Dialogue expert system at command line interface – DES – CLI Ryahovetz

Iv Vasilev¹, N Nenkov²

¹University of Veliko Tarnovo “St.Cyril and St.Methodius”, Veliko Tarnovo, Bulgaria
²Konstantin Preslavsky University of Shumen, 115, Universitetska St, 9700 Shumen, Bulgaria

Abstract

The article describes the construction of a dialog expert system that supports the work of the system administrator. In its operation it uses the command line, which greatly improves its functionality and flexibility.

Keywords: Expert system, Command Line Interface - CLI, Data Hoard - DH, Logical Unit - LU, Dialogue Interface - DI

1 Introduction

Many commercial companies and existing open source communities compete in attracting more customers with a variety of innovative hardware and software developments. The race is to achieve simpler and easier to use devices and programs that guarantees them a larger share of the market. But despite the strong development of these technologies, we cannot ignore the fact that all these decorated, aesthetically appealing and easy to use interfaces obscure the existing functionality of the products. Knowledge of the system are encrypted in the database implicitly and facilitate the handling of the operating environment via the command line.

2 Exposition

The well known from the past command prompt, called CLI, reliably serves professionals in this field. For the inexperienced user it is difficult, but there are strong advantages for professionals compared to the graphic environments. It is used actively at server level, especially in UNIX and UNIX like platforms and even in the Windows server platforms in all business environments, banks and others. Practically the command environment is the foundation and not the GUI. In many server platforms, it is even impossible to launch GUI because the architecture does not allow it: namely “RISK Reduced instruction set computing”, which relies on productivity and security. They have not even written a graphical interface or if they have, it will strongly tend to the command view with the purpose of stability. Each layer adds extra risks to the security and potential crashes.

We can indicate a number of advantages of CLI such as: stable and fast interface for communication with the operating system; low consumption of system resources; full control over hardware resources and operating system; full control over processes; ability to carry out complex activities; fast communication, configuration and control from one environment to another; opportunity to easily automate routine tasks through scripting; full control over standard streams stdin, stdout, stderr; opportunity for easy routing and localization in logs formatting for subsequent filtering and monitoring.

• Architecture of the system

The system includes the following modules:

1. **DH – Data Hoard**. Module organizing the storage of input – output data and control of information. This module can be divided into sub-modules.

2. **LU – Logical Unit**. This module has the specific task to perform only and exclusively logical operations.

3. **DI – Dialogue Interface**. Dialog interface (fig. 1) for interconnection between end users and the system. This module has the main task to accept requests from the end user and returns the necessary information.

• Functioning of the system

Operation of the system is based on events occurring in the dialog interface DI or the SB segment for performing scheduled tasks. On a proper command it transfers to the main controller of the dialog interface "CSID". It turns to the main controller „LUIODI“ of the logical unit “LU” with the information submitted, almost unchanged.

The controller is duplex, because it works both ways, it can receive and transmit and determines where the request came from and what its purpose is. For example: an incoming request on "DI ➔ LU" and thus the data is passed on the simplex sub-controller LGR (logical unit receiving data).

The received data can trigger logical operations for
transforming and translating them in a convenient form for the next module "DH". After formatting the data if needed by more complex logical operations, the main controller "LUIODI" turns to "LEU" for their implementation.

Data, passed on this conveyor, is ready to be put on the next main controller "LUIODH", serving for connection to the module that stores data and determines the type of operation – extracting or entering information. If the event is data entry, it passes to the sub-controller LID for input data. For large amounts of data, several stacks can be formatted and submitted asynchronously for faster performance, otherwise they pass synchronously.

![Dialogue Interface Diagram](image)

FIGURE 1 Dialogue interface

For dependent data, the asynchronous method fails, and the synchronous-transaction method is used. Thus the prepared data are submitted for entry to the module "DH" and a record, change or deletion is performed. Modification of data can be direct or through "DB-LPO" if additional logical processing is needed. The need of this logic at the lowest level is for achieving faster performance when processing large volumes of information. This avoids unnecessary iterations with the "LU" logical segment.

At optimization, the most common requests are implemented in the section "MSS" and are invoked in matching events. The idea is one statement to be used repeatedly. For additional control or its reduction is used "COS" controlling optimization section of the module "DH", which creates, prohibits, permits or deletes a set of constants, triggers, indexes and other control-optimizing techniques within the selected database.

The logic controller is for primary processing of raw data, which transmits to the controller LUIODI, which in turn distributes them to the sub-controller LRD. In complex dialog templates, data is divided into stacks and returned asynchronously to the dialog interface and loaded at the locations indicated on the template. In simple consoles the data is returned synchronously.

The element carrying out scheduled implementation of routine tasks is SB – Scheduler Batches. It runs certain processes in background, such as calculation or routine data extraction for a period of time, updating the system.

### 3 Conclusion

The described system solves everyday tasks of administration and management of different platforms. The quality of the resulting advices depends on the knowledge entered into the system base. Further development would be in the direction of adding new modules and increasing the relevancy of its recommendations.

### Acknowledgments

This development was funded by Project RD-08-306/12.03.2015 “The business challenges of the new realities”, Shumen University “K. Preslavsky”, Bulgaria.

### References