Applying TRIZ in Large Scale System Analyzing

Mehdi Akbari, QIAU Univ. PhD student in software engineering.

Abstract

Today, one of the most important issues in software engineering is developing large scale systems such as reservation systems. In general system development need to modeling, design and implement component and combine them depend on system properties. In this study, first a large scale system will be defined and then a new model based on TRIZ methodology to develop large scale and ultra large scale systems will be presented.

1. Introduction

Today the requirement of popular business systems can basically be solved by a large scale system. Therefore the need of a suitable processing model is of great import. However, unfortunately this problem has not given enough attention. In this article a processing model for analyzing of large scale system will be presented. To start with, the two main parameters of TRIZ and large scale system are being evaluated and analyzed. Later a processing model for developing of large scale system will be purposed. System developing includes some stages. The way these stages are related to each other depends on the system properties. In TRIZ the evaluation of these relation ships is performed through both system analyzing and integration.
2. Definition of large scale systems

The large scale systems have been defined by different researchers. Below some of these definitions will be presented:

In 2000 large scale systems were defined as systems with two properties mentioned below [1]:

- Complexity
- Distribution

In 2001 the large scale systems were defined as following[2]:

- Contain a large number of codes.
- The components have a high complexity and are closely related to each other.
- Mostly available components are being used.
- Several programming languages are being used.
- There are a large number of developers located in various countries.
- The data are distributed in files, relational data bases and object oriented database.
- The component distribution on different platforms.
- The ability of concurrent utilization of data bases and programs.

In 2002 these systems were defined as following[3]:

- Have many different sub systems.
- Have a large number data.

In 2004 these systems were define as following[4]:

- Have a large number of complexes tasks, which leads to an increase in number of codes. These numbers at least 0.25 millions line of code and reaches 50 millions line of code or more at its most. The number of program classes is in range of 1000 to 10000.
- Further it is suitable for non functional requirements.
- They demand large scale data such as Giga byte, Tera byte, and even Peta byte.
- They equipped with distributed processing and distributed data. These data ought to be distributed in more than 5 sites.
- The necessary of system integration due to the variation of hardware, software and connection system. In fact there are more than 5 different systems with should all be integrated.
- A large number of human resources are required. (100- 100000 person in a month). It takes a long time to arrange them.

3. Theory of invention problem solving
This theory is based on the principles by the Russian scientist (Altshuller) which includes 40 inventive and innovation principles. During his work time in an invention registration office since 1946, Altshuller gathered and registered several similar inventions which led to invention of new products. Among all information gathered by him 40 items were proposed as the main items in invention. The TRIZ was then suggested by him based on these 40 principles of invention. Today this theory has different applications in different scientific fields such as production, detection and solving problems in factories, modeling, and so on.[5]

4. Using TRIZ in large scale system analyzing

In software engineering, software can be developed using TRIZ. Many of these principles are being used in current methods. In this part, these principles and their application in software will be briefly discussed. (Worth mentioning that, here only those principles that are most related to software engineering will be discussed.) Later the possibility of using these principles in analyzing large scale systems will be studied.

4.1. Segmentation

Dividing a system into its atomic elements which is applicable in information systems and design data bases is also related to this principle. This principle contains of a large scale system made of many sub systems.

4.2. Separation, Extraction

This principle is widely used in compilers which are based on division of commands and recognition of its components. Further, in software used for pattern processing, each pattern is distinguished and recognized. In large scale system, the recognition of main part of the system is necessary.

4.3. Local quality

In computer network the sending data is converted into useable information while received by target computer. This means that the sending data which is not useable by other computers in the network will be converted into useable data in target computers. The data quality is different in different location. In large scale systems, each sub system has its own standard.

4.4. Asymmetry, symmetry change

An example for usage of this principle is in dynamic resource allocation in operating system and networking. Due to program prioritizing, the operating system allocates hardware and software resources. Further, the computers in networks have no symmetrically access to resources and utilize more resources based on their task. The absence of orchestration in large scale systems can be studies and also in ultra large scale the ecosystem property can be explained based on this structure.
4.5. Combining, integration

Computer programs are divided into several objects in order to be executed in parallel. This is to increase execution speed and hardware performance. In this way programs are first executed in parallel and then the final result is monitored at the same time. Integration in large scale systems is based on this property.

4.6. Universality, multi functionality

In this section developing systems that supports multi functionality is being discussed. For example, each user has his personal account and by entering the system PID for that user will be loaded and the system runs differently with respect to this property. Multi functionality can even be expected in large scale systems. In these systems each section (part) should be uniformed with environmental changes.

4.7. Nesting, embedding

In object oriented programming this principle is used for embedding of one object in another. Derivation of one object out of base object is also founded on the same principle. Since the general properties are defined and embedded in base object. This principle can be used in system integrating and sub system communicating in large scale systems.

4.8. Weight compensation

In computer networking the informal access of the users to the resources is an example of load balancing. One of the applications of this principle in development of these types of systems is making orchestration in large scale systems.

5. Conclusion

Due to large scale system properties, these systems can be used to solve integration and orchestration problem in small systems that we want them to integrated. In this way TRIZ can be used for finding a way to analyzed and large scale system implementation. In the other hand in ultra large scale systems that have ecosystem properties the main problem is sub system linking that can be solve with TRIZ principles. In this study, TRIZ and its application in software will be briefly discussed.
References


Mehdi Akbari Phd student in software engineering IAUQ University.

Some published papers include:

- TRIZ Oriented Artificial Intelligence TRIZCON2005
- Applying TRIZ in AI 1stIFIPWC-CAI 2005
- Applying TRIZ in Algorithm Design TRIZCON2008