EnergoWatcher – the platform for creating adaptable energy monitoring systems

Evgeniy A. Kalashnikov
Department of Software and Computing Systems Mathematical Support
Perm State University
Perm, Russian Federation
E-mail: keatrance@gmail.com

***Abstract*. Energy monitoring systems are widely spread among large productions and smart houses. It's designed to collect, process, store data and provide users with aggregated and detailed information about the consumed resources. However, the developing price of these systems is large enough. It is required to have a technology for fast and cheap creation of power consumption monitoring system. These systems should be adaptable to varying of operating conditions and designed for small objects. EnergoWatcher is a development technology of program complexes of power consumption monitoring, which is described in this article. The systems created by means of this technology suppose dynamic adapting to varying of operating conditions and users requirements. The software provides possibility of connection to various sources of the information (to controllers of technical state of objects, to databases of the external applications, and so on).**

Domain-specific modelling; monitoring; metamodelling; power consumption; EnergoWatcher;

I. Introduction, Problematic

This paper is devoted to the power consumption: a lack of information is the main cause of wasteful use of resources, since you can only effectively manage those that can be measured.

The problem of the resource consumption dynamic opacity can be achieved through creation and using **monitoring systems** [1, 2] that allow you to automate the process of collecting data from external sources (sensors, meters, indicators and so on) to obtain a more complete picture of what is happening. It should be noted that data from the energy accounting devices are handled by software developed by equipment manufacturers to work with specific devices. This specialized software often makes it impossible to integrate data from different sources, processing and presentation in one program. The key objectives of monitoring systems are to collect, process, store data and provide users with aggregated and detailed information about the consumed resources collected from heterogeneous sources.

It should be noted that the need for such monitoring systems are increasingly seen not only in industrial scale (large production), but at the household level. There are government agencies and private companies seeking to control the level of water / heat / electricity consumption etc. in order to save his own funds. However, the monitoring system - an expensive, "custom made" product, developed under the specific type and configuration of equipment, so there is a problem on the establishment of flexible software, which could be configurable while condition changes of maintenance (connection of new devices, monitoring of certain parameters, etc.).

So it’s necessary to have a technology for easy dashboard construction. Using generative programming approach we could generate monitoring systems with minimum costs without involving programmers. Such software can be reconfigurable for new connected devices and their parameters.

The main requirement for such technology is the ability to adapt the system constructed under specific operating conditions [3]. The system should be dynamically adjusted to the new hardware and user needs, without the programmer (without changing the source code and subsequent recompilation). To this end, a toolkit is being developed, that allows construct virtual panel for energy consumption monitoring, which in future will be discussed by experts and will lead to relevant policies to improve energy efficiency. In this paper we propose a technique EnergoWatcher - the platform for creating adaptable energy monitoring systems.

II. Monitoring systems: architectures, approaches

There are two basic approaches to the development of monitoring systems [4] - "manual" and "automatic". In the first case, the system is programmed from scratch for a specific hardware system architecture; system's components are strongly coupled; the code is optimal in terms of performance. However, it would lead to significant development costs, inflexible solutions and inability to reconfigure the system when adding new hardware.

While using an automatic generation of software development the code and database generation happens on the basis of the system’s model defined by developer. This is the standard approach with CASE-tools [4], based on transformations of the original model of monitoring system in code. In this case, we have a universal solution to the high rate of development to the detriment of generated code performance. However, this solution is not flexible too, because the slightest change in the architecture of complex hardware / system requirements will need to re-generation of source code and database.

In both approaches adapt facilities of the final applications are limited. The specificity of the modern software is that the application life cycle never ends with the completion of its development. Therefore it is required to provide the ability to dynamically configure system to new operating conditions, the needs of its users.

To create an adaptable system of this kind is needed to store a description of the hardware configuration (new connected devices, parameters, their tracking), and other settings. If you change the configuration, the application code should not be changed. The program functioning is based on the interpretation of these descriptions (metadata) [1]. User interface (UI) is generated based on this metadata every time when you run an application, what makes possible to construct and keep UI settings. That mode of interpreting the metadata allows achieving the necessary flexibility and balance between the rate of development, flexibility and adaptability of the application possibilities.

III. Architecture of EnergoWatcher system

The core of the system is metadata that describes components of the system from different viewpoints [2]. Metadata's model (meta2model) of logic level is the main. It is based on the ER-model of Chen (entity-relationship model). Each entity is an abstract real-world object (e.g., meters), with its own properties (attributes) and relationships with other entities (links). Instances of the entity represent the existing concrete objects and their attribute values and relationships reflect their current state.

Let's consider the EnergoWatcher's application architecture, the multi-level monitoring system. The system consists of five logical and physical levels (see Fig. 1).

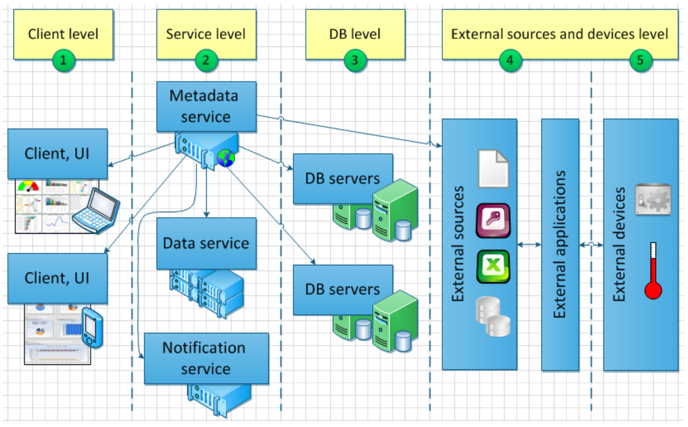

Figure 1. Levels in EnergoWatcher's architecture

Levels 4 and 5 are responsible for connecting to external sources, configuring the import data parameters. The data, which obtained with the metering of energy consumption, are handled by foreign applications. After processing the received device parameters are stored in external sources (databases, spreadsheets, CSV files, etc.). The system asks for new sources of data at defined intervals of time. If this data is available, the system processes import into the database (level 3) caching obtained values at the same time [5].

For each configuration of hardware on database server (level 3) based on the metadata its own database is created to store rates selected for import. The database structure is determined by the metadata of logical and physical level (Fig. 2) in accordance with the existing hardware configuration. It is updated automatically at metadata changing via component restructuring.

The components of levels number 1 and 2 are part of a client-server service-oriented architecture. The end-client establishes a connection with the services in polling or duplex mode and receives data (data services, and events) and metadata (metadata services and settings UI). Metadata service is available on user request and provides metadata of logical level. UI service, which is actually part of metadata service, represents metadata of presentation level. Event service notifies connected clients automatically if emergency situations arise. Data service sends new data when new imported values appear on server side. The client defines its own interface of dashboard and configures the connection UI controls to data sources (binding) in return.

The logical connection between these levels is provided by interdependent layers of metadata (MD), describing the system from different angles (Fig. 2). Key metamodel - a model of logic level - provides a link of metadata and system operation under terms of a specific domain for the user (in this case in the field of energy auditing).

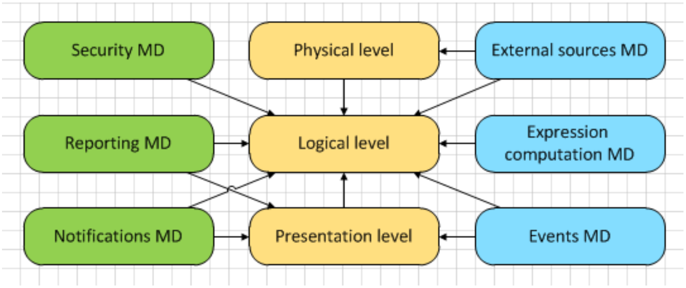

Figure 2. Interdependent metadata (MD) layers

IV. Requirements for UI system

One of the main goals of the monitoring applications is a convenient and timely notification of the ongoing processes, emergency situations, displaying the real picture of energy consumption (energy audit) for further analysis [6]. It's all about the data visualization, which perhaps has the most important value in operational dashboards.

In general, to create own interface user (or administrator) will need to perform several actions:

- create instances of the visualization components (controls);
- place these controls in the required locations, given their mutual nesting and hierarchy;
- configure the basic properties of the controls;

- specify the data sources for display - those attributes, whose values you want to display, and, if necessary, set the missing values.

To ensure maximum system flexibility and adaptability to dynamically changing the operating conditions were determined following requirements for UI system [7]:

1. Setting up the visual interface should be simple and understandable to the end user-nonprogrammer:

- using of intuitive action in WYSIWYG way: Technology Drag & Drop, the component is resized with the mouse, etc.;

- a set of display properties for component configuring and their descriptions should be dynamically configurable - in fact, they become part of the metadata system. It is required to display user only necessary properties and group them into predefined categories;

- the problem of localization of all the properties of the components need to be solved: they need to be shown in the user native language (in our case, in Russian) when configuring the component. Thus, ordinary users are unlikely to know what the terms mean "Width" and "Height";

- meta-description of the parent control properties (within the concept of the OOP) to be inherited by child controls with the ability to override them (to avoid massive duplication of descriptions and provide illustrative descriptions).

2. The set of supported components to be dynamically extensible:

- adding a new component must be clear to the user without data addition in the database;

- format for storing information on the control and its properties should be open – e.g., XML;

- .NET component of visualization must have clear interfaces to implement the logic to display the incoming data and, if necessary, entry of missing values on data service.

3. The UI components must support one of the possible types of data binding:

- none binding: component is used only to display static information and / or grouping other components;

- single binding: component displays only one parameter of the system over time (e.g., reading a thermometer);

- multi binding: component displays multiple parameters of the system over time (e.g., a graph with multiple indicators).

4. All changes in the system should be applied at run time, without recompiling the source code, using the principle of WYSIWYG. In many systems, the interface is not the model is interpreted at runtime, but only serves to generate user interface code [6, 8].

It should be noted that the controls themselves, as well as connected data sources, should be able to be dynamically connected to the system without recompiling the source code.

V. Presentation Metadata and Binding Components

In EnergoWatcher technology, based on the interpretation of interrelated metadata, user creates his own dashboard, described by the metadata of presentation layer (Fig. 3). User selects indicators what he wants to watch and sets their configuration, using the means of creating a visual interface of customizable panel in EnergoWatcher.

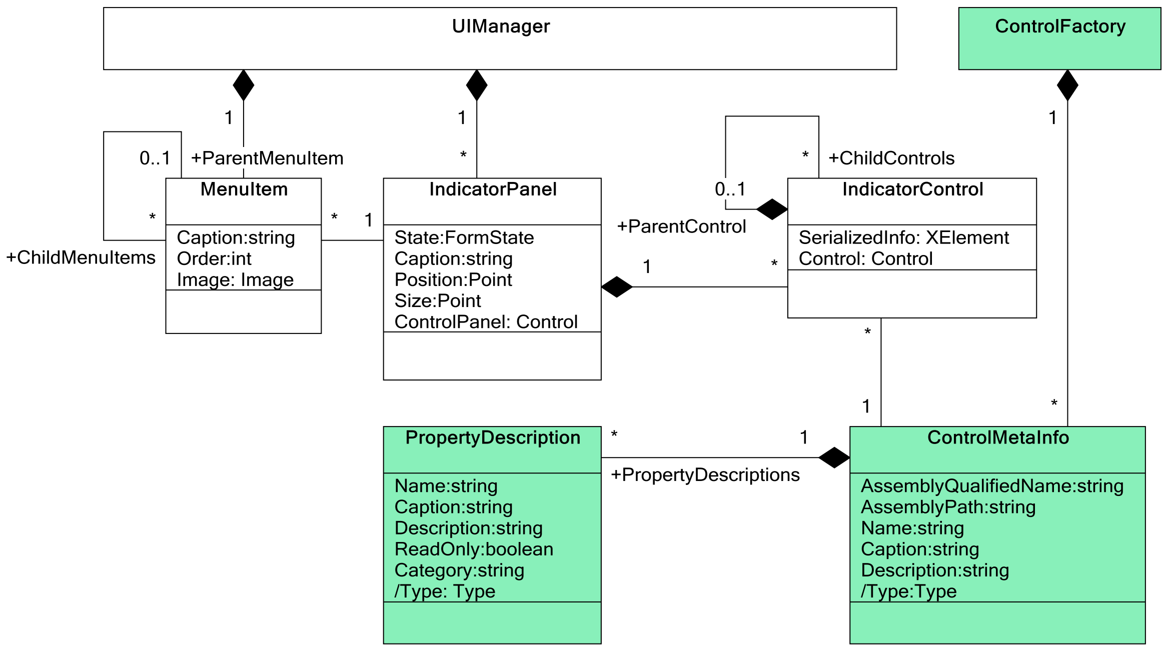

Figure 3. Two parts of presentation layer metadata

For the dynamic connection management components and configuration UI presentation layer metadata is divided into two levels: UI level and level of the interface components.

The UI level (see Fig. 3, classes without background) describes the user's desktop: tree menu (menu items *MenuItems*), and a lot of dashboards *IndicatorPanel*. Dashboard is a collection of custom visual components *IndicatorControl*, located in the hierarchy relative to each other. Each component has its own internal serialized state, which is used to saving and restoring the configuration user interface. During system operation the visual components are dynamically linked to the sources, the values in which come from the server through a duplex channel.

Component level (see Fig. 3, green background) contains meta-information about the components of imaging: an indication of the type and assembly, in which the control is located, as well as a list of descriptions of the component properties. Meta-descriptions are stored in XML file, resulting in the connection of new components takes place directly at run time, besides the issues of descriptions localization and names of configurable properties are resolved.

As mentioned above, the presentation model has a connection with logical model. It provides binding UI components to server sources. There are 2 kinds of objects in the capacity of server sources: attributes and expressions (see Fig. 4). Every time when the application starts, controls establish communication with *LightSource* element. Server metadata [1] of available sources is too heavy to be transported to client, that's why there is light representation of them (it is realization of pattern "Data Transfer Objects", DTO).

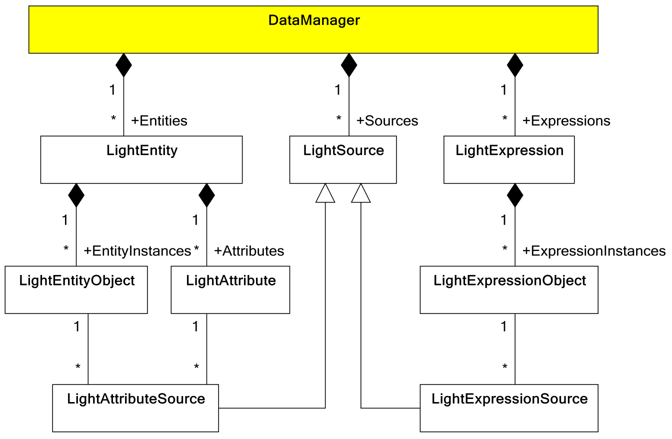

Figure 4. Client metadata DTOs for data management

LightAttributeSource represent values of attribute *LightAttribute* of instance *LightEntityObject* of entity *LightEntity*. *LightExpressionSource* represent values of expression instance *LightExpressionObject* of expression *LightExpression*, which is computed on server. Actual data is collected from server automatically using duplex mode of service communication.

Conclusion

This paper proposes an approach to building monitoring systems, based on an interpretation of metadata in real time. The article describes presentation model of EnergoWather system for building user interfaces in runtime by WYSIWYG way.

At this moment system are being developed and it is planned to product a prototype version of EnergoWatcher. It's need to implement set of subsystems:

- operations with event metamodel, including describing specific events in domain specific language (DSL);

- straight connections to devices in real time using standart transport protocols;

- data analizer for automatic recognition of leaks and other emergency situations;

- services for supplying data, events and metadata to clients;

The system was tested at the tenders U.M.N.I.K. and B.I.T., nowadays it is in developing state. It's planned to integrate it to municipal authorities and resource-demanding facilities and provide the power consumption audit.

The proposed system focused on monitoring in order to save own funds. First of all, it will be interesting to introduce EnergoWather in small industrial and social facilities (schools, hospitals, kindergartens), as well as other municipal property objects. Generated software will analyze the dynamics of daily energy consumption and compare it with historical state. So it is possible to find an illegal consumption size as well as to track the occurrence of abnormal situations.

References

[1] V.A. Voronov, E. A. Kalashnikov, L.N. Lyadova Technology of development of power consumption monitoring system // Proceedings of the Congress on Intelligence Systems and Technologies "AIS-IT'10". Scientific publications in 4 volumes. Moscow: Physmathlit, 2010, Vol. 4.

[2] E. A. Kalashnikov Tools for creating dynamic dashboards of energy consumption on the basis of related metadata models // III All-Russian Student Research Forum (electronic conference, http://rae.ru/forum2011/104/285).

[3] Joseph W. Yoder, Ralph E. Johnson: The Adaptive Object-Model Architectural Style. WICSA, 2002, pp. 3-27.

[4] K. Czarnecki, U. Eisenecker Generative Programming: Methods, Tools, and Applications. Reading, MA, USA, Addison-Wesley, 2000.

[5] E. A. Kalashnikov Caching in the systems for monitoring of energy consumption // IV All-Russian Student Research Forum (electronic conference, http://rae.ru/forum2012/219/2697).

[6] Ekkerson U. Performance Dashboards: Measuring, Monitoring, and Managing Your Business. Moscow: Alpina Business Books, 2007.

[7] E. A. Kalashnikov Creation user interface in WYSIWYG way for energy consumption indicators monitoring system. Math of software system, Perm University, Perm, 2010, pp. 84-91.

[8] Mohan R., Kulkarni V. Model Driven Development of Graphical User Interfaces for Enterprise Business Applications // MODELS'09 Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems. Springer-Verlag Berlin, Heidelberg, 2009, pp. 307-321.