# Evaluation criteria for management of large communication networks

***Karel Richta[1], Ivan Vrana[2], Jan Vrána[3]***
***[1]Katedra softwarového inženýrství; FMF UK Praha, richta@ksi.mff.cuni.cz***
***[2]Katedra informačního inženýrství; PEF ČZU Praha, vrana@pef.czu.cz***
***[3]Komix s.r.o. Praha, vrana@komix.cz***

***Abstract**: This paper discusses aspects of how to compare suitability of different approaches for building and operating a system to support a telecommunication network configuration management. Proposed evaluation criteria will serve for comparison of main features and effectiveness of individual implementation methods considered for the network management. Applying these criteria can help to select such an implementation method, which enable reduction of errors, better exploitation of capacity of individual network elements and significant reduction of hardware costs.*

**Keywords:** Communication network management, computer aided support, implementation methods, comparison criteria.

## 1. Introduction

A telecommunication network is usually composed from a large number of communication nodes (switches) of various types. Each communication node has its complex internal structure. It varies from simple repeaters to sophisticated backbone communication nodes. Individual nodes are interconnected by links of various types and levels. Some of these links have a form of cables connecting individual devices and equipment. Other links have a less tangible form of mutually corresponding adjustments of some parameters of interconnected nodes. For example, mutually correctly set links, link-sets, circuits, circuit-groups, records in routing tables, etc. in a case of a GSM network. This arrangement directly follows from the structure of communication nodes.

Also much subtler associations can exist, which represent empiric observations and experience, how individual elements can indirectly affect each other. If, for example, a throughput of one node was reduced for some reason, then a throughput of entire network could be preserved by redirecting part of communication to other more suitable nodes. Some parts and parameters of the network can moreover have other than communication character. They can deal, for example, with records of users, tariffs for using network or with providing further superstructure functions and services.

Management of this kind of network consists of maintaining and adjusting individual parameters in individual communication nodes in such a way, that the network as a whole had required properties (e.g. a total

throughput, immunity to local dropouts, capability to provide additional services, etc.).

A typical administration tasks of management might be e.g. preparing the network to an expected (planned) higher (total or local) load by redirecting some parts of the load to the less loaded nodes, reconfiguring the network in case of drop-out of some nodes, incorporating new nodes or, on the contrary, removing unused or defective nodes from the network, changing a method of addressing terminal stations, changing tariffs for using the network, introducing new services, etc.

Each intervention to the network requires a very good knowledge of its technological construction and topology, but also a detailed knowledge of the actual setting of all parameters in communication nodes, which were affected by the intervention. Still before the intervention, operating staff should collect, read over, and put into mutual context information about a present state and then propose a new setting of individual parameters in order to achieve a required effect. A volume of information about a current state needed by an operating personnel rapidly increases with a size and complexity of the network. In the networks comparable by size with communication networks of any Czech phone operator, any larger change required intensive work of many people, who spent most of the time by revising reports about values of actual parameters of nodes and by tracking parameters affected by the change. This is a very time consuming, painstaking and non-effective work, which is also a frequent source of mistakes.

The idea is to use computers to support this demanding and non-effective work. Such computer support should allow network managers and administrators to quickly and effectively browse and examine individual parts of the network, their settings, their mutual relationships and to make all interventions easier and more effective. Besides visualization of network setting, it would be desirable to use a computer support also for automation of some repetitive tasks, for verifying impact of interventions and (as expert systems) also for proposing solution in critical situations. In order that the computers provide required information quickly and effectively, it is necessary to enter into the computer all necessary information about settings of entire network, in all its complexity of all parameters and their mutual relationships. This is one of the tasks, which are easy to formulate in a small scope, but computational and storage complexity of which quickly increases with a growing scope of the network. Effective importing and storing of information about the managed network and consequent processing of this information is a major and the most important problem of any computer supported management of a telecommunication network.

A telecommunication network configuration management can be considered as a general network management. Because most of commercial products from this category are proprietary ones from network

hardware vendors like Cisco, Siemens, Motorola, Alcatel, etc., information about governing principles is usually kept confidential. The possibility to find out principles utilized by these products to store and manage their data is unfortunately also very limited. Out of many potential approaches to store and manage data in implementing a system for management of large telecommunication networks let us consider the following three different implementation methods:

- General object-oriented approach
- Fixed data structures
- Dynamic relational data storing.

In the following paragraphs we shall outline basic principles and features of these implementation methods.

# 2. Implementation methods

## 2.1 General object-oriented approach (GOA)

In this implementation method a general object model is stored in a relational database. The main aim is generality and flexibility of the method. This implementation method is based on transformations of a domain model (communication nodes, their interconnections, parts of nodes, their parameters, mutual associations, concrete parameters of routing tables, their attributes, etc.) into a completely general model of objects and their associations. In this case a physical data model (with relational data storing) is simplified into two basic entities: Object and Association and two auxiliary entities: Object type and Association type, see Figure 1.

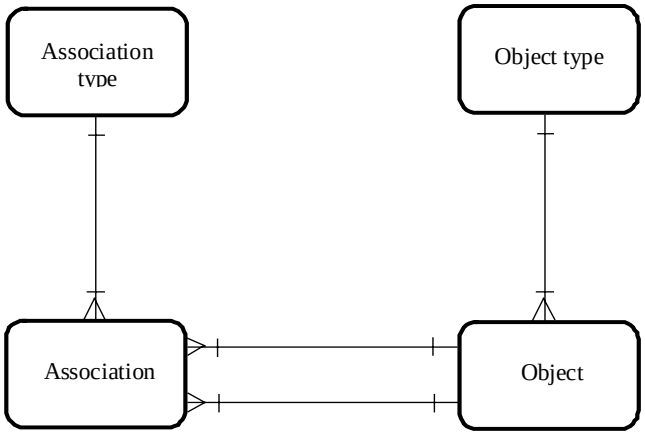


**Fig. 1 - Structure of relational database for a general object approach**

## 2.2 Fixed data structures (FDS)

In this implementation method a fixed-structure relational model is stored in a relational database. The main aim and objective of the FDS is to achieve maximum runtime efficiency of the created management application. It is like building a legacy information system when a considered system is described through a normalized E-R model by means of entities and relations. Entities are then stored in a host relational database. Functionality of the system is described by a set of algorithms which should process individual entities and their relations.

The FDS is characteristic by its high specificity with respect to the given application domain. This brings higher runtime efficiency at the expense of lower flexibility. Structure of the host application database is fully subject to the given application domain. It has character of mutually interlinked application-specific tables, as schematically depicted in the following picture.

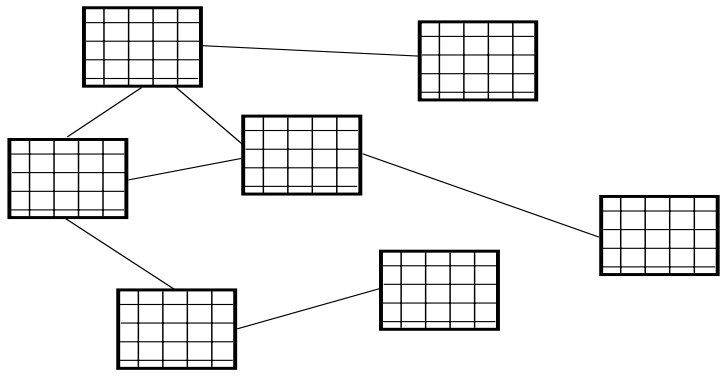

**Fig. 2 - Fixed data structure of application**

Transformation between the logical model, the structure of a host database and the structure of data management is relatively complex. As a result, in contrast from GOA, it is difficult or impossible to develop application logic separately from development of data management. Interface between both these parts of application has a form of a great number of very concrete functions. Thanks to this arrangement it is possible to individually optimize separate branches of data management and application logic and to achieve maximum performance and effectiveness. Extensive utilized potential of relational technology enables to solve selected types of operations effectively, see e.g. [1], [2].

## 2.3 Dynamic relational data storing (DRD)

In this implementation method a general object model is stored via dynamically constructed structures in a relational database. The aim of the DRD is to keep very high flexibility with respect to changes of an application domain while simultaneously preserving high runtime efficiency based on efficient utilization of a relational database engine.

The DRD is governed by the following principle: Data structure of the application domain is described by a logical model similarly as in the GOA. This logical model is transformed into a form, which is similar to a logical model of a data structure of the application domain used by the FDS method, i.e. into a system of entities and relations. The transformed logical model of a data structure of the application domain is stored in a host database in a form of metadata in a data space separated from a “payload” data and it becomes a key control element of behavior of the application, based on this method. A stored logical model then serves as a metadescription or a metaprogram (a control element) for most of business-logic algorithms. According to the metadata, a relational structure of specific tables and their relations are then dynamically generated, which in fact corresponds to the structure used by the FDS. Generated specific relational structures are then accessed through SQL queries which are dynamically composed according to the given request and to the metaprogram. Execution of such a dynamically built query is very similar to execution of query in the FDS, i.e. it effectively utilizes capabilities of the relational DB engine particularly for mass operations.

The DRD successfully combines majority of advantageous aspects of both extreme approaches: the GOA and the FDS methods in such a way, that “payload” data is always stored in a fixed “native” structure, where they can be efficiently processed, but at the same time, it is possible to change the structure of stored data and behavior of the application automatically, just by changing the metaprogram. This could be performed at any time and without any intervention to the program code.

During modeling of the application domain, first a general logical model of the application domain is created. Objects, attributes and associations are its basic building stones. This transformation of the general logical model of the application domain to its image converts an originally general object system into a relational system, thus into a system described by concepts of entity and relation. This transformation is therefore manageable only for application domains, which data side could also be described in entity and relation terms. This is possible in all or at least in majority of practical cases. Concrete methods of data description of application domain were published in numerous literature resources dealing with building "classical" information systems.

The principle of the storing data in DRD is depicted on Figure 3.

After transformation, which logically describes a data side of the given application domain, the general logical model of the application domain is stored into tables in a host database corresponding to an E-R schema in a left part of Fig 3. Thus a metadescription, i.e. metadata, is stored there, see e.g. [5]. Then, according to the metadescription stored in the static tables, further tables and association between them are dynamically generated in the host database. They serve for storing real data of the managed system, i.e. rows of entities. Tables in the host database after generating the dynamic data part are symbolically depicted in a right part of Fig. 3. This transformation takes advantage of tabular features of some sets of the logical model objects. These tabular sets are further transformed into dynamically generated tables in the host database, which are then accessed in a relational manner. This enables an increased efficiency of processing of stored data. In special cases, some tabular sets and their corresponding tables can degenerate to one-column tables, one-row tables or even to one-column-one-row tables. If there were many such tables in a dynamically generated part, it could partially degrade performance and efficiency of processing of data, stored in this way. But all other mechanisms would remain unchanged. Thanks to that, practically arbitrary application domain or its logical model could be generally described in this way.

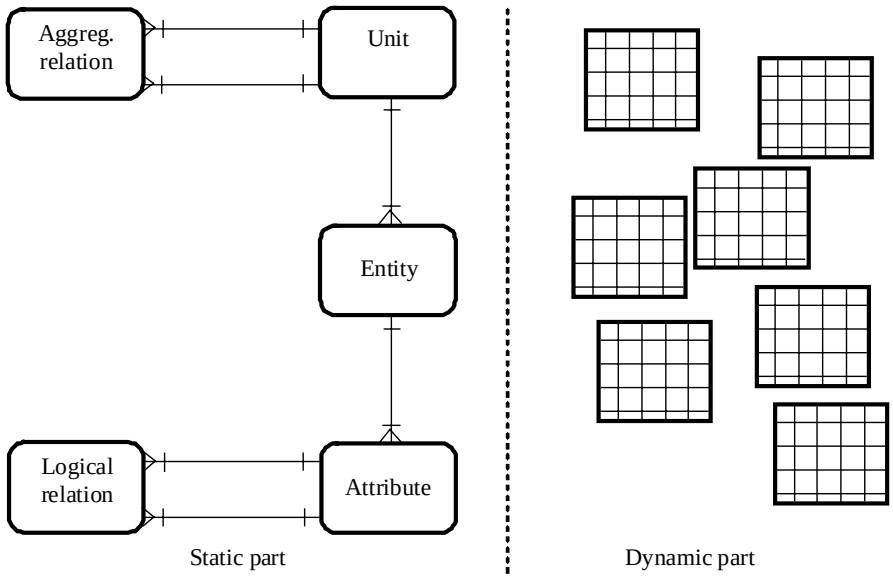


**Fig. 3 - DRD static and dynamic part**

A firm structure and general transformation algorithms thus remain identical for arbitrary application domain or for arbitrary modification of the given domain. Only a dynamically generated part varies, which is generated from a logical model (metadescription) of a given application domain and which corresponds to all specific requirements of this domain.

These approaches were described in detail in [3] and [4]. Objective of this paper is to show aspects of how to compare suitability of different approaches for building and running a system which supports a telecommunication network configuration management and also to compare main features and effectiveness of individual alternatives.

## 3. Evaluation criteria

In order to enable comparing different implementation methods, let us first determine evaluation criteria for this comparison. The main objective is to make development of applications for management of large data systems (for short we shall simply call them "applications"¨) easier and also to make management and enhancement of these applications easier. A crucial aspect in assessment of alternatives of applications and other commercial services is the effectiveness of a given alternative and its return of investments. All aspects are usually expressed in terms of the following factors:

- costs of development of initial functionality of the application,
- costs of maintenance and extension of (additional) functionality of the application,
- a runtime efficiency of the developed application.

The costs concept represents here the total demands for development of the examined application or its part. It need not be expressed in terms of money; it is rather some generalized rate of complexity, a rough labor demand, etc. of the examined part. To convert these generalized costs to a real economical value, several other aspects (i.e. a labor cost, an experience of members of the development team, etc.) should be considered.

Values of the above-mentioned three factors have a fundamental influence on an overall success of the created application. The first two factors represent a direct economic value of the development process and the third factor has a principal influence on a practical usability (or non-usability) of the created application. We shall further focus at assessment of the software of the application. Costs can be also assigned to individual parts of the application, e.g. business logic, user interface, etc. For simplicity, we shall separate the costs associated with storing and processing of data from costs associated with other parts of the development process and we shall consider only the total value of the data

management part throughout this paper. Now, we shall outline a spirit of the above mentioned assessment factors.

## 3.1 Costs of development of initial functionality– DIF

Costs of the initial development of the application functionality can be considered as total costs needed to create initial functionality of the prospective application. Initial functionality development costs consist from costs of individual project life-cycle and its phases: i.e. design/development, implementation and testing phases.

Costs of the **design phase** considerably depend on complexity of the problem domain itself, but also on used implementation technologies. These costs consist particularly from identification and design of all possible branches of application functionality and of a design of its logical algorithms. The character of the task, a necessary experience of designers, etc. can be expressed by a *coefficient of complexity*, which grows with increased complexity of the application. Concrete values of this coefficient usually follow from experience of a project manager.

Costs of the **implementation phase** could be quantified with regard to a program scope (e.g. a number of lines of a program code). But also here we should take into account a variable complexity of created program code with regard to use of different implementation methods.

Costs of the **testing phase** have a similar dependence on the used implementation method as had two preceding phases. To quantify costs of this phase we should consider e.g. a number of program branches, which should be checked and tested in individual alternatives. Again, a coefficient of complexity can be used to distinguish between different implementation methods.

Total costs of development of initial functionality can be obtained as a sum of costs of all activities during individual phases of the development lifecycle.

## 3.2 Maintenance and extension costs - MEC

After initial development, majority of real (information) systems goes through further development cycles. Their main purpose is extending applications' initial functionality and adapting it to changing external conditions. Costs of maintenance and expanding of any real system often considerably exceed initial functionality development costs. This fact should be considered since the very first moment of a lifecycle, when strategic decisions concerning used technologies were adopted. A utilized implementation technology can radically influence a labor consumption of the subsequent development cycles of this software task.

Design and elaboration of the solution is a very important phase for comparing costs and a labour input of different implementation methods. Costs and a labour input of this phase distinctively depend on the selected implementation method and on character of intended changes and on extension of functionality. Small changes of functionality have rather an *additive character*, i.e. it is merely adding a new functionality without changing the existing one. Then dependence of costs on selected implementation method need not be strong. But in the case when proposed changes of functionality might have stronger influence onto existing functionality or they have a *multiplicative character* (when a small change of a functionality caused a radical growth of complexity), then costs and a labour input is strongly dependent on the used implementation method. A relative size of areas influenced by some change with respect to a total size of an application and a coefficient of complexity can serve as possible criteria for the quantitative assessment.

Costs and a labour input for implementation of the above mentioned changes depend directly on the implementation method. In a non-appropriately chosen implementation method, each change of functionality of the whole application can cause changes in functionality of the data management part. But in other cases, majority of later modifications need not manifest themselves to the data management part. Costs and a labour input for debugging and testing of a new functionality is proportional to the size and character of changes, which happened in the previous step.

Again, total costs of a maintenance and extension of application functionality are a sum of costs of individual phases of the lifecycle. As a rule, later phases have the main contribution to the total costs.

Regardless of the implementation method, complexity of development initial functionality could be generally expressed as a function

$$C = f(S, \rho)$$

where $S$ is a scope of application domain and $\rho$ is a density of application algorithms, which follows from an implementation method. Value of C monotonically depends on a scope of application domain for a given density of application algorithms. Dependence of C on a density of application algorithms has two components: the gross time consumption and a coefficient of time consumption. With decreasing density $\rho$ the gross time consumption grows whilst its coefficient decreases. An overall time consumption is a product of both components.

Besides the scope of an application domain, complexity of maintenance and extension depends also on a number and character of original algorithms to be modified and also on character of a proposed change.

### 3.3 Runtime efficiency

Previous two paragraphs dealt with economic aspects and measures, which should be taken into account in considerations concerning an implementation method used for creation of the application. Also the aspect depicted in this paragraph has its economic dimension, but its principal dimension is the usability or non-usability of created application in real operational conditions.

When thinking about runtime efficiency of created application, or of its data storing and management, respectively, it is necessary to examine requirements laid by each implementation method at the storing of the expected volume of input data. Depending on the way of data storing used by a particular implementation method, it is then necessary to examine a principal complexity of the most often used operations, which would be performed on the data (single selections, mass selections, comparison, modification, etc.). When considering capacity and computational complexity of the compared methods, it is further necessary to take into account not only requirements on data storing, on a processing and on a volume and complexity considered in the beginning of the design, but also dependence of mentioned features on a possible increase of the volume and complexity of input data. Underestimation of the runtime efficiency might bring considerable troubles during future real use of the developed application and can also degrade invested resources and means.

## 4. Conclusions

The usefulness of the described evaluation criteria was examined in the case study of building a real-life system for management a GSM network of the Eurotel Czech (O2) mobile phone operator. They made possible to select the most adequate implementation method for management of this complex network, which led to dramatic speedup of most management activities, dramatic reduction of errors introduced into the process of implementing changes, improved exploitation of capacity of network elements and significant reduction of hardware costs. Results of that case study will be published in the subsequent paper.

A great number of real problems of management of large and complex systems can be transformed to the abstract case of network management. The method, which enabled to effectively solve the task of a network management, could then achieve a broad utilization for solution of many real problems.

## Acknowledgement

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic (Grant No. MSM 604070904 – Information and knowledge support of strategic management and Grant No. 2C06004 – Intelligent instruments for assessment of relevance of content of data and knowledge resources).

## References

[1] Biskup, J.: *Achievements of Relational Database Schema Design Theory Revisited*. In: Semantics in Databases, Lecture Notes in Computer Science, vol. 1358, pp. 29-54. Springer-Verlag, Berlin, 1998.

[2] Von Halle. B.: *Uncovering Business Rules*. Database Programming and Design, 8(7): 13-18, December 1995.

[3] Vrána, J.: *Dynamic relational data storing method for management of large data systems.* Doctoral dissertation. Czech University of Technology in Prague, 2003.

[4] Vrana, I., Vrána, J.: *Effective method for management of large data systems*. SCI 2004, Orgando, ISBN 980-6560-13-2, pp: 143-147.

[5] Richta, K., Valenta, M.: *Using Metadata to Information Transfer* (Paper in Conference Proceedings) In: Proceedings of Workshop 2002. Prague : CTU, 2002, vol. A, p. 182-183. ISBN 80-01-02511-X.