МАТЕМАТИКА И МАТЕМАТИЧЕСКО ОБРАЗОВАНИЕ, 2010
MATHEMATICS AND EDUCATION IN MATHEMATICS, 2010
Proceedings of the Thirty Ninth Spring Conference of
the Union of Bulgarian Mathematicians
Albena, April 6–10, 2010

ARCHITECTURE OF EXTENSIBLE COMPUTATIONS DRIVEN SYSTEMS*

Nikola Valchanov, Todorka Terzieva, Vladimir Shkurtov, Anton Iliev

One of the main areas of applications of computer informatics is the automation of mathematical computations. Information systems targeting various fields like accounting, e-learning/e-testing, simulation environments etc. work with computational libraries, usually specific for the scope of the system. Though such systems could be flawless and work for years without a single error, if not supported some day they become obsolete. We could create a mechanism that loads dynamically the computational libraries and makes runtime decisions (intelligent or interactive) about how and when to use them. This paper targets the architecture of such computations driven systems. It focuses on the benefits of using the right design patterns to provide extensibility and fight complexity.

1. Overview of computations driven systems. We face computations driven systems every day, sometimes without even giving a thought about the complex computational library working behind the interface.

For instance, we can mention e-testing environments like DeTC [1, 2]. Such systems provide tools for creating parameterized questions. These are usually mathematical problems with varying parameters. Every time a student has to answer such question the parameters are randomly picked from a previously given range of values. That type of questions work with a computational library that evaluates the generated problem and verifies the answer, given by the student.

Another example of computations driven systems are simulation systems like Simplex, Vensim [3] and Stella [4]. Those types of systems are used to create mathematical models and run simulations. They usually have model designers that allow the user to graphically build the model using system provided tools. Once the model is complete the systems generates equations and uses computational methods to simulate the model behavior.

Computation driven systems are not always directly bond to mathematics. Some accounting software uses computational libraries for evaluating statistical information. Such libraries are used in the pricing process. The implemented algorithms in these libraries are often well kept company secret and are considered as a valuable know-how.

All mentioned systems have something in common. They all use centralized computational libraries that allow them to function. So in order for them to be able to

*2000 **Mathematics Subject Classification:** 68U35.
Key words: computations driven systems, mathematical computations.
This paper is partially supported by projects ISM-4, RS09–FMI–041 and MU–3 of Department for Scientific Research, Plovdiv University "Paisii Hilendarski".

meet the user's demands, be extendable and prove themselves robust in time their design should foresee their possible evolution and provide mechanism that facilitates the system's support.

Computation library architecture. With the appropriate use of design patterns we could build an extensible framework that would facilitate support and allow third-party development of computational methods or other problem solving algorithms.

Simulation environments. Let's take the simplest case – a simulation environment. When a simulation starts the model is translated into context information that is passed to the computational library as argument. The computational library consists of one or several computational method implementations. It takes the context and decides which computational method to use for the simulation. Once the simulation starts the library returns a collection of values representing each result line from the iterations of the computational method. When the result is obtained, it is displayed in a suitable way to the end user by the user interface.

In order to achieve this, we need to separate not only logically, but physically the implementations of the computational methods used in the library. We could place each implementation in a different assembly and make it implement specific interfaces common for the library. This way the library could load runtime all the computational method implementations with the additional information from their manifests using reflections [5]. This architecture would enable third party developers to implement computational methods for the computational library. The folder of the assemblies and the default implementation could be specified in the application's configurations. This design pattern is used commonly by Microsoft's ADO.NET and is called "provider" [6].

Next we need the library to have a single entry point that conceals its inner complexity. For that we can use the "facade" pattern [7]. It would take care of extracting the context

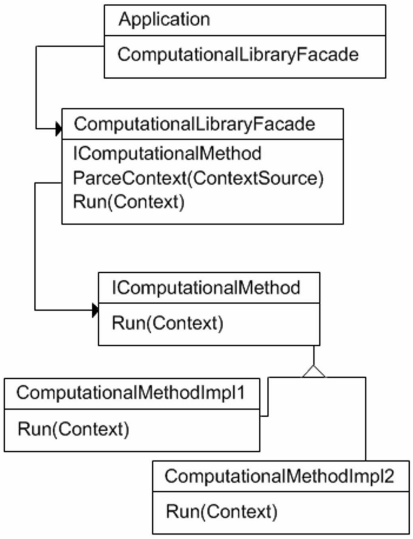

Fig. 1. Architecture of a simple computational library

from the context source, create the appropriate computational method implementation and run the simulation.

Here the facade would be used as a bridge class ("bridge" pattern) [7] that will act differently, depending on the specific instance (IComputationalMethod in Fig. 1. Architecture of a simple computational library).

2. Potential applications of Computations driven systems.

2.1. E-testing frameworks. When we start talking about e-testing frameworks, things get a little more complex. In the common case we have problems from different fields each of them with more than one way of solving.

Suppose we need a common computational library that has classes for solving the different problem types. A good start would be to manage the mapping of problem types and computational classes that solve them in a common way. This could be described in the metadata [5] of each problem solver implementation. That way the system would know what problem types it can solve based on the loaded solvers. This also allow the system to enable just the tools for managing problems for the available solvers.

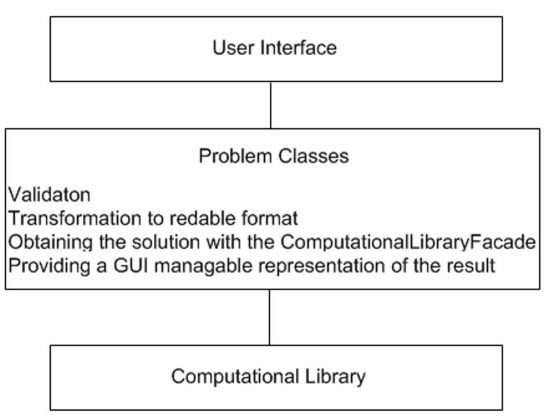

Fig. 2. Layering of an e-testing library

The vast range of problem types demands extracting the problems logic in a different layer. Each problem class contains context information about the problem it represents and has validation mechanisms that prevent the user from entering invalid data. It also contains methods for rendering the problem context information into a format that is readable by the computational library.

The computational library on the other hand searches for the problem type in the context information. If the desired solver implementation is present in the context information, the facade creates the appropriate instance and solves the problem with it. If not, the facade searches through the solver implementations and makes a runtime decision which one to use.

2.2. Business information systems (accounting software). Another area of implementation of computational frameworks is the accounting. The information systems servicing this field usually don't contain complex computations libraries. The relatively simple taxing though could prove to be difficult to support.

An important quality of a business information system is how well its functionality corresponds to the business processes in the client corporation. The separation of the layers and shaping of the communication between the modules more or less defines the extendibility of a system and the complexity of its support [8].

The initial planning of an information system is decisive for its future growth. The good design can make the system robust. It can guarantee easier support, simpler integration in different organizations and code reuse. There's no universal pattern for building business information systems. The choice of approach is still a subject of debates.

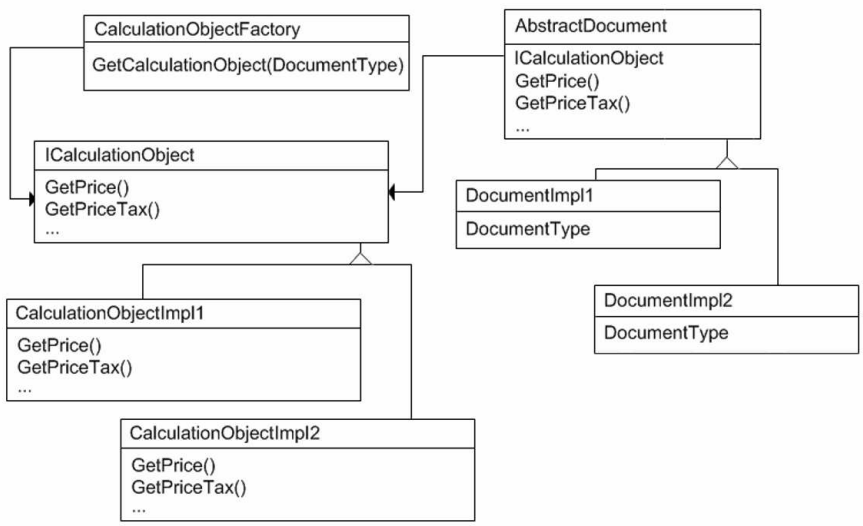

Fig. 3. Tax computation library design for information systems

3. Conclusion. Designing computational libraries usually cannot be generalized. Often computational libraries created for a specific system are ported and adapted for others using wrapper classes and glue code. Regardless of the library's original purpose its architecture should make it extensible. This way it has better chances withstanding changes in computational methods. This would make the library robust and eventually allow its reuse in other applications.

The discussed computation library architectures are extensible. They facilitate the system development and support. Make the projects more readable and allow development by third-party companies. The discussed fields of application vary between business, science and education and cover the basic problems in each of these fields.

REFERENCES

[1] O. Rahneva, A. Rahnev, N. Pavlov. Functional Workflow and Electronic Services in a Distributed Electronic Testing Cluster – DeTC. Proceedings of 2nd International Workshop on eServices and eLearning, Otto-von-Guericke University Magdeburd, 2004, 147–157.

[2] A. Rahnev, O. Rahneva. Examination and Evaluation in Accounting through Distributed Electronic Testing Cluster – DeTC. *Scientific Works of European College of Economics and Management*, **4**, (2008), 182–189.
[3] Vensim simulation software, `http://www.vensim.com`.
[4] Stella simulation software, `http://www.iseesystems.com`.
[5] J. Hart, B. Mathew, S. Gilani, M. Gillespie, A. Olsen. Visual Basic. NET Reflection Handbook.
[6] Microsoft Corporation, Provider Model Design Pattern and Specification, `http://msdn.microsoft.com/en-us/library/ms972319.aspx`.
[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley, 1994.
[8] N. Valchanov, T. Terzieva, V. Shkurtov, A. Iliev. Approaches in Building and Supporting Business Information Systems. Collection of Scientific Works of International Scientific Conference, Varna, 16–17.10.2009, (accepted).

Nikola Valchanov
Todorka Terzieva
Vladimir Shkurtov
Anton Iliev
Faculty of Mathematics and Informatics
Plovdiv University "Paisii Hilendarski"
236, Bulgaria Blvd.
4003 Plovdiv, Bulgaria
e-mail: nvalchanov@gmail.com
e-mail: dora@uni-plovdiv.bg
e-mail: vsh@uni-plovdiv.bg
e-mail: aii@uni-plovdiv.bg

Nikola Valchanov
Anton Iliev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria

АРХИТЕКТУРА НА РАЗШИРЯЕМИ, УПРАВЛЯВАНИ ОТ ИЗЧИСЛЕНИЯ СИСТЕМИ

Никола Вълчанов, Тодорка Терзиева, Владимир Шкуртов, Антон Илиев

Една от основните области на приложения на компютърната информатика е автоматизирането на математическите изчисления. Информационните системи покриват различни области като счетоводство, електронно обучение/тестване, симулационни среди и т. н. Те работят с изчислителни библиотеки, които са специфични за обхвата на системата. Въпреки, че такива системи са перфектни и работят безпогрешно, ако не се поддържат остаряват. В тази работа описваме механизъм, който използва динамично библиотеките за изчисления и взема решение по време на изпълнение (интелигентно или интерактивно) за това как и кога те да се използват. Целта на тази статия е представяне на архитектура за системи, управлявани от изчисления. Тя се фокусира върху ползите от използването на правилните шаблони за дизайн с цел да се осигури разширяемост и намаляване на сложността.