AN ARCHITECTURAL MODEL OF RIGHTS MANAGEMENT FRAMEWORK FOR INFORMATION SYSTEMS[1]

Nikola Valchanov, Anton Iliev

Abstract: *The demands towards the contemporary information systems are constantly increasing. In a dynamic business environment an organization has to be prepared for sudden growth, shrinking or other type of reorganization. Such change would bring the need of adaptation of the information system, servicing the company. The association of access rights to parts of the system with users, groups of users, user roles etc. is of great importance to defining the different activities in the company and the restrictions of the access rights for each employee, according to his status. The mechanisms for access rights management in a system are taken in account during the system design. In most cases they are build in the system. This paper offers an approach in user rights framework development that is applicable in information systems. This work presents a reusable extendable mechanism that can be integrated in information systems.*

Keywords: information systems, user rights, framework, software architecture

2010 Mathematics Subject Classification: 68M14

The demands towards the contemporary information systems are constantly increasing. In a dynamic business environment an organization has to be prepared for sudden growth, shrinking or other type of reorganization. That kind of reorganization is related to the differentiation of new departments and opening new managerial positions. The distribution of the responsibilities between the departments leads to changes in their rights. Such changes demand adaptation of the information system, servicing the company. The association of access rights to parts of the system with users, user groups, roles etc is essential for the differentiation of the different activities in the company and the restriction of the rights of every employee, according to his status.

The rights management mechanisms in an information system are planned during the system design. Usually they are custom for the system they serve. That

[1] This paper is partially supported by projects RS09–FMI–041 of Department for Scientific Research, Plovdiv University "Paisii Hilendarski" and National Science Fund from 2010.

way the resources, demanded for the implementation of these mechanisms are invested in the beginning of each project.

This paper offers an approach for designing rights management frameworks for information systems. It presents a reusable and extendable mechanism that can be easily integrated in modern information systems.

Demands towards the framework

The main purpose of the right management mechanisms for information systems is restriction of user access. These restrictions are applied, based on previously defined rights, associated with system users and the structures that unite them. These structures are usually specific for the information system and depend on a series of factors like purpose of the system, scale of the servicing company etc. Despite that software giants offer different implementations of authentication management instruments that support users, roles, user groups, memberships etc. [1, 2].

The framework should not be customized to work with specific type of organization of the system users. This enforces the development of mechanisms for binding the system structures that should be affected by user rights to the framework.

In order to accelerate the development the framework must provide means for automated extraction and support of the system objects that are subject to user rights. The adequate support of this information between system versions is one of the most difficult problems in such frameworks. The stored information for the rights distribution within the system should not be lost between versions changes on the workstations. Regarding this the system should foresee workflows that ensure the data consistency when changing the versions of the information system.

The purpose of the framework assumes the development of visual components for rights management. They can be integrated in the systems that work with the framework. The graphical user interface in these components can be build dynamically, based on the registered associations between the framework and the system structures that can be given rights.

Let us review an architecture that meets these demands.

Inner processes of the framework

The main goal when designing such frameworks is accelerating and facilitating the development process of information systems. Regarding this they should be easily integrable, comfortable to work with and should demand minimum effort for supporting the instruments that they manage during development. This is handled by inner mechanisms for managing the rights types within the system, association of the entities of the information system with the

framework and generation of façade classes that work with the framework. These instruments facilitate the integration and the support of the framework.

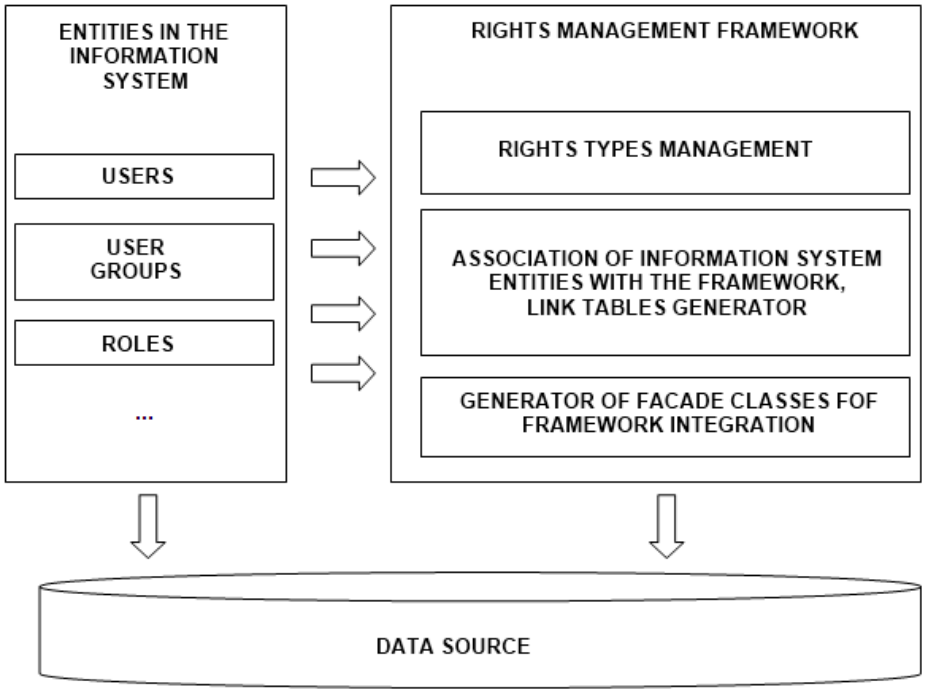

Fig. 1. Inner processes of the framework

Often during development of information systems occurs the need of additional functionality that wasn't foreseen in the planning stage. When working with fixed number of rights types adding a new rights type requires severe changes in the system's source code. The mechanism for rights types management brings the efforts needed for editing, adding or removing a rights type from the system to minimum.

The framework provides a specialized tool for registration of system structures that can be associated with rights. This instrument allows the developer to select the tables from the data source that model the system structures that can be associated with rights. Once the tables and their primary key columns are specified the instrument allows the user to enter association name for each table. The instrument generates many to many tables for each association. These tables allow the framework to assign rights to each of the registered structures.

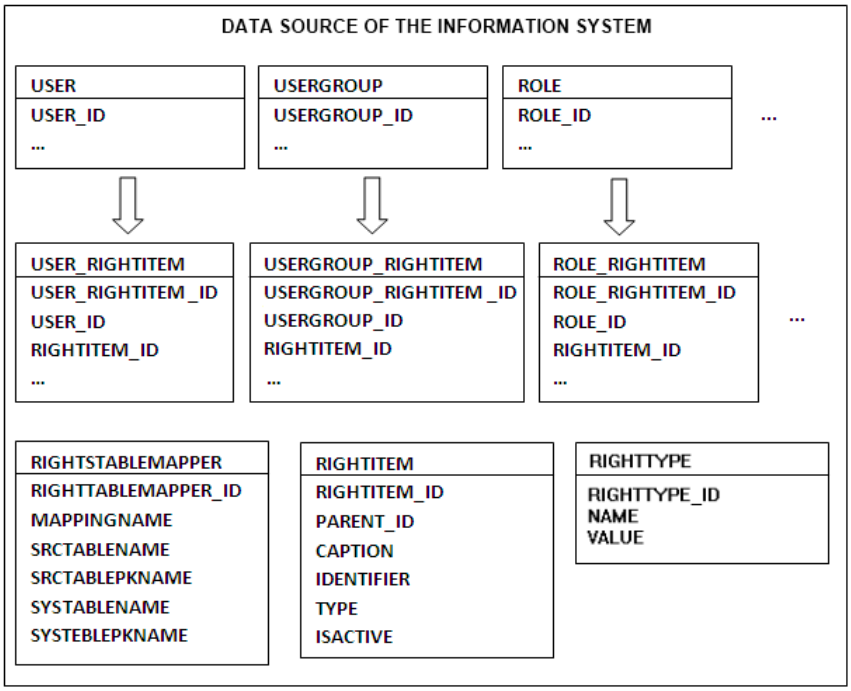

Fig. 2. Association of system structures with the framework

The framework provides an instrument for automated processing of system components that are subject to rights. This instrument indexes and extracts information about these components. After the extraction is complete this information is stored in the data source. During the persistence of the information the instrument checks for duplicate entries before adding a new record. When duplication is detected the new record is skipped. If the instrument identifies components that are registered in the framework but are no longer supported by the system the records for these components are removed. This way the instrument facilitates greatly the support of backwards compatibility between system versions.

The framework provides instruments for generating helper classes that facilitate the development process. The generated classes provide methods for identification if the rights a user, group of users, roles etc have over system components. Using these façades the information system establishes the communication with the framework.

Although the framework offers a good application programming interface (API) it provides visual components for rights management in the information system. These components can be integrated directly into the graphic user interface, as their inner logic does not depend on the structure of the information system.

Framework architecture

During the planning stage of the framework development our team invested a great deal of time in identifying the most probable technical challenges that might occur during the implementation. Foreseeing eventual technical obstacles during planning and resolving them by building abstract layers makes the system support easier.

The first obvious problem is information persistence. The framework uses the data source of the information system because it does not have its own. An abstract layer for data access assures the successful integration of the framework in systems that use different data sources. It isolates the data source from the business logic of the framework. This way the data source type and the connection information can be stored in the configuration of the framework.

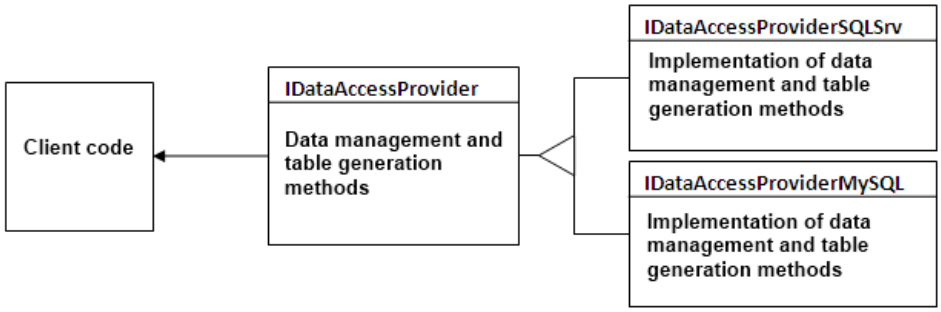

Fig. 3. Architecture of abstract layer for data access

Another specific aspect of the framework is the instrument for automated processing of system components that can be associated with rights.

The framework provides a set of interfaces that have to be implemented by the system components that are subject to rights. The interfaces cover two main component types – single purpose and multi-purpose component containers.

Single purpose containers offer a collection of records that contain a list of the managed elements within them. An element is identified by name, type and identifier that is unique within the container.

Multi-purpose containers allow their components to behave differently, depending on the purpose of the instance of the container. They offer a list of records that contain information of the different modes of the container. The containers within the information system are associated with unique identification number. Multi-purpose containers have multiple identification numbers one for each mode of the container.

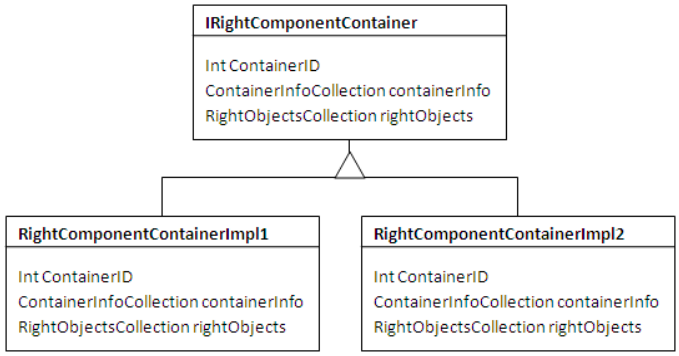

Fig. 4. Identification of components managed by the framework

The indexing and the processing of classes that are defined in a given assembly is possible thanks to the "reflection" technology. When the indexing starts the instrument finds all the dynamic class libraries in a given directory. Once they are identified the instrument processes the classes that are defined in each one of them. The classes that implement the interfaces provided by the framework are instantiated and their structure information is extracted. The gathered data is stored in the data source.

The framework generates classes that facilitate the communication with the information system. The generated code provides means for checking whether a given system structure has rights to access specific component. A checker method is generated for every structure that is associated with the framework. This way the framework provides the developers means to manage visual components based on access rights.

Conclusion

The business keeps trying to optimize the process of building information solutions. But though the generalization of system functionality into frameworks accelerates the development process there is no universal approach for designing them. The more logic the framework contains the less flexible are the instruments supported by it. That is why new frameworks dealing with problems in various areas constantly emerge.

The framework presented in this paper offers base functionality for rights management in information systems. The instruments provided by it automate much of the processing of system components that are subject to rights. This helps shorten the time needed for system development. The framework offers solutions to common problems related to supporting of information systems as backward compatibility and migration between system versions. The framework architecture facilitates the integration in new or legacy projects and allows simplified support and ease of extension of the framework.

References

[1] B. HAIDAR, Professional ASP.NET 3.5 Security, Membership and Role Management with C# and VB (WROX Programmer to Programmer), 2009.

[2] J. GARMS, MUIR, D., SOMERFIELD, Professional Java Security (Programmer to Programmer), 2003.

Nikola Valchanov, Anton Iliev
Faculty of Mathematics and Informatics
236, Bulgaria Blvd.
4003 Plovdiv, BULGARIA
e-mail: nvalchanov@gmail.com, aii@uni-plovdiv.bg
&
IMI, Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8,
1113 Sofia, BULGARIA