Design Stages of Self-Organizing Distributed Systems

Mikhail Kupriyanov

Electrotechnical State University
Saint-Petersburg, Russia
e-mail: mikhail.kupriyanov@gmail.com

Andrew Kochetcov

Electrotechnical State University
Saint-Petersburg, Russia
e-mail: gaidn@yandex.ru

Abstract. The questions of self-organizing distributed systems design are considered in this article. Ten main stages of such design are described. Special attention is made for the possible issues at every stage and methods for resolving them.

The design of any complex systems, which include self-organizing, should be conducted on the models of such systems. In the course of the research a model of self-organizing systems was created, and the software to simulate the processes of self-organization on the basis of this model was developed, descriptions of which can be found in [1,2]. Note that the feature of the design of self-organizing distributed systems (SODS) is the need to carry out, in general, a large amount of preparatory work, not directly related with the introduction of the model in the simulator, or starting the process of modeling. The following report focuses on all stages of SODS design, as well as the implementation of these steps on the simulator.

Step 1. Defining the goals of the system, the formation of the target functional. Any self-organizing system, by definition, must have a purpose of its operation. There is some difficulty in defining the goals of the SODS functioning due to the following key issues:
- Goals should have a clear mathematical formulation, and to be represented by a real numbers;
- The system may have several purposes, some of which may carry conflicting nature. At this stage, one has to be formed - the overall goal of the entire system.

- The purpose of the system must be represented in the form of a mathematical function whose arguments are the parameters of the elements that constitute the system, and the parameters of the system as a whole. Note that all of these parameters should also have a numerical interpretation.

After the system goal definition is made, it is necessary to form the mathematical model of the system goal - the target system functional. It is necessary to take into account the following factors:

- The dimension of controlled variables should be minimal, otherwise the running time of self-organization can be not acceptable;
- All variables in this function must be normalized and have the same measurement units.

Step 2. System topology design. At this stage we should build the common model of the distributed system. We assume that the SODS has a tree structure, otherwise, if we have connections in the element graph forming some cycles, further advice may prove to be incorrect. It must also be prepared to answer the following questions:

- Determiningof how many levels of the hierarchy has the system;
- Determining the number and location of the nodes for information processing, including nodes where the self-organization processes are possible to start;
- Develop or select a backup method for the system elements;
- Defining the connections between the elements of the system.

In the SODS simulation program the model is represented by two components: a graphical and a tree. The user has the option to add / remove a node, and the absence of cycles and presence of the system root are monitored (Figure 1).

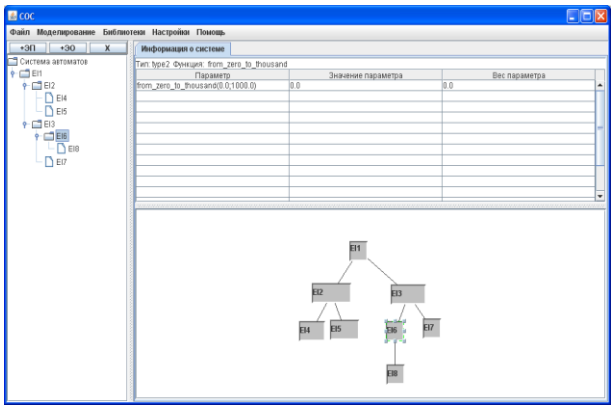

Figure 1 - Representation of the SODS tree in the simulator.

It will be appreciated that all elements of SODS can be classified as either elementary objects (EO) or an intermediary element (IE). For EO knowledge of its possible functions is needed; for IE at this time, moreover, it is needed to determine its communication capabilities: maximum number of objects that can be connected to it.

When we add new items to the system tree, for each object we should specify its name, type and function. The object type is selected from the predefined set of object types that can change in an arbitrary manner by the user and stored in the library. The type library contains information about the cost value of adding an element of this type, the maximum number of its children (for IE), and provides a set of functions that can implement this object. The function of the object is selected from a pre-determined set of features that can be modified by the user and contained in the function library. Function Library contains a list of all functions that can be implemented by any object in the system. Note that the elements of intermediary library function contain a set of so-called bundle of features that consolidate in some way the results of its child elements.

After specifying the types of all elements in the system and all links, the topology of the system is considered as fully determined.

Step 3. Selection or design of communication protocols for interaction between the system elements. This problem occurs in self-organizing distributed systems, information exchange in which is due to send and receive messages. Any system of this kind is characterized by the presence of various kinds of failures and malfunctions that results in overall coherence of the entire system likely to be at risk.

To prevent this kind of mismatch processes at every system element they should periodically save their state on a sustainable medium. This process is called "taking a local checkpoint." To maintain a consistency of the whole system, with inter-process communication it is needed to involve special communication protocols. The result of such protocol is to define a global control point (GCP) of the entire system - a common, consistent status of all elements forming the system. Upon the occurrence of failure in the system, its state is restored from the most recent GCP. The goal of this design phase is to select such protocol of taking control points, or create it. We will further imply that the protocol is already integrated into the communication subsystem, and therefore making a connection between two objects in the simulator, we believe that the problem of the system consistent state is solved.

Step 4. Defining the parameters of system elements. The function of each system element contains a set of controlled parameters. When we set these parameters, we should consider the following key points:

- All parameters must be normalized. This allows to avoid the implementation of parametric self-organizing algorithms of many computational problems, such as problems of the small difference of large quantities or accumulation of errors for small quantities;
- Qualitative assessment should be translated into quantifiable;
- For stochastic parameters it is necessary to determine what mathematical abstraction will be used: calculated mean value option or calculation for the worst case;
- The removal of restrictions on the parameters by changing the variables.

In the simulator for all the required parameters Parameter Library is used, so the user can modify and supplement any of them.

For each parameter, we could specify its name, range, initial value, and indicate that it may take part in self-organization processes.

Step 5. General self-organization concept development. After setting the system structure and all its components, it is necessary to determine what kinds of self-organization (SO) and in what sequence it can be run in the system for the implementation of the objectives laid down in it. A typical order of starting the appropriate mechanisms looks like: parametric SO → functional SO → structural SO, but it can be changed depending on the specifics of particular SODS. Another objective of this phase is to define a list of nodes that can run the mechanisms of self-organization over their descendants. Such parallelism gives a significant effect on productivity in systems containing large number of elements, where the information process in one central node is associated with a longer time and hardware costs. There is a need to develop criteria to judge the success of the ongoing processes of self-organization. The following three points should be developed in case of relevant type self-organization existence in the system. Settings allow the simulator to include / exclude any kind of self-organization, and change their order.

Step 6. Developing mechanism of parametric self-organization. Choosing a family of parametric self-organization algorithm depends on:

- The dimension of the controlled parameters vector;
- The kind of target system function;
- Required convergence rate of the algorithms;
- Acceptable amount of computation;
- Specificity of a particular task.

During the research two methods of parametric optimization were chosen: method of cyclic coordinate descent and exponential relaxation. This choice is due to the versatility of the algorithms and ease of preparation tasks for computation. Adapting the speed of convergence of implemented algorithms can be achieved using a variety of strategies that could significantly speed up the process of finding extreme, depending on the specific task.

Step 7. Developing mechanism of functional self-organization. If in the system functional self-organization is presented, the function is chosen from some set of functions. The recommended solution is to offer the system to use the results of past successful implementations of functional self-organization. In this case, the function selection is determined by its priority formed on the basis of the experience gained.

Step 8. Developing mechanism of structural self-organization. Structural self-organization is associated with the hardware cost, so its implementation should be approached very carefully. In the simulator three types of changes are implemented in the structure of the system:

- Changing types of elements in which there is no change in the system topology;
- Tree reconfiguration, in which the location of some elements is changed to more advantageous in terms of growing the target functional value;
- Addition of new elements, in which the system topology and it's composition of elements are changed.

Application of each of these three types of algorithms must be based on the particular system specificity and be economically feasible.

In the simulator for the purpose of determining the place to add new elements two parameters were introduced: the cost of adding the element of appropriate type, and the maximum number of children for IE. By varying these parameters in the proper range, it is possible to calculate the many combinations of the elements in the network system, to assess their appropriateness, and choose the most appropriate one.

Step 9. A series of experiments on the simulator. Once the model of SODS, identifying all the mechanisms of self-organization and the corresponding algorithms was built, it is possible to run the procedure of system time simulation. Program settings include the choice of modeling time scale and display the values of specific elements. In the

left part of the window the program displays the results of self-organization procedure with the corresponding type and the moment of time at which it was affected. At any time, the user can interrupt the modeling and change the structure of the model.

Step 10. The analysis of simulation results. Based on these experiments the developer can take the following basic solutions:

- A system copes with its task;
- A system copes with its task, but the degree of structural, functional or parametric change is too high;
- The system cannot cope with the task, or the degree of changes in the implementation processes for self-organization is unacceptable.

Depending on making judgments the developer has the right to change the purpose of the system, the parameters of the topology, can include/exclude the required forms of self-organization, or make a decision of inappropriate use of the system to solve the problem.

After the decision, all the previous stages could be repeated iteratively.

References

[1]. M.S.Kupriyanov, A.V.Kochetkov "Simulation of self-organizing systems.", *International Conference on Soft Computing and Measurements. St. Petersburg.,* pp. 57-60, 2008

[2]. M.S.Kupriyanov, A.V.Kochetkov "The use of parametric optimization techniques for self-organization of distributed systems", *International Conference on Soft Computing and Measurements. St. Petersburg.,* pp. 144-148, 2008