Middleware Based Model of Heterogeneous Systems for SCADA Distributed Applications

Nicoleta-Cristina GĂITAN[1], Vasile Gheorghiță GĂITAN[2], Ștefan Gheorghe PENTIUC[3], Ioan UNGUREAN[4], Eugen DODIU[5]
Ştefan cel Mare University of Suceava
str.Universităţii nr.13, RO-720229 Suceava
[1]*cristinag@eed.usv.ro,* [2]*gaitan@eed.usv.ro,* [3]*pentiuc@eed.usv.ro,* [4]*ioanu@eed.usv.ro,* [5]*edodiu@usv.ro*

***Abstract* — Infrastructure underlying the distributed information systems is heterogeneous and very complex. Middleware allows the development of distributed information systems, without knowing the functioning details of an infrastructure, by its abstracting. An essential issue on designing such systems is represented by choosing the middleware technologies. An architectural model of a SCADA system based on middleware is proposed in this paper. This system is formed of servers that centralize data and clients, which receive information from a server, thus allowing the chart displaying of such information. All these components own a specific functionality and can exchange information, by means of a middleware bus. A middleware bus signifies a software bus, where more middleware technologies can coexist.**

***Index Terms* — middleware, SCADA, client-server technology, heterogeneous system, distributed system**

I. INTRODUCTION

The essential part of middleware [1] consists in managing the complexity and heterogeneousness of the distributed infrastructure. On the one hand, the middleware provides the abstracting of programming [5], so as to hide the complexity of a system's building; on the other hand, there is a complex software infrastructure that implements this abstracting.

The architectural model of a heterogeneous system oriented on SCADA applications and based on middleware is presented in this paper. One might learn some simple lessons:

1. The complete transparency (an application's property on preserving the behavior when passing from a centralized environment towards a distributed one) is not feasible [2]. The transparency has been an aim since the beginning of middleware. The good engineering practices admit the limitations of transparency and also accept the fact that distributed applications have to taken as such; these focus at least on fundamental aspects related to fault tolerance and performances [3].
2. Some useful patterns have been revealed. One of the most common patterns of the middleware consists in using a local representative (the Proxy design pattern), in order to organize the communication between remote entities. Another universal pattern is the one accomplishing a correspondence between client and server, by means of a service name, which acts as a registry (the architectural pattern named Broker). Other patterns less apparent are used in order to organize the server's activity, by means of creating the lines or area of lines (pooling), or by its reaction on falling, by using detection-reaction schemes.
3. The development of a distributed application, even when a conceptual simple scheme is used, such as RPC, which involves an important engineering infrastructure: IDL generators and stubs, unitary representation of data, (un)marshallers (serializers/deserializers), mechanisms of fault tolerance, name services, development instruments.

As in the situation of processors, there is no architecture able to satisfy the entire requirements specific to applications diversity. Some architectures react better to certain requirements, such as the distributed applications necessary to implement the databases; other react better to some other requirements, as those related to systems of data acquisitions.

The middleware classification will be represented by: transactional, object oriented, procedural and object or component oriented. To this classification, one might add the flexible (adaptive), reflective, scalable, real time, of mobile calculus, peer-to-peer, grid, model driven and QoS-enabled middleware.

Web services do not bring anything revolutionary. These services represent a normal progress on technologies development of computers science and are practically based upon component programming. The support for such technology seems to be amazing, since it is based on XML accepted by all organizations. Starting from the proof performed by [4], web services will be assigned as both middleware and software component.

II. MINIMAL REQUIREMENTS AND BASIC ARCHITECTURE OF THE PROPOSED MODEL

The minimal requirements imposed for the proposed model are:

- To be oriented on SCADA applications, necessary for process data acquisition;
- To support wide distributed and heterogeneous systems;
- To prove real time characteristics;
- To be scalable;
- To use the already existing technologies and to adapt easily to new technologies.

Figure 1 illustrates a first approximation of this model.

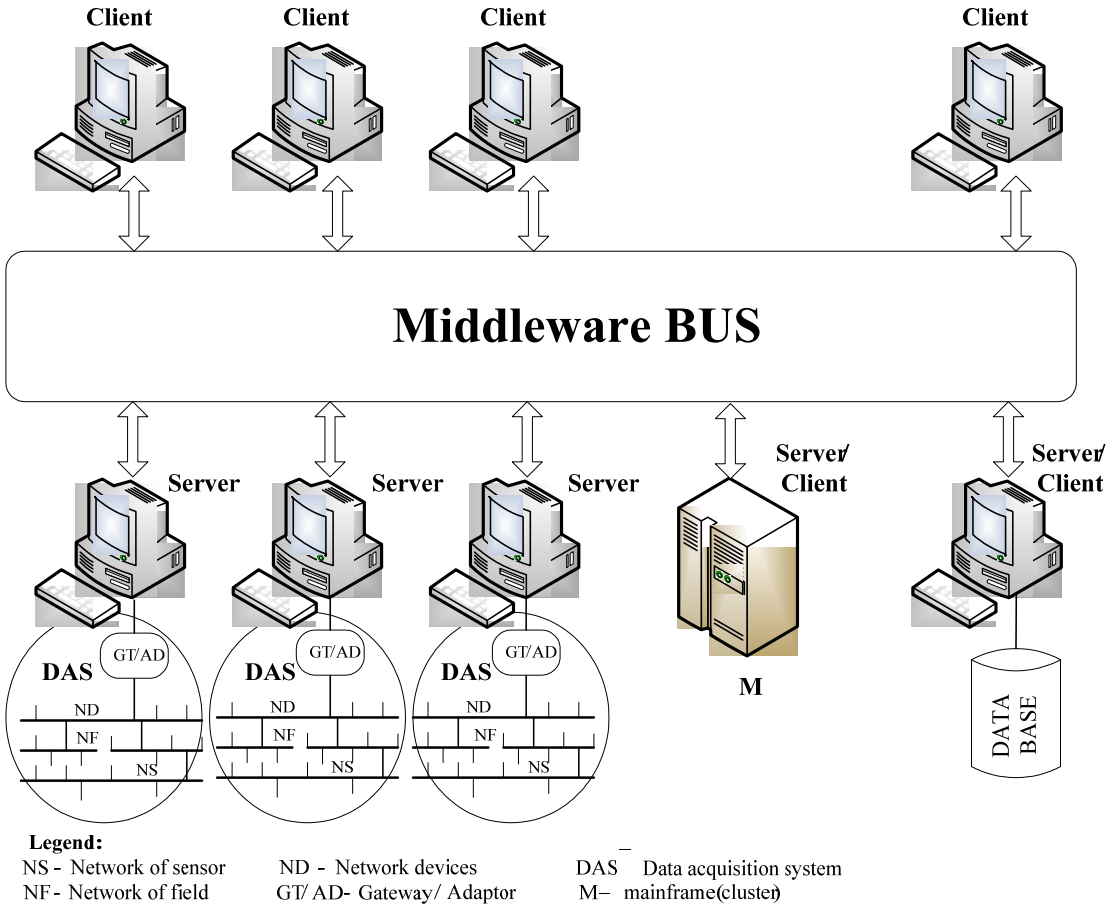

Figure 1. Block scheme of the architectural model, specific to a heterogeneous system oriented on SCADA applications and based on middleware.

Concerning this model, some ascertainments have been imposed:

- Middleware bus signifies a software bus, where more middleware technologies can coexist, and which allow the implementation of the client/server model (and also peer-to-peer, if it is necessary) between the client applications and server applications. The middleware has to ensure the client server communication for the next three situations:
  1. the client and the server are placed on the same computer and:
     - ✓ are part of the same process (there are potentially different execution lines) or,
     - ✓ are various processes, which run on the same kernel (processor) or run on different processors, in the situation of multi-kernel or multiprocessor architectures (usually, the operating system provides some mechanisms of communication and synchronization between processes, such as local procedures calls, pipes with names, spinlocks, etc.);
  2. are placed on separated computers, but connected within local networks, case where the client and server communication can be optimized, so as to reach superior performances concerning the data transfer rate (a potential solution would be the immediate using of TCP/IP stack)
  3. are placed on separated computers, but connected in Intranet, Internet or Extranet networks.
- choosing the middleware will be accomplished depending upon specific features of the application, such as: imposing real time requirements; creating and using some databases; the applications chart; connecting to applications placed on superior levels of an institution's organization; already existing software licenses; the infrastructure of the existing networks; constraints imposed by other applications;
- a host computer can support either a client application or a server application, or both of them;
- a server application can be the client of other server applications;
- a server does not compulsory have to connect to a SAD (it can be connected to a database);
- a mainframe can be chosen (cluster or grid), so as to implement a server of collecting the data from other servers, and which subsequently will create a database, by means of processing the data achieved, able to face the requirements launched by a high number of clients;
- the client application will be object oriented, and objects will be provided with one or two interfaces:
  - ✓ the standard interface towards the client application – allows the communication between the application's objects;

  - ✓ the interface towards the middleware – allows the connection to a server, by using a specific middleware application.
- the objects of client application will be of three types:
  - ✓ normal objects – are those objects connected by the standard interface to other objects of the client application, which can represent data consumers, data producers or both of them;
  - ✓ objects of displaying (text or chart) – are the same as normal objects, with the distinction that information of the human operator is shown under text or chart form;
  - ✓ acquisition objects – are the same as normal objects, but are provided with a supplementary interface, which ensures the connection to middleware and implicitly to servers implemented upon basis of that middleware (these objects allow the connecting of client application to more already existing middleware, as well as the immediate adapting to those newest).
- the servers can be of three types: of data (DA); of alarms and events (AE), historical (HDA) or combinations of these;
- the server has to implement a software component for data acquisition (SCDA), on which a standard interface will be connected, no matter the middleware used, so as to allow the connection to SAD or databases; SCDA will also have to define another standard interface, able to allow the coupling of drivers specific to the significant number of protocols within local industrial networks field;
- at the level of SCDA, two types of special drivers will exist, meaning: one able to allow the simulation of data acquisition (for abbreviation, acquisition means both the properly speaking acquisition, and the commands sent towards the process), and the second one, able to stimulate the client towards another type of server;
- SCDA will allow the hierarchical organization of the process data or of other types of data, under the form of objects' dictionary, easily accessible of server's level and implicitly, of the client application;
- A unitary language on SCDA level should also be defined, so as to describe the equipments or data for their placing into objects' dictionary, and a file manager should be also implemented, so as to manage the dictionary and the local industrial networks.

III. THE ARCHITECTURE OF CLIENT AND SERVER APPLICATIONS

In Figure, the model proposed at the client and server level is described.

The proposed level should also ensure the integration with other hierarchical levels of the enterprises. Within the control of industrial processes, the simplified hierarchical architecture of information involves the following levels:

- The business and processes management: dedicated systems exist for the management of resources and of relationships with the clients and providers. The main task at this level consists in the analysis of the current state, in providing support for decisions making, in adjusting the parameters, in managing the infrastructure and in providing support so as to plan the activities.
- The most known applications at this level are ERP (Enterprise Resource Management) – the management of enterprise resources, SAP (Systems Analysis and Product) / systems of analysis and production, SCM (Supply Chain Management) / the management of supply resources.
- PLM (Product Lifecycle Management) – the management of products life cycle.
- The control systems of the processes: these are in generally responsible for planning of operations and commands, with the view of leading the process as an entire. The applications at this level include all types of display modes to manufacturing and supervising. Typically, the systems placed at this level are: MES (Manufacturing Execution Systems) – systems of execution manufacturing, SCADA (Supervisory Control And Data Acquisition) – supervising the control and data acquisition, HMI (Human Machine Interface) – the human-machine interface.
- The management of field devices: takes care of digital field devices, which control and adjust in real time the processes, using a feedback control (close loop). As concerns the distributed processes, islands of automation can be emphasized, whom cooperation has to be smoothed by a supervision system. on principle, the main functions of this level are carried out with PLCs (Programmable Logic Controller) – programmable logic controllers or DCS (Distributed Control System) – systems of distributed control.

IV. CONCLUSION

Notwithstanding, these levels do not know too clearly the limits and competences, but a vertical communication with the process level is always necessary. This communication needs:

- an adequate level of reliableness;
- to meet the temporal requirements;
- to have regard for limits of delays;
- using a diversified structure of communication;
- using of standards specific to manufacturer, so as to access the process data;
- using of open architectures;
- A uniform model of displaying the data.

The novelty elements of the model are:

- Middleware objects of the client application allow the adding of new servers, which were developed by using environments of development and integration on applications, specific to various types of middleware technologies (such as ACE and MICO for CORBA, OPC UA for web services, OPC Xi for WCF, etc.). For these applications, only the software for the middleware object has to be written, which will perform the access and will expose the dictionary of objects, supported by the server.

- All servers see the same type of objects dictionary, and as result, a standardization of data exposition will be accomplished, both at servers and application levels (the middleware object will expose data in the same manner, so as to allow data to be accessed by other objects of the application).
- Implementing a standard interface at the application level allows the facile connecting of objects to each others, so as to communicate various data.

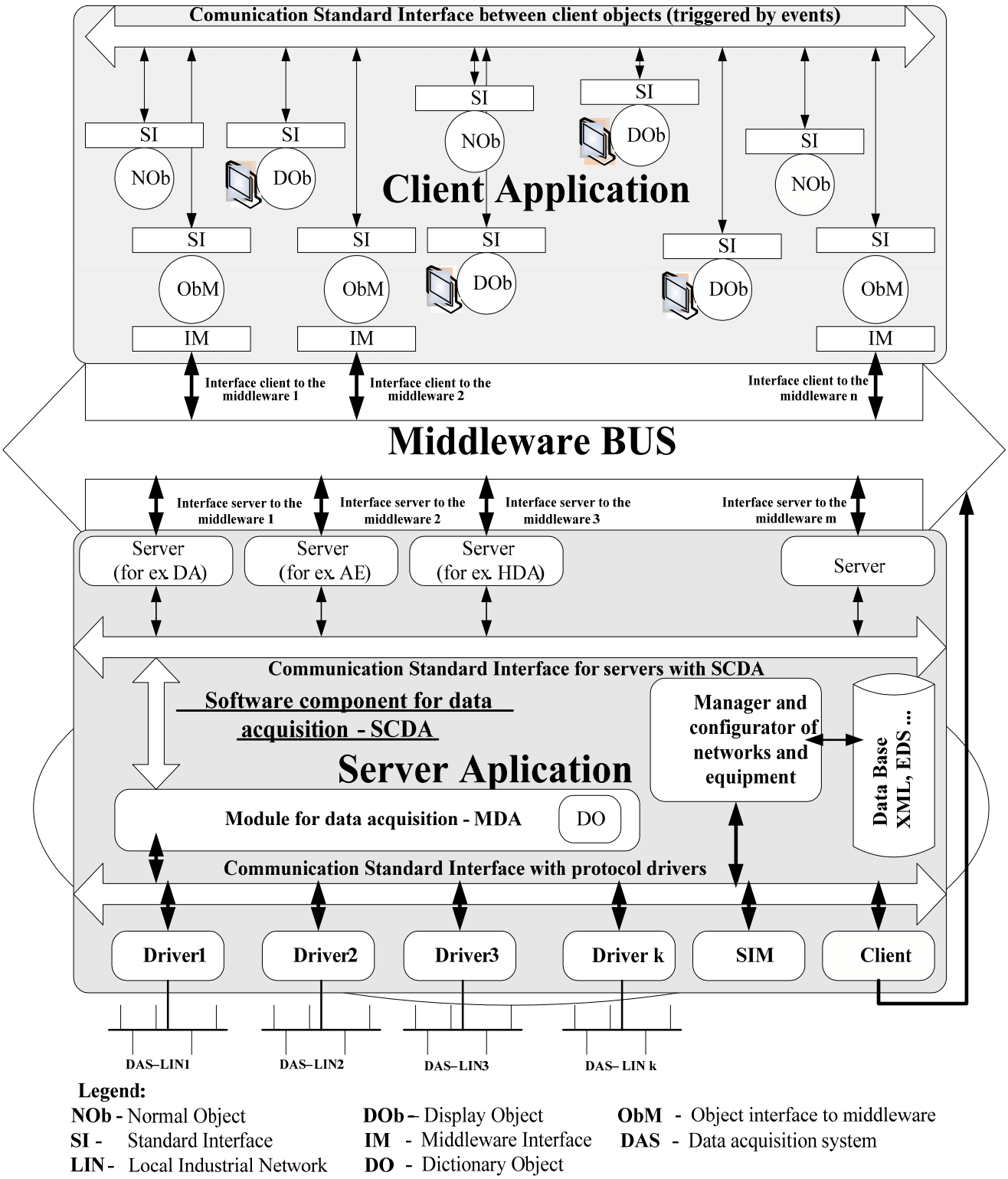

Figure 2. Representation of the proposed model, as concerns the clients and servers structures

- Moreover, the middleware objects can immediately connect in an easy way, without the server and also to other applications of process data collection, where the exposition to client application will remain unitary.
- All servers, no matter the middleware technology, see in the same way the process data, by means of objects dictionary, wherewith they communicate by a standard interface. All devices or other types of data are unitary described, by using the same description language.
- The second interface allows the connection to server of different written drivers on various protocols, by means of simple "wrapper" (casing or adaptor) software.
- The extension of functionality specific to client applications, by adding new objects, and specific to server applications, by adding new networks, without rerunning the already existing programs.

V. FUTURE WORKS

The future extensions of the model will ensure the compatibility with standards of describing the devices, as AutomationML, OpenPLC XML, EDDL, FDT or CANOpen.

REFERENCES

[1] D. Eberle, Semantic Management of Middleware. Springer, ISBN 978-0-387-27630-4, 2006.

[2] S. Krakowiak. Middleware Architecture with Patterns and Frameworks. Distributed under a Creative Commons license, http://creativecommons.org/licenses/by-nc-nd/3.0/, 2009.

[3] J.Waldo,; G. Wyant,; A. Wollrath,; S. Kendall. A Note on Distributed Computing. In Vitek, J. and Tschudin, C., editors, Mobile Object Systems: Towards the Programmable Internet, volume 1222 of Lecture Notes in Computer Science, Springer Verlag, 1997, pages 49–64.

[4] D. Karastoyanova,; A. Buchmann,. COMPONENTS, MIDDLEWARE AND WEB SERVICES. In IADIS International Conference WWW/Interne, Volume II, IADIS, 2003.

[5] R. E. Schantz, D. C. Schmidt. Middleware for Distributed Systems, Evolving the Common Structure for Network-centric Applications. Wiley Encyclopedia of Computer Science and Engineering, 2008.