IPFIX Mediation Framework of the SLAmeter Tool

Rastislav Kudla, Peter Feciľak, and Adrián Pekár

Department of Computers and Informatics
Technical University of Košice, Letná 9, 042 00 Košice, Slovakia

Abstract. Presented paper deals with the IP Flow Information Export (IPFIX) Mediation Problem. The problem was elaborated by IPFIX working group in RFC 5982 and RFC 6183. The aim of this work is to design and implement an IPFIX Mediation Framework of the SLAmeter monitoring tool based on these documents. The analytical part is devoted to the IPFIX protocol and the IPFIX Mediation Problem. Monitoring applications SLAmeter and BasicMeter are analysed as well. The core part of the paper is represented by the design and description of implementation workflow of IPFIX Mediation Framework. Accuracy of solutions is confirmed by experimental verification at the end of the paper. The main contribution of this work is to extend the possibilities of monitoring application SLAmeter.

Key words: IPFIX, Mediation, Mediator, Collector, Exporter, SLAmeter, BasicMeter, MONICA

1 Introduction

The trend of recent years in computer networking are converged networks which combine data, voice and video transmission into one common infrastructure. However, with increasing growth of opportunities raises demand for quality. In order to compare or even increase quality of networks and service, it is necessary to have the ability to measure network parameters.

SLAmeter is measuring architecture being developed by the MONICA research group in Computer Networks Laboratory at the Technical University of Košice. It is a tool for passive flow-based measurement and subsequent network traffic analysis with the purpose of determing the grade of network quality.

Flow-based measurement is a popular method for various network monitoring usages. The sharing of flow-based information for monitoring applications having different requirements raises some open issues in terms of measurement system scalability and capacity, flow-based measurement flexibility, and export reliability. *IP Flow Information Export (IPFIX) Mediation* may help resolve these issues.

2 Goals

The goal of this work is to design mechanism based on algebraic relations in evolution model. Based on semantic of pointcut designators (as they are defined in AspectJ) to identify the pointcuts in evolution model. The mechanism illustrate on simple example.

3 Analysis

A new sight on software development brings an aspect oriented programming [6, 4]. This sight mentions on the availability of evolution in the designing phase of software development. Based on semantic of pointcut designators (as they are defined in AspectJ [3]) can be in the data flow diagram of the designed system identified the pointcuts. In this pointcuts are inserted functions (events [2]). The selection of the functions into the pointcuts can be adapted according some requirements. In this work it is requirement on run time. It is time, which the system needed to reached the final state, moving from the beginning state.

To be able to know, if the designed system meets the run time requirement, the time evaluation for this system must be counted. The method, which is used for calculating starting and end - time of activities in software management, is critical path method. This method can be used for acyclic data flow diagrams. For this characteristic of this method it couldn't be used for counting the run time of the system. The data flow diagram of the system may contain a cycle. Because in the designing phase it is not clear how many times the cycle will be executed, only one execution of the cycle is considered. If the requirement is not accomplish for one execution, it could not be accomplish for more executions.

To solve the run time computation, two approaches were considered. The first one is based on the idea of the cycle compensation by one place (memory cell [5]) in data flow diagram (Fig. **??**). The disadvantage of this method is, that it will be difficult to write an algorithm to find and replace the cycles.

The second one is to count the run time by table. This is more easier, because it is established on simple computation of the table (see table 1).

Table 1. Table for graf b) form figure **??**.

Place	Preconditions	Time evaluation
a	-	0 s
b	a	10 s
f	b	20 s

4 Solution and Results

4.1 The design of deduction mechanism in evolution model

The ambition to achieve some evolution in the designing phase of software development is based on the idea that the selection of the functions (events, which should be used for the implementation of the system) can be adapted according some requirements. A deduction mechanism has been designed to reach this goal. This mechanism works with run time (time, which need the system to reach the final state, moving from the beginning state) requirement. This mechanism is inspired by biological evolution. From the generated variants of solution (designed by this mechanism) are selected only the ones, that accomplish the compliance with the requirement. From them is selected the best one. The mechanism is described in follow steps:

1. For the designed system a data flow diagram (desribing the flow of data in this system) is designed.
2. Then the file of functions (representsenting the events in the designed system) is created.
3. In the data flow diagram the pointcuts are identified.
4. Then the pointcuts are devided according the number of transfers, oriented into this pointcut.
5. The functions from input file are devided according the number of arguments, too.
6. A check must be run, to determine whether a compatible group of functions exists for each group of pointcuts. For example: for the group of pointcuts, where only one transfer is oriented inside, a group of functions with only one agument is considered compatible.
7. Number of members is counted for each group of pointcuts and group of functions.
8. All combinations (how can be functions inserted into compatible pointcuts) are generate for each group of pointcuts.
9. Accordingly the combinations variants of realization (of the designed system) are generated.
10. The max run time value is choosed. This time is the upper limit.
11. The begining marking is inserted.
12. According to the next algorithm the time evaluation for each variant is calculated.
    (a) The table for each variant is created.
    (b) The time evaluation calculation is started with table item, which all of its preconditions have been marked. If such item doesn't exist, the calculation is stoped.
    (c) In the next step, items, whose all preconditions have marking or some value are calculated. If such item doesn't exist, the calculation is stoped.
13. The generated variants are divided into the group of usable or unusable according to the results form step 12. The variant with the shortest run time is choosen, assuming this run time does not exceed the limit.

The figure **??** shows the example of this mechanism on the design of the system, which should realize a simple mathematical expression (1).

$$-\,(\,(\,a\,*\,b\,)\,-\,c\,) \tag{1}$$

5 Acknowledgment

This work is the result of the project implementation: Development of the Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

6 Conclusion

The goal of this work was to design mechanism established on algebraic relations in evolution model. Based on semantic of pointcut designators (as they are defined in AspectJ) to identify the pointcuts in evolution model.

The method provides some advantages. In the implementation phase it is clear, whether the specified requirements can be reached. Secondary, some partial automation may be brought into the software development process.

There are also some disadvantages consider. Some of the generated variants are not the realization of the system. With growing number of functions and pointcuts, the time needed for generating all variants grows. Therefore the application of this method is effective for small systems only. This disadvantages might be improved by the next research.

References

1. Anlauff Matthias, Pavlovic Dusko, Smith R. Douglas: Composition and Refinement of Evolving Specifications. In: Kestrel Institute, Palo Alto, California 94304, USA. , 26-29 Nov. 2001, pp. 157–165
2. Hudák Štefan: *Rozšírenia Petriho sietí.* Habilitačná práca. Vysoká šškola technická v Košiciach. Elektrotechnická fakulta. Košice, 1980. 107 p.
3. Kicyale Gregor, Hilsdale Erik, Hugunin Jim, Kersten Mik, Palm Jeffrey, Griswold G. William: An Overview of AspectJ. Department of Computer Science, University of British Columbia, Vancouver, 2005, 354 p.
4. Kollár Ján: Structural Proposition for Aspect Oriented Software Evolution. Proceedinds of 7-th International Scientific Conference ECI'2006, Košice - Herľany, Sep. 20-22, 2006, pp. 180–185
5. Kollár Ján, Novitzká Valerie: Semantical Equivalence of Process Functional and Imperative Programs. Acta Polytechnica Hungarica, Vol. 1, No. 2, 2004, pp. 113–124
6. Kollár Ján, Tóth Marcel: An Experiment with Aspect Programming Language. Proceedings of 3-rd Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence, Herľany, Slovakia, Jan 21-22, 2005, pp. 225–235