Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 116 (2017) 318–325

Procedia Computer Science

www.elsevier.com/locate/procedia

2nd International Conference on Computer Science and Computational Intelligence 2017, ICCSCI 2017, 13-14 October 2017, Bali, Indonesia

Towards A Design for An Extendable Reporting Interface

Leo Andika*[a], Andrej Dyck[b], Horst Lichter[b]

[a]*King Mongkut's University of Technology North Bangkok, The Sirindhorn International Thai-German Graduate School of Engineering, Bangkok, Thailand*

[b]*RWTH Aachen University, Research Group Software Construction, Aachen, Germany*

Abstract

This paper proposes a design of an extendable reporting interface. The design of this interface is focused on how the data can be retrieved from a data processing software and how reports are generated to be consumed by another system. The reporting interface should be independent from its client software and from the report format that has to be produced. Therefore, there is a need for a very general data structure to handle various type of incoming data. We decided to use XML as a structure to create a report which later is consumed by another system. However, not all systems are able to process XML as an input. Therefore, the reporting interface should have the capability to produce several output formats. In addition, live reporting should also be supported to allow the user to retrieve information as soon as possible while the final report is being generated.

Peer-review under responsibility of the scientific committee of the 2nd International Conference on Computer Science and Computational Intelligence 2017.

Keywords: Live Reporting, Extendable Report Interface, Decoupled Report Interface, Data Processing Software, Clean Architecture

1. Introduction

A data processing software (DPS) performs several phases to process some input data in order to produce meaningful information as an output[1,2]. To create the output data, usually a report module is used. However, most of the report modules heavily depend on the DPS. In addition, the generated reports have also a predefined format[3] which is not always compatible to be consumed by other systems. However, the DPS is not always able to use different report modules to create needed specific output formats. Furthermore, if there are any needs to create a different report format, the used report module needs to be refactored. Obviously, this is not an effective process[4]. These problems lead to the following three research questions, addressed in this paper: What are the requirements on a extendable reporting interface? How does the architecture of such an extendable reporting interface looks like? How to use the extendable reporting interface to create various report formats?

E-mail addresses: leo.a-sse2015@tggs.kmutnb.ac.th (Leo Andika*)., andrej.dyck@swc.rwth-aachen.de (Andrej Dyck)., lichter@swc.rwth-aachen.de (Horst Lichter).

1877-0509
Peer-review under responsibility of the scientific committee of the 2nd International Conference on Computer Science and Computational Intelligence 2017.
10.1016/j.procs.2017.10.082

This paper implements a design for an extendable reporting interface (ExtRI) which aims to overcome those problems. This interface used by two clients. The first client of ExtRI is the DPS that provides the data to be processed. The second client of the ExtRI is software that performs the formatting and generates the reports.The ExtRI supports both linear as well as hierarchical data.

The paper is organized as follows. Section 2 presents the requirements of the ExtRI. Afterwards, section 3 presents the ExtRI architecture, then section 4 presents the implementation example of the ExtRI, followed by section 6 discussing the related work. Finally, conclusions and future work are presented in section 7.

2. Requirements

Based on the overall goals of the ExtRI, we can specify the following requirements on the ExtRI influencing its design and architecture:

1. It should be able to be used by any DPS, i.e. it has to be coupled as loose as possible
2. It should be able to report any data regardless of its structure
3. It should be able to produce report in multiple format
4. The generated reports should be consumable by any another software
5. It has to support live reporting. Live reporting is very useful to track the sequence of activities performed by the system or to give the immediate result to the client.

Every time a DPS needs to generate a report, the ExtRI retrieves the data and transforms it to a internal general data structure. To create the report, this data is subsequently formatted by one or more dedicated reporter plugins depending on the required format. The generated reports can later be consumed by other systems to be processed further.

Based on the requirements mentioned above, the ExtRI should be loosely coupled to the DPS that uses it. In order to process all kind of incoming data, the ExtRI should provide a general data structure to fulfill this requirement. In case new report formats are needed, the ExtRI should be able to use chosen plugin design to generate report in specific format.

3. Software Architecture

The Clean Architecture Principle introduced by Martin[5] is widely accepted and has been instantiated in different shapes such as hexagonal architecture, onion architecture, screaming architecture, etc.[6,7,8,9]. Therefore, the design of the ExtRI software architecture follows this principle.

It is not necessary for ExtRI to identify how the DPS is structured, causing the ExtRI become loosely coupled with the DPS. However, the client needs to adapt the classes and interfaces provided by ExtRI in order to use ExtRI functionalities. ExtRI provides three abstract classes and three interfaces which perform specific tasks. The design of the ExtRI is shown in figure 1.

The explanation of each components shown in figure 1 will be provided as follows:

3.1. ExtRI Data Structure

The DPS which using ExtRI might produce data in the linear or hierarchical structure. Therefore, ExtRI should be able to handle both structure types. This requirement is referred to the second requirement mentioned in section 2. Using string as a data structure for ExtRI was considered. However, string is not suitable for hierarchical data structure as the report data might have subgroup of data. Therefore, we applied a composite pattern to the data structure to allow ExtRI retrieves both linear and hierarchical data from the DPS. This data structure represented by three abstract classes, which are: *ReportData*, *ReportDataGroup*, and *ReportDataValue*.

ReportData class represents the top-most class in the hierarchy. This class contains the attributes and methods needed in order to use ExtRI. *ReportData* is extended by two other classes which are *ReportDataGroup* and *Report-*