] E. Evans, *Domain Driven Design: Tackling Complexity in the Heart of Software* (Addison-Wesley Professional, Boston, 2004)

[8] M. Feathers, *Working Effectively With Legacy Code* (Prentice Hall PTR, Upper Saddle River, 2004)

[9] PHPUnit The PHP Testing Framework, URL https://phpunit.de/ [accessed 12-Jul-2023].

[10] GitLab CI/CD, URL https://docs.gitlab.com/ee/ci/ [accessed 12-Jul-2023]