МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2007
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2007
Proceedings of the Thirty Sixth Spring Conference of
the Union of Bulgarian Mathematicians
St. Konstantin & Elena resort, Varna, April 2–6, 2007

ACCELERATION OF STRUCTURED AND HETEROGENEOUS CONFIGURATION OF THE APPLICATIONS[*]

Alexander P. Penev, Dimcho S. Dimov, Dobromir P. Kralchev

This paper treats some aspects of configuring applications, libraries, and object classes. We are proposing heterogeneous and structured approach, in order to define parameters of the systems, based on hierarchies of attribute-value (pairs) with additional metadata. This approach uses multiple levels and multiple formats of the configuration sources (files, databases, etc.). We are proposing a method for caching configuration sources, which are security insensitive, in order to accelerate them.

1. Introduction. The present-day applications more and more often pay attention to contrivances for settings from system administrators (super users) and users. Therefore, usually one of the basic subsystem of the core of every application is a configuration subsystem.

Configuring is a process of setting some elements – parameters of the system (functions, methods, objects, classes, modules, libraries, subsystem, dynamic loaded modules, etc.).

There are many [2–13] approaches for configuring applications and libraries. They have pros and cons. Usually every library, framework, or application has its own approach to configure and its own file format for saving configuration. The basic differences among them are configuration file format, number and organization of the files, as well as their location in the system.

For the needs of the developed by the authors applied system, OpenF [1], is used an approach, which is based on the below described characteristics of the methods for configuring. This approach is open for integration of almost all aspects of the other systems. In addition, a configuration files hierarchy and metadata are used. A caching system is applied and it uses the specific facilities of .NET Framework and C# (the approach is applicable for other frameworks and languages), typical for the cache mechanisms of some Web-based systems [13], developed with interpreted (script) programming languages.

2. Approaches for configuring. The different configuration file formats presume different usage method of configuration, as well as different flexibility, performance, and protection of the parameters from unauthorized (restricted) modifications from the users. The following configuration (file) formats are widespread:

[*]**2000 Mathematics Subject Classification:** 68U01, 68N19

Key words: Configuration, Hierarchy, Cache, INI, XML

- “INI“ and “conf” text files [5];
- Windows Registry hierarchical database [6];
- XML files [2, 3, 10, 11];
- YAML files [4, 13];
- Binary files with a specific for the application format;
- Source code containing constants (parameters) [13], etc.

Sources of configuration data could be not only files, but also databases, environment variables, command line parameters, etc. Therefore, we will call them by the name Configuration sources.

The logic of configuring can be:

- Constants – hard coded constants in the application source code. It requires module or application recompilation after each change of the parameters (constants);
- Configuration file – each element of the application reads (loads) parameters from a file. This may multiply one and the same processing logic of the configuration file to many places;
- Forced configuring – configuration system loads the parameters and assigns variables or fields of the objects, which must be configured. Disadvantage of this approach is that it configures all parameters independently – without paying attention to their future use;
- Configurator – an object in the system that loads parameters when it is necessary and replies to the queries of an object or a function having parameters for settings. It can implement the logic of merge of more than one source (file), priority, security, caching, on demand loading, etc.

The parameters organization can be:

- Flat – usually organized as an attribute-value list;
- Group (hierarchy on two levels) – the attributes’ names are grouped by some criterion;
- Hierarchy – the attributes’ names are hierarchically organised and grouped by some criterion;
- Graph – usually hierarchy with internal or external (or both) hyperlinks that refer to the values of the other attribute or attributes. In this case, we may have inheritance of attributes, profiles, etc.

The configuration sources (homogeneous or heterogeneous) organization can be:

- One or more sources on one level – usually these are sources containing settings for all users. This organization is used often at simple applications;
- One or more sources on two levels – usually these are sources, containing settings for the whole system and for the each user separately. This is the most often met case;
- Sources organized on several levels – for example for a user, for a whole system, for a local network as the values in them are searched sequentially from the more specific to some more general sources. Thus, the missing parameter in a configuration “inherits” its values from the more general configuration, etc.

The rights to change parameters values (also depends on protection of configuration sources – files, databases, etc.) could be: without restrictions; from a user; only from the system administrator; from a developer; etc.

3. Structured heterogeneous configuration model. Fig. 1 shows the main idea (scheme) for the heterogeneous hierarchical structure of a system configuration. In order to work, to find and to select the exact set of parameters a description of configuration structure (meta-configuration) is needed. This description must contain: levels structure; where places, values are stored; types of attribute-values; rights, inheritance and visibility of the settings; other necessary information for the application.

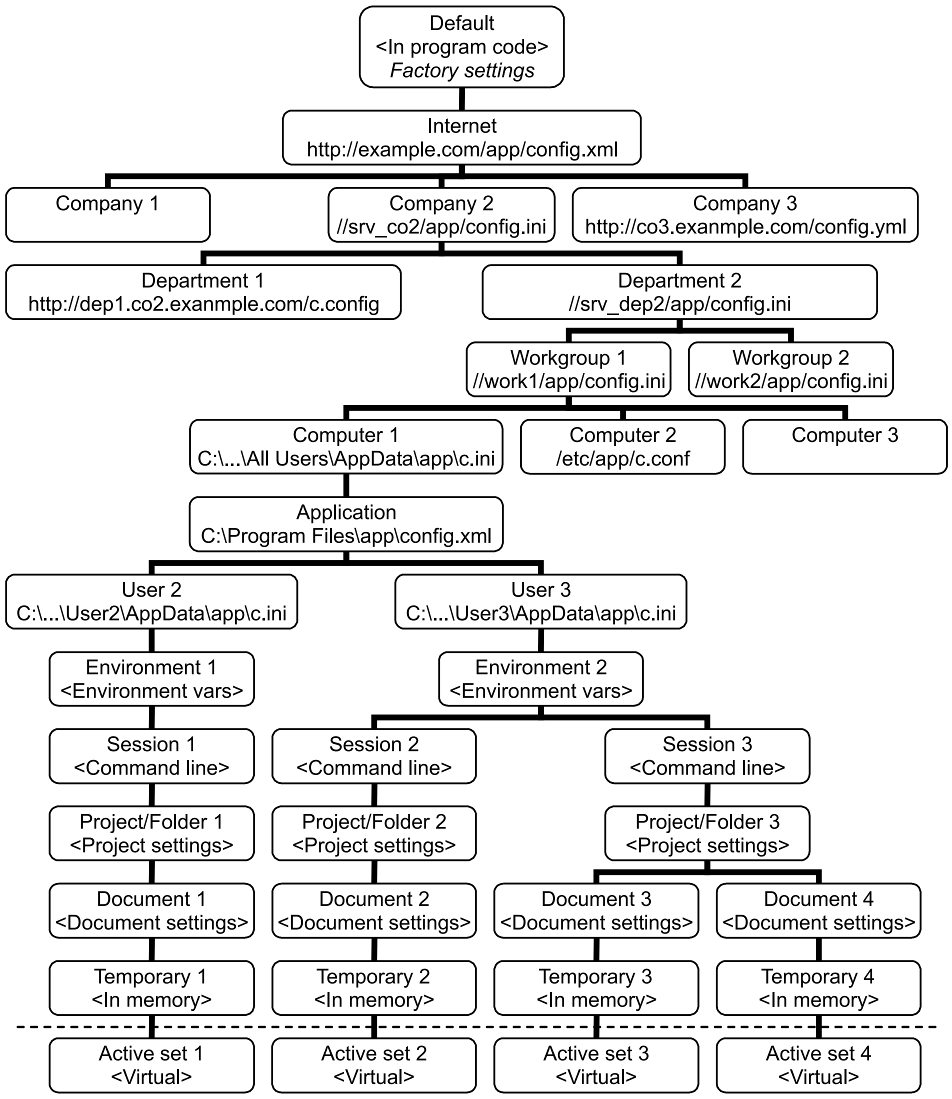

Fig. 1. Structured and heterogeneous configuration sample

The configuration system gives the application a model, covering the possibilities (capability) of different formats, as well as methods for defining the parameters from different system users. At the same time, configuration subsystem must remain opened for an extension with specific requirements of an application. For example, with meta-configuration we can include a new level, if a company is a holding company or if a document has subsections with different configuration. In addition, if we have Web application folders and subfolders, having their own configurations, we can make them inherit each other on unlimited levels.

The levels shown on the figure are just an example as only levels “Default” and “Active set” are necessary. An “Active set” is a virtual merge of all above levels form the leaves to the root. They are created only when it is necessary and only in these parts, which the application uses.

Each level can have different storage format of keys (attributes) and values of configuration. The keys are also organized hierarchically (in namespaces).

4. Configuration sources and channels. Depending on the needs of the system configuration can be stored in different places and in different formats – file, database, document, memory, etc.

System configuration could be separated and distributed to more than one place (these places should not be of the same type). For example, XML file with the settings for a whole company, would be stored on the main company server and one file for each user would be stored on the local workstation (or in their profiles in YAML or INI format). The separated parts of the configuration can be merged or inherited each other, depending on the meta-configuration.

Therefore, in the system two types of objects (interfaces) are defined: configuration sources, and channels. The sources are responsible for the access to a data format (for example INI, XML, and YAML). The channels are responsible for the access to data locations (for example files, databases, and HTTP resources). In addition to them, other basic classes are Configuration and Configurator.

5. Security. Usually user changes values of parameters when configuring the application. This is not always desirable (or it is not desirable for the part of the parameters). Therefore, in the systems some additional information for rights, defining what exactly the user could change or see, must be available. For example, general for the whole company configuration files (parameters) are changed only by a system administrator, while the user settings could be changed by users and administrators.

It is recommended to divide the configuration parts with different rights in separated files (places). These files are protected by the operating system (access rights system). The rights (as part of meta-configuration) must be stored separately. Usually access to the meta-configuration is enough to be read-only (after the ending the system development). This is one of the reasons to store the meta-configuration separately from the configuration.

6. Configuration caching. Caching the configurations has three main goals: usability of an application without network; network traffic decrease; speed of configurator acceleration. Caching is applicable on each configuration level (we cache higher hierarchy levels). Caching is made only for those parts of configuration described in the meta-configuration, which are security insensitive. In other cases, we require a network connection (to configuration source) to insure configuration authenticity. It is necessary to fetch data from a configuration source.

Other type caching is made in the memory for faster access to already received configuration data. A basic task of this cache is to achieve a better application performance (keys are hashed).

Another idea used is that each configuration source has been read; translated to C# code; and finally compiled (using built-in C# compiler in a .NET framework). Result assembly is stored (cached) and may be loaded dynamically and used instead of the configuration source, while it is unchanged. This saves multiple analyse of the configuration sources in every run of the application (usually it is done even if configurations are not changed and it could be time consuming). Table 1 shows an example of a configuration, transformed into C# source code. The first column shows the INI file with two sections. The second column shows equal to the first column – XML configuration file. The third column contains the generated corresponding C# source code (note: a source code is simplified).

INI file	***XML file***	***Generated Assembly Source***
`[A]`	`<?xml version=”1.0”>`	`using System;`
`Key1 = abc`	`<config>`	`using System.Reflection;`
`Key2 = 10`	` <section name=”A”>`	`using OpenF.Core.Config;`
	`  <key name=”Key1”>abc</key>`	`[assembly:AssemblyVersion("1.0")]`
`[B]`	`  <key name=”Key2”>10</key>`	`public class CachedConfig: Config`
`Key1 = 1`	` </section>`	`{ public CachedConfig() {`
`;comment`	` <section name=”B”>`	` add(”/A/Key1”, ”abc”);`
	`  <key name=”Key1” val=”1” />`	` add(”/A/Key2”, 10);`
	` </section>`	` add(”/B/Key1”, true);`
	` </config>`	`}}`

Table 1. Auto generated C# source example

7. Conclusion. The proposed model for structured and heterogeneous configuration has indisputably advantages over the all other listed approaches. It gives possibility for easy and flexible adaptation to the needs of any system. The hierarchy contributes for a better and an easier implementation of parameters security. This approach is opened for an easy extension with other configuration source types (connections and file formats). The proposed method for caching of the configuration parameters contributes to accelerated processing of configuration files and leads to a better application performance.

REFERENCES

[1] A. Penev, D. Dimov, D. Kralchev. Open Hybrid System for Geometrical Modelling. 17th International conference SAER-2003, vol. **1**, 2003, 131–135.

[2] R. Lhotka. Application Configuration Files Explained
`http://msdn.microsoft.com/library/en-us/dnadvnet/html/vbnet04222003.asp`, 2003.

[3] Microsoft Corporation, ASP.NET Configuration Overview
`http://msdn2.microsoft.com/en-gb/library/ms178683.aspx`, 2006.

[4] O. Ben-Kiki, C. Evans, B. Ingerson. YAML Ain't Markup Language (YAML) Version 1.1, `http://www.yaml.org/spec/`, 2004.

[5] INI File – `http://www.cloanto.com/specs/ini.html`

[6] J. Honeycutt. Microsoft Windows Registry Guide, Second Edition, Redmond, Microsoft Press, 2005, 608 pp., ISBN 0735622183.
[7] LibConfig Configuration Library – `http://www.hyperrealm.com/libconfig/libconfig.pdf`
[8] Zend Framework Config – `http://framework.zend.com/manual/en/zend.config.html`
[9] Elektra Project (LibElektra) – `http://www.libelektra.org/`
[10] NINI .NET Configuration Library – `http://nini.sourceforge.net/`
[11] Carbon Component Framework – `http://carbon.sourceforge.net/`
[12] UniConf – `http://open.nit.ca/wiki/attachments/uniconf.pdf`
[13] Symfony Project – `http://www.symfony-project.com/`

Alexander Plamenov Penev
Dimcho Stojkov Dimov
University of Plovdiv, FMI
236 Bulgaria Blvd.
4000 Plovdiv, Bulgaria
e-mail: apenev@pu.acad.bg
dimcho_dimov@abv.bg

Dobromir Pavlov Kralchev
UHT – Plovdiv, Faculty of economic
4000 Plovdiv, Bulgaria
e-mail: dobromir_kralchev@abv.bg

УСКОРЯВАНЕ НА СТРУКТУРНАТА И НЕХОМОГЕННА КОНФИГУРАЦИЯ НА ПРИЛОЖЕНИЯТА

Александър П. Пенев, Димчо С. Димов, Добромир П. Кралчев

Обсъждат се някои аспекти на конфигурирането на приложения, библиотеки и обектни класове. Предлага се нехомогенен структурен подход при задаването на параметрите на системите, базиран на йерархии от атрибут-стойност двойки с допълнителни мета данни. При него се използват много нива и формати на конфигурационните източници (файлове, бази данни и др.). Предложен е и механизъм за кеширане на нечувствителните от гледна точка на сигурност конфигурационни източници с цел ускорение.