

Utilization of Anticipatory Failure Determination for Debugging Software

Developing the debugging process

Creating lists of typical mistakes (bugs) made by software developers

Various lists of typical mistakes should be developed and structured according to:

- The program (module) type under development
- Stages of development
- Utilized shells, libraries, languages

Today such lists already exist, however, they have certain limitations. Since they are statistically based they are always “behind” and therefore cannot include new possibilities of failure brought about by new languages, shells, libraries, or other new software products utilized in the process of programming.

AFD recommends the utilization of a pro-active approach to creating lists of typical mistakes, that is, “inventing” (with the help of TRIZ) mistakes which might be made under various conditions, and then checking later to determine if these mistakes have manifested. This kind of work is most crucial for revealing conceptual and architectural mistakes, as these pose the greatest danger. This systemic approach and structurization, based on the Patterns of Technological Evolution, can significantly increase the efficiency of creating these lists.

Unveiling typical reasons and conditions of making mistakes

In reality, it is often difficult to identify why a certain mistake has occurred. TRIZ offers a method for “inventing” the mistake, along with the conditions that produced it.

Revealing signs of the existence of certain mistakes

Certain groups of TRIZ Operators (“Operators for Detection”) can be applied to reveal the signs that certain mistakes exist.

Creating specific lists of mistakes

For every software product, and even for every software developer, a list of mistakes that are typical for a particular individual should be created.

Development and debugging

Utilization of Patterns and Lines of Technological Evolution in software development

Today, basic Patterns and Lines of Evolution covering various technological systems have been developed. Also, the extensive experience of TRIZ professionals has shown that the development of software systems is governed by the same (or similar) patterns and lines (and some specific patterns/lines exist as well). Based on these patterns/lines, it is possible to control software development in the most promising direction, avoiding typical and painful conceptual mistakes. To assure successful results, additional specific lines applicable to software development should be created.

Utilization of problem-solving methods in software development

Often, a mistake is the result of an unsolved problem or contradiction. In many cases the problem is not even formulated. To unveil and formulate the problems related to development, various TRIZ tools such as the Problem Formulator, and universal and specialized Operators can be applied. Initially, the existing tools can be utilized (with some adaptation); then, specific tools applicable to software problems should be developed.

Utilization of the AFD approach at each stage of software development

Every stage of development should begin with an AFD session to unveil the undesirable effects that might occur, “invent” all possible mistakes, evaluate their severity, and create a plan for avoiding them. At the end of each stage another AFD session should be performed to unveil and fix all the mistakes made during this stage.

Utilization of problem-solving methods to fix mistakes

After a software program or module is ready, the mistakes must be revealed. For this purpose, specialized Operators aimed at detection and measurement should be utilized. ARIZ should also be utilized, especially in the most difficult cases. When a mistake is unveiled, the problem becomes the following: how can this mistake be fixed without introducing new mistakes? And here we have a contradiction: It is necessary to make changes, but changes must not be made in order to ensure there are no mistakes. To address these contradictions, TRIZ tools and approaches can be used.

Identifying reasons for the erroneous behavior of software

Sometimes, the reasons for mistakes in software are unclear. In these cases, the Failure Analysis module of AFD, usually applied to solve scientific problems, can be applied. The inverted problem formulated with the help of the Failure Analysis module forces one to look for resources capable of providing the existing (and erroneous) software behavior.