Easing the Control System Application Development for CMS Detector Control System with Automatic Production Environment Reproduction

I. Papakrivopoulos, G. Bakas, G. Tsipolitis (NTUA, Athens, Greece), P. Brummer, C. Deldicque, M. Dobson, J.R. Fulcher, D. Gigi, M. Gladki, F. Glege, J. Hegeman, F. Meijers, E. Meschi, K. Mor, L. Orsini, D. Rabady, K. Raychino, A. Racz, Rodriguez- Garcia, H. Sakulin, C. Schwick, P. Soursos, U. Suthakar, C. Vazquez-Velez, A. Zahid (CERN, Geneva, Switzerland), J. Branson, S. Cittolin, S. Morovic, A. Petrucci, M. Pieri, (UCSD, San Diego, California USA), D. Da Silva-Gomes, N. Doualot, A. Mecionis, R. K. Mommsen, V. O'Dell, M. Stankevicius, P. Zejdl, (FNAL, Chicago, Illinois, USA), G. L. Darlea, G. Gomez-Ceballos, C. Paus, (MIT, Cambridge, Massachussetts, USA), U. Behrens, W. Li, A. Stahl (Rice University, Houston, Texas, USA), D. Simelevicius (Vilnius University, Vilnius, Lithuania)

The CMS Detector Control System (DCS)

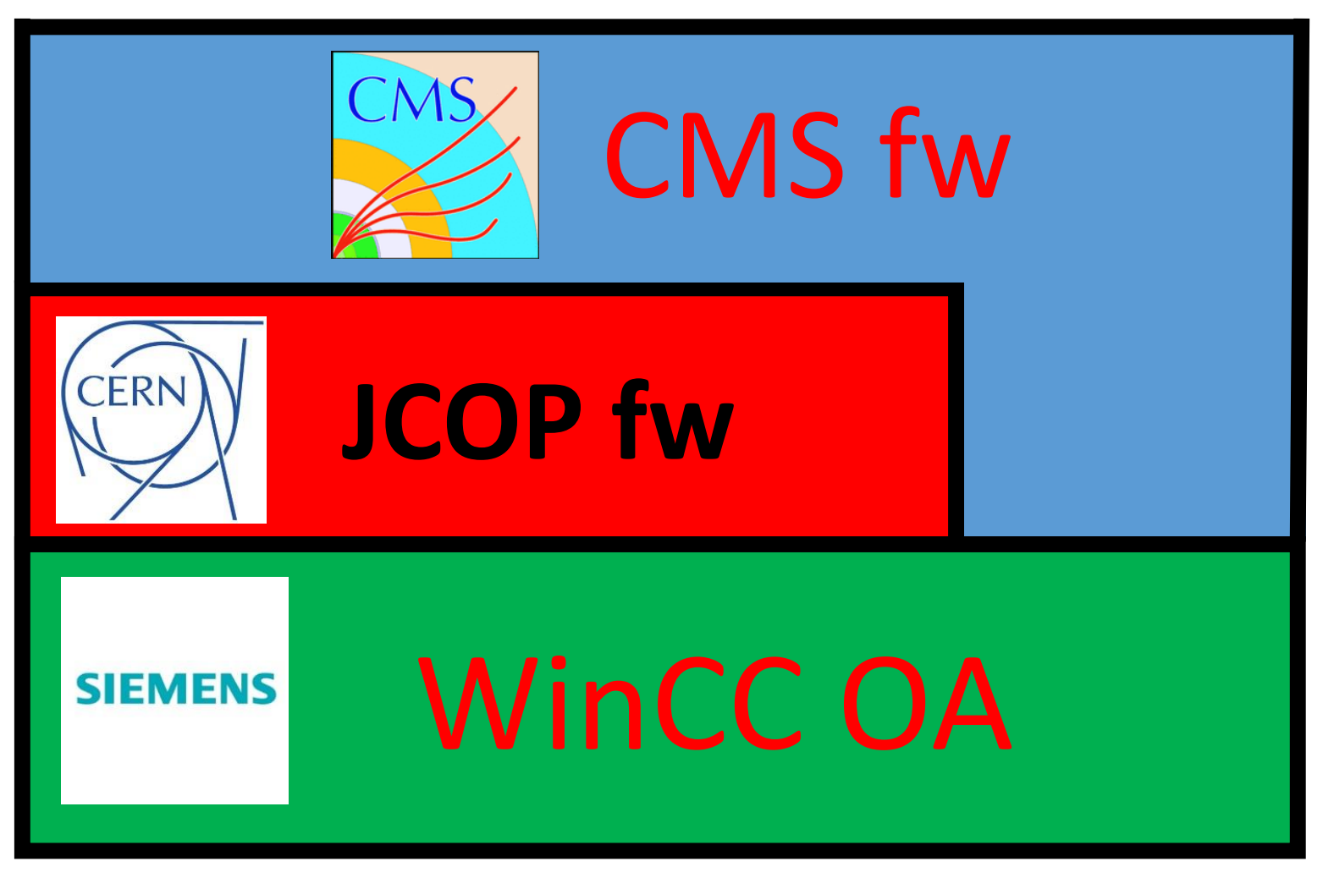

Figure 1: The software stack on the LHC Control systems

CMS, like all LHC experiments, is using WinCC Open Architecture (OA) and the JCOP framework to develop its control system. Additionally, it has created its own framework which consists of more than 200 individual components. The system is structured as a **distributed system** where each subsystem has different components installed depending on its role. This architecture implies that if a component is modified or upgraded, it needs to be reinstalled in all the corresponding systems. Consequently, this creates a system which is not dependent on the state of any of its subsystems and can be recreated at any point. To fully utilize this feature, all the components must be constantly in an install-ready state. This **modular architecture** allows for cleaner design and easier maintenance.

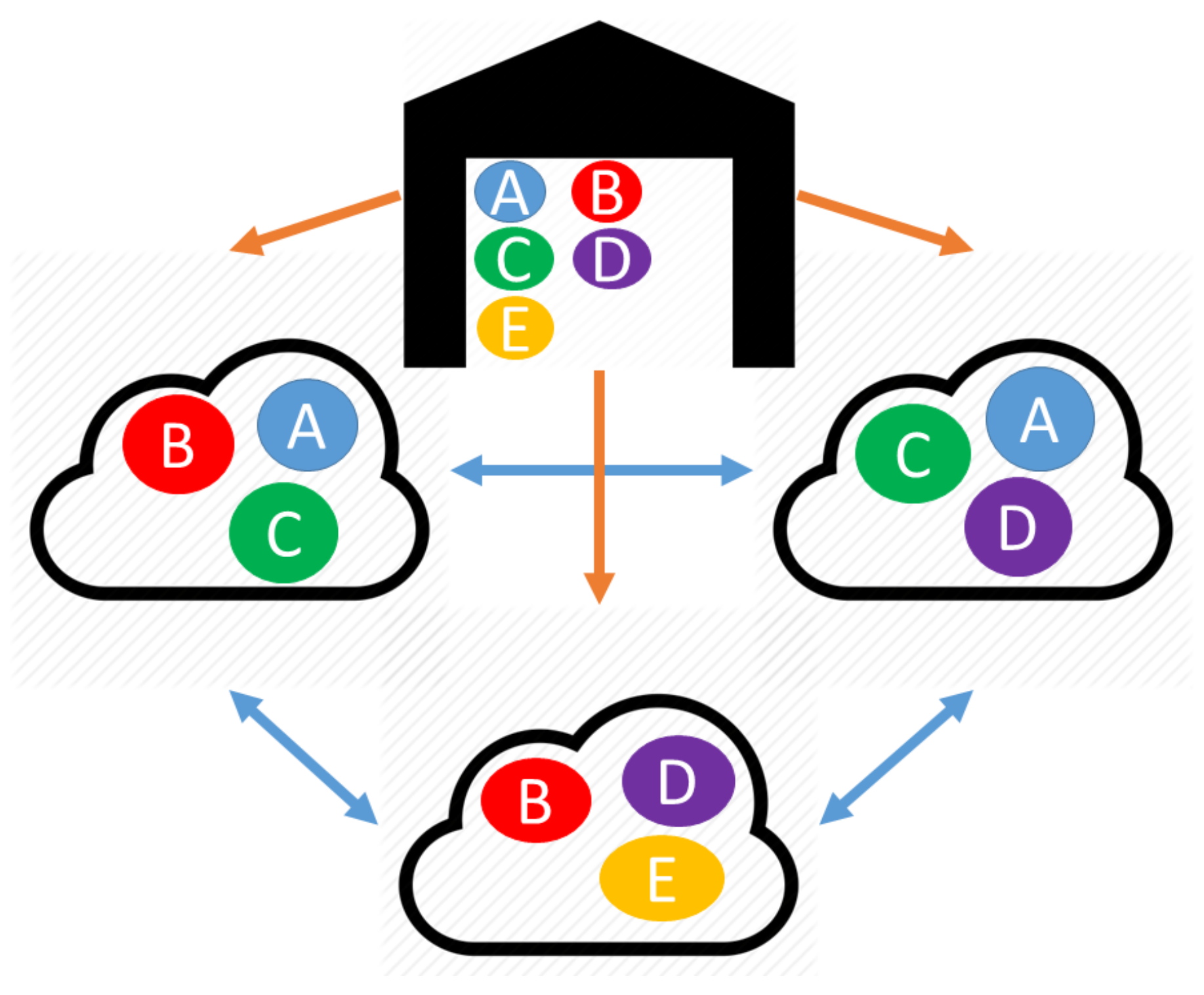

Figure 2: The modular architecture of CMS

Production System Administration

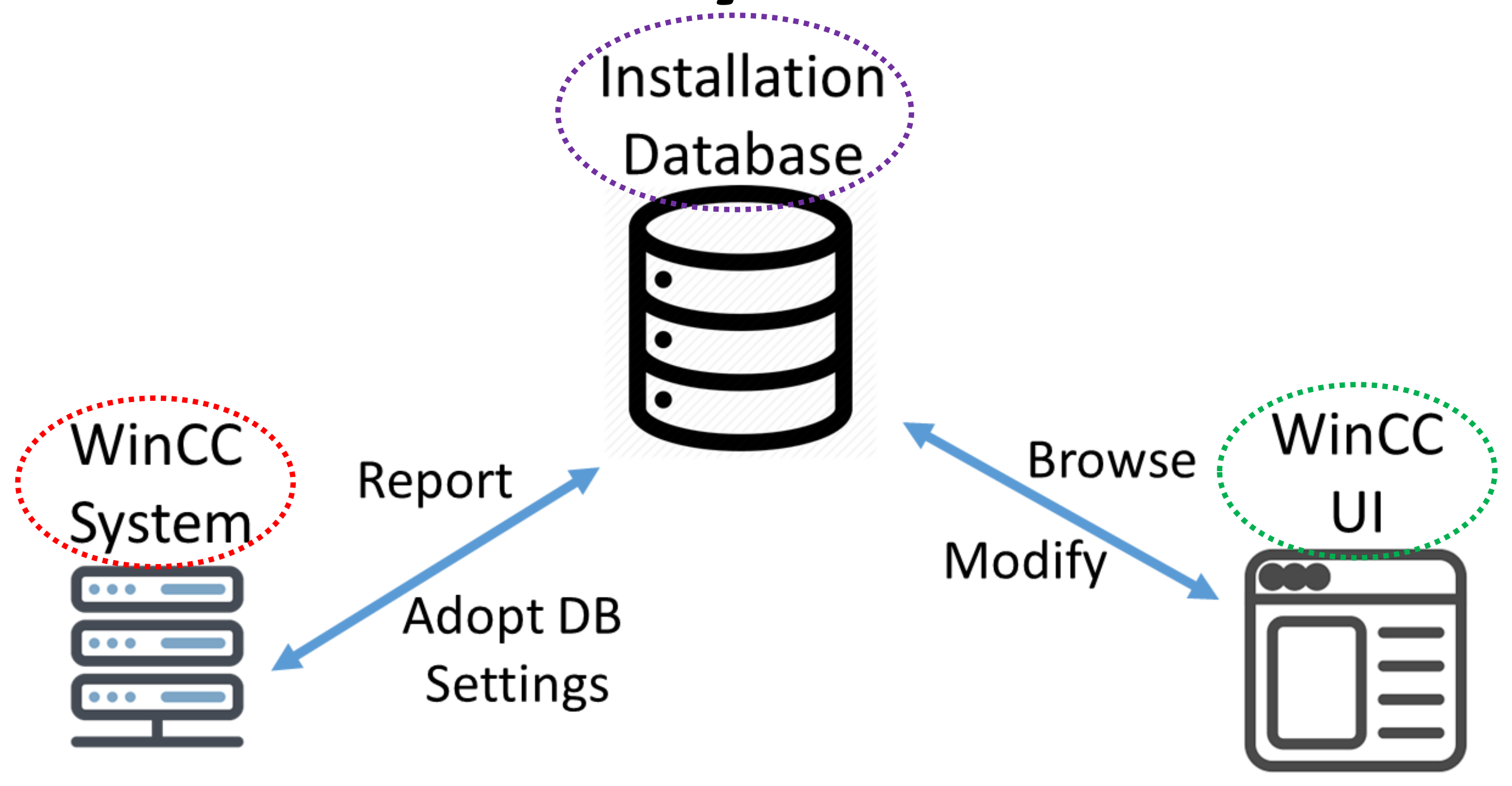

Figure 3: Schematic representation for the design of the tool that CMS central team uses to administer the projects.

All systems are handled **centrally** as they are created and maintained by the central team in terms of hardware infrastructure and control system projects. The components installed on each system are developed by the various subdetector groups in CMS. To administer the projects, the CMS central DCS team uses a tool developed by the JCOP framework team, which provides a database (**Installation DB**) schema that can store system information and the component configuration for each project. Each **system** reports its local configuration which is changed according to the database contents. The DB configuration can be modified through a **WinCC User Interface** (UI) that is part of the tool. Since the subdetector teams have **no direct access** to the system, a web interface is provided in order to modify the DB contents and as a result the corresponding system.

Production Environment Reproduction and more

The aforementioned tool has been extended in order to further meet the CMS needs. To ensure that there will be no disruption to the system, all the components have to be tested in a **production-like** environment before deployment. Such environments can be quickly created using the installation database contents.

In order to prevent production data from being modified in the process, a custom DB implementation was used (Development Database in Fig. 4).

- One empty schema (**SchemaB**) similar to production holding all the development data
- A second empty schema (**SchemaA**) with no tables but only with views

This feature enables us to copy data in order to implement a **"Create Like" mechanism**, a way to easily and effortlessly reproduce a production system in a development environment. This tool offers:

- Easier problem analysis, debugging and general support provision
- More efficient testing of the development process
- Faster software development
- Improvement of system stability

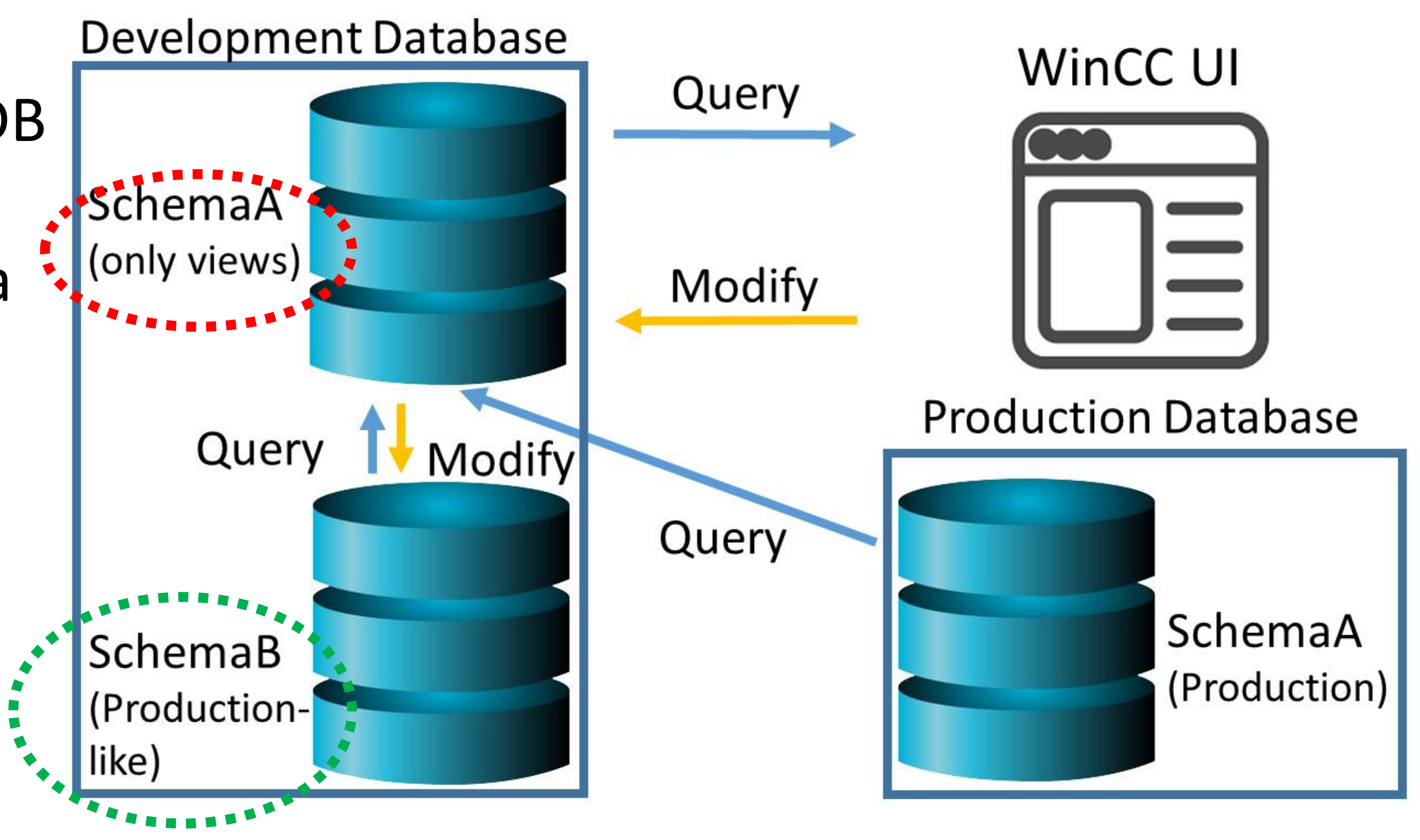

Figure 4: The database configuration used for the extension of the tool ensuring that production data stay protected.

Another newly added feature is the extension of the **component targeting mechanism**. The native targeting mechanism works by creating groups and adding components to them. This allows for: • quicker targeting • dependency handling • simultaneous component installation

During major software upgrades, where multiple components have to be updated into newer versions at once, some limitations appears:

- new groups have to be created as an existing group can not have different versions of the same component
- the synchronization mechanism can cause abnormal behaviour and component deletion has to be disabled

After these conditions are met, the user un-targets the groups from the projects and targets the new ones.

The **solution** to these limitations is adding one more constraint in the component targeting that has the form of a **tag**. This way :

- only the components that have a matching tag with the project are targeted
- we can just update the group containing the newly tagged components and then update the corresponding project's tag, without the need to stop or un-target anything

Finally, since the tool has greatly eased the development lifecycle of the central team, it is publicly available in the CMS community. To ensure an uneventful transition, the central team created an **ownership** mechanism that enables users to only modify things that they have create in the database and each team to work in an isolated environment.