CENTRALIZED SOFTWARE AND HARDWARE CONFIGURATION TOOL FOR LARGE AND SMALL EXPERIMENTAL PHYSICS FACILITIES

Alexander Makeev, Pavel Cheblakov, Nikita Atuchin,
Dmitry Bolkhovityanov, Sergey Karnaev, BINP, Russia

Abstract

The software configuration tool that combines the advantages of *centralized* and *decentralized* approaches has been created. It has been achieved by application of different configuration storage types for two types of clients: users and developers. The tool provides availability of all configuration information in one place, light connection to any configurable software applications and a single set of tools for searching and modifying configuration information. Light debugging of software applications unrelated to the main system configuration storage and creation of any configuration backup can be achieved by the tool.

INTRODUCTION

All software of control system, starting from hardware drivers and up to user space PC applications, needs configuration information in order to work properly. This information includes such parameters as channels calibrations, network addresses, server responsibilities and so on. Each software subsystem requires a part of configuration parameters, but storing them separately from whole configuration will cause usability and reliability issues. On the other hand, storing all configurations in one centralized database will decrease software development speed by adding extra central database querying.

In the present paper different configuration approaches will be considered, and the approaches will be compared. Generally, there are two opposite approaches that could be named as *centralized* and *decentralized*.

CENTRALIZED AND DECENTRALIZED APPROACHES

Decentralized approaches use local configuration storages, like configuration files. Such approaches allow users to configure software by most simple and suitable way, but cause difficulties during configuring whole software complex.

On the other side, *centralized* approaches use lonely configuration storage like relational databases. Such approaches allow users to configure software complex from a single place making configuration process simpler and faster. Disadvantage of such approaches is slowing down software development speed, which is caused by extra database querying from software and fast database scheme complication. Example of such approach is unified configuration system for physical equipment and software [1].

Users need the tools to search, view and edit that would be the same for the entire configuration of the complex. All listing functionality refers to the advantages of centralized approaches. Developers need to be able to test the operation of software applications on the local storage configuration to use the most suitable local storage in each case. All of these requirements can be realized by decentralized approaches.

Short comparison of *centralized* and *decentralized* approaches is given in Table 1. The comparison is produced using two main criteria: configuration usability (software users perspective) and configuration flexibility (software developers purpose). Term “data” will mean the variety of all configurable software parameters.

The configuration tool of *centralized* and *decentralized* configurations and their storage of automated synchronization approaches are proposed below. This configuration tool allows one to combine the advantages of both approaches without most of their disadvantages.

Also, there are a lot of in-between configuration approaches between *centralized* and *decentralized* approaches. Such approaches aim to combine flexibility of *decentralized* approaches and usability of *centralized* ones. For example, it could be a file system directory, placed on one network PC and shared to all PCs that run configurable software. The limitation of such combination is incompatibility of some approach qualities. For example, data querying autonomy and data centralization, optimal data format in each case and universal configuration tools may be considered. So, combinational approaches could not be constructed without disadvantages.

ISBN 978-3-95450-139-7

Table 1: Comparison of C*entralized* and *Decentralized* Approaches

Criteria	Centralized approaches	Decentralized approaches
Configuration usability (user-side)		
Data storage access	Simple, admin panel	Complex, storage seeking
Data seeking and manipulation	Simple, querying storage	Complex, different formats
Data access securing	Simple, user/password	Complex, multiple storages
Users activities logging	Simple, one control point	Complex, multiple storages
Whole data observation	Simple, one storage	Complex, storages switching
Data reservation and recovering	Simple, full and by parts	Only by local parts
Data structure documentation	Complete, storage scheme	Complex, different parts with different structures
Configuration flexibility (software-side)		
Data storage connection	Complex, network	Simple, local
Data querying	Complex, storage scheme knowledge	Simple, optimal local storage format
Data querying autonomy	Bad, network querying	Good, all data stored locally
Legacy software configuration	Complex, creating new queries	Simple, each legacy software has its own legacy storage format
Debug configuring	Complex, separate server	Simple, local copy of storage
Data format optimization	Bad, storage scheme priority	Good, optimal data format priority

UNITED CONFIGURATIONS CONCEPT

The heart of this concept is a configuration tool that could combine all advantages of both *centralized* and *decentralized* approaches, without offering their disadvantages. Close example of the concept is generation of EPICS DB files from relational database [2].

For embodiment the concept into reality has been created a tool, which combines both implementations of *centralized* and *decentralized* approaches. Each implementation will have its own storage. The tool also will contain a set of import and export scripts that will automatically synchronize data between two storages.
The structure of the tool is shown in Fig. 1.

ISBN 978-3-95450-139-7

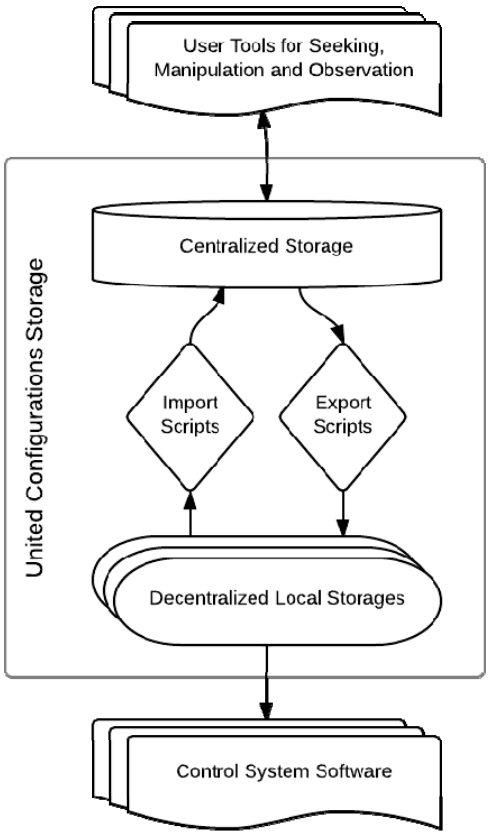

Figure 1: Structure of the PyCDB tool.

The tool reaches the concept purpose by offering two different types of storages for two types of clients: Users (User Tools), which positioned at the top, and Software Developers (Control System Software), which showed at the bottom. Centralized Storage, Decentralized Local Storages and its synchronization mechanism constitute United Configurations Storage, which located in the center. Clients communicate with different part of the storage: Users exchange information with Centralized Storage and Developers with Decentralized Local Storages. The concept offers two types of automated scripts: Import Scripts (which are moving data from local storages to centralized one) and Export Scripts (which are moving data from centralized storage to local ones). This structure makes it possible to use the advantages of both approaches and avoid their disadvantages.

Operators use tools to search, view and edit the configuration that work directly with the centralized storage. When the operator changes the configuration of the storage, it will automatically run scripts and data exports to the local configuration store, where they can directly use the application software system. If there is a need to import data from local storage, Developers and Users can use Import scripts. Thus, the United Configurations Storage allows Operators to directly manage the entire configuration of the software system as a whole, and technical experts - to work directly with local repositories of the most comfortable manner, in each case.

The price of configuration tool simultaneous flexibility and usability is its complication by synchronization processes. Triggering Import and Export scripts manually or by change events will synchronize two storages making two sets of data actual and consistent. When the centralized storage is changed necessary to verify all the scripts of imports and exports. Synchronization leads to the presence of copies of the data from centralized storage to the local storage (the need for reliable synchronization system).

The tool was named PyCDB(Python Configuration Database. PyCDB was written in Python, using the Django framework. Centralized Storage is a graph-based database [3]. Supported local storages are configuration files and relational databases.

The tool PyCDB (Python Configuration Database) has been developed by the concept. Now it is applied at BINP (Novosibirsk, Russia) and is used in configuring VEPP-2000 electron-positron collider (BINP, Russia), Electron Linear Induction Accelerator (Snezhinsk, Russia) and NSLS-II booster synchrotron (BNL, USA).

CONCLUSION

Despite the fact that this type of software developed for a long time, the present elaboration showed that there is still the potential for future development. The combining advantages of *centralized* and *decentralized* approaches were achieved due to a composite storage with a system of synchronization between the individual parts. Thus, this tool is an example of software that illustrates the possibility of using various innovative approaches.

REFERENCES

[1] Makeev A.V., Bolkhovityanov D.Yu., Karnaev S.E., Cheblakov P.B. 692-024 Unified Configuration System for Physical Equipment and Software.

[2] Keitel R., “Generating EPICS IOC Data Bases from a Relational Data Base– A Different Approach“, ICALEPCS’01, San Jose, November, 2001.

[3] *Robinson, I. and Webber, J. and Eifrem, E.* Graph Databases. — O'Reilly Media, Incorporated, 2013.

ISBN 978-3-95450-139-7