Chapter IV
On the Load Balancing of Business Intelligence Reporting Systems

Leszek Kotulski
AGH University of Science and Technology, Poland

Dariusz Dymek
Cracow University of Economics, Poland

ABSTRACT

The UML model consists of several types of diagrams representing different aspects of the modeled system. To assure the universality and flexibility, the UML involves only a few general rules about dependence among different types of diagrams. In consequence people can have the different methodologies based on the UML, but in the same time we haven't the formal tool for assure the vertical cohesion of created model. To test and reach the vertical cohesion of the model some auxiliary information about the relations among the elements belonging to different types of diagrams should be remembered. In this chapter the authors present the method of formal representation of such information in a form of the relation, called Accomplish Relation. This method is based only on the UML properties and is independent from any methodology. Additionally, they show how to use the UML timing diagrams for representing the users' requirements in association with use cases. To illustrate the usefulness of this approach we present how it can be used for load balancing of distributed system in case of a Reporting Systems based on Data Warehouse concept.

INTRODUCTION

In modern concepts of using IT in business organizations, one of the crucial elements are systems supporting business decision processes generally called Business Intelligence systems. This class of information systems includes data warehouses, OLAP systems, report generating systems etc.

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

Their complex structures reflect the multifaceted of modern business decision processes and the large scale of necessary information. The common feature of all mentioned kinds of systems is a large amount of data and a high computational complexity. Additionally, there are time limits[1] set on response time of these systems which result in high hardware requirements. On the second hand, some parts of these systems are not used all the time with full efficiency. Generally, BI applications generate several periodical cycles of a hardware nodes workload. The basic time cycles are relevant to periodical reports and adequate processes: we can distinguish daily, weekly, decadal and monthly cycles and a few longer cycles: quarterly, half-yearly and annual ones. Beside periodical processes we have also processes linked with everyday analytical tasks, which generate system workload, and must be taken into account.

Analyzing of the workload schedule for the whole system, based on aggregated time cycles, we must take into consideration the structure of the system. Usually, it consists of many single components: subsystems, software applications and hardware nodes. Considering the workload schedule for each hardware nodes we can indicate the situations in which one node is overloaded whereas other nodes are on low level of their efficiency. To assure optimal resource utilization, throughput, or response time we can increase the computing system power (by redundantion of some hardware components) or reschedule some processes. Such techniques, called load balancing, strongly depend on the software structure. So it seems to be useful to start considering the timing characteristic of the developed software from the software modeling phase. This situation forces formalization of this phase.

Unified Modeling Language (UML), being an uncontested modeling standard, in version 2.x offers 13 types of diagrams (Object Management Group, 2007a). In the load balancing context we are especially interested in ***timing*** diagrams introduced for describing timing properties of the modeled system. However, we suggest using them to describe timing characteristic of user requirements (represented at ***use case*** diagrams) and to trace their influence to other stages of the software modeling processes, represented by ***class***, ***object*** and ***deployment*** diagrams.

Let's note that UML as a tool became a base for some software development methodologies like RUP (IBM Rational Unified Process) or ICONIC (Rozenberg & Scott, 2001). It bases on such a fundamental concepts like an object-oriented paradigm or a distributed and parallel programming but is independent from those methodologies. This fact gives UML some advantages; especially it can be treated as a universal tool for many purposes. On the other hand, UML needs to be supplemented when we consider the vertical consistency of the model (Kuźniarz, Reggio, Sourrooille, & Huzar, 2002; Dymek & Kotulski, 2007a; Kotulski & Dymek, 2008), i.e. when we are interested in the formal description how one type of the UML diagrams influences on the model described by the other types of the UML diagrams. In the section below, the relational model, based on the graph theory, is proposed for describing the vertical consistency of the model.

Timing diagrams are one of many new artifacts introduced by second version of UML. They are the tool for describing the dynamical aspect of the modeled system and expressing the time characteristic of system components. The brief description of ***timing*** diagrams concept is presented in the following section. We also present the way of using the timing diagrams in cooperation with previously presented the relational model for obtaining the time characteristic for elements from different kinds of UML diagrams.

Successive section presents an example of using previously described models and methods, in case of the Reporting Data Mart based on the Data Warehouse concept. We describe how to use timing diagrams to obtain the time characteristic of system components, and how these characteristics

15 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/load-balancing-business-intelligence-reporting/30013

Related Content

Analysis of Existing Software Cognitive Complexity Measures
Sanjay Misra, Adewole Adewumi, Robertas Damaseviciusand Rytis Maskeliunas (2017). *International Journal of Secure Software Engineering (pp. 51-71).*
www.irma-international.org/article/analysis-of-existing-software-cognitive-complexity-measures/204524

A Method to Design a Software Process Architecture in a Multimodel Environment: An Overview
Mery Pesantes, Jorge Luis Risco Becerraand Cuauhtémoc Lemus (2018). *Application Development and Design: Concepts, Methodologies, Tools, and Applications (pp. 416-440).*
www.irma-international.org/chapter/a-method-to-design-a-software-process-architecture-in-a-multimodel-environment/188217

Design Space Exploration for Implementing a Software-Based Speculative Memory System
Kohei Fujisawa, Atsushi Nunome, Kiyoshi Shibayamaand Hiroaki Hirata (2018). *International Journal of Software Innovation (pp. 37-49).*
www.irma-international.org/article/design-space-exploration-for-implementing-a-software-based-speculative-memory-system/201484

Modeling Approach for Integration and Evolution of Information System Conceptualizations
Remigijus Gustas (2011). *International Journal of Information System Modeling and Design (pp. 45-73).*
www.irma-international.org/article/modeling-approach-integration-evolution-information/51578

Analyzing Human Factors for an Effective Information Security Management System
Reza Alavi, Shareeful Islam, Hamid Jahankhaniand Ameer Al-Nemrat (2013). *International Journal of Secure Software Engineering (pp. 50-74).*
www.irma-international.org/article/analyzing-human-factors-effective-information/76355