TRIZ for Software (T4S) Initiative

Update (April 20, 2005)

Kevin C. Rea
kevin@trizforsoftware.com

The concept of applying TRIZ to software problems has created some stir – which overall is not a bad thing. Question: TRIZ is designed for technical systems, so is software now a technical system? I believe that depends on the context and your view of software. Regardless, with the complexity and integration of software into so many products, methods for improving software by applying TRIZ and/or other systematic innovation methods is worth the research effort.

It is a fair observation that software engineers often model their software after physical systems in nature. As this trend continues, we should expect to see various natural systems implemented in software algorithms and the like. Have we seen this? Yes – for starters, genetic algorithms, as well as entropy have been used in software routines. Also, there have been articles written that both support and reject the notion of applying TRIZ to software problems; as such, there is a need for "open" research to investigate this.

Regardless of our views on T4S, companies are starting to ask whether TRIZ can be applied to the design and creation of software. We should be able to answer that question with as much knowledge, detail and experience as possible.

1 WHAT T4S IS

Basically, the vision of the T4S initiative is to see what, where, and how TRIZ can apply to software problems minus any *destructive* criticism. It has an "open" structure similar to the open-source software initiative, but with an accompanying creative commons copyright.

The goals of T4S are:

1. Investigate different lines of thought concerning the application of TRIZ for the software domain;
2. Produce open-source tools and resources that support TRIZ, and in particular T4S;
3. Produce articles and conference reports from the T4S discussions that will advance T4S and TRIZ in general.
4. Secondarily, look at other methods of innovation and quality improvement in their relation to TRIZ and application to the software domain;
5. Not to intimidate, but to edify and promote personal and professional experimentation of T4S.

To date, we have several resources, investigations and papers that are currently in the works, or will be soon:

1. Looking at ways of re-building certain parts of TRIZ and/or abstracting relevant notions (keeping TRIZ beneath pure);
2. XML[1] schemas for the 40 principles and matrix along with corresponding data files for software, and testing;
3. 'TRIZ for Testing' investigation;
4. Contradiction prevention and avoidance in T4S;
5. Ideality in the software context;
6. Research in applying Su-Field analysis in software engineering/modeling;
7. Subversion Analysis / Anticipatory Failure Determination (AFD);
8. More lines of thought being added almost weekly.

Our hope is that others will join us in our efforts - especially those that are actively applying TRIZ to software problems.

Putting this subject on the back-burner and saying it can't be done isn't good enough – we need to see where it *can* apply and this requires a sincere and joint effort. ***Ideally***, we want to make software better using TRIZ where applicable. Is this a worthwhile cause? Time will tell.

You can join the TRIZ for Software Initiative mailing list by going to: http://trizforsoftware.com/ then click on the Internet links page. You'll then see the link for the mailing list. At present, we have over thirty members and growing.

---

[1] eXtensible Markup Language - A widely used system for defining data formats. XML provides a very rich system to define complex documents and data structures such as invoices, molecular data, news feeds, glossaries, inventory descriptions, real estate properties, etc. As long as a programmer has the XML definition for a collection of data (often called a "schema") then they can create a program to reliably process any data formatted according to those rules.