TRIZ and the evolution of CAPE tools
From FLOWTRAN® to CAPE-OPEN¬® and beyond

Bertrand.Braunschweig@ifp.fr
Institut Français du Pétrole, France

Kerry Irons (ironsk@dow.com)
The Dow Chemical Company

TRIZ

Last century, Genrich Altshuller, a Russian engineer, analysed hundreds of thousands of patents and scientific publications. From this analysis, he developed TRIZ, the theory of inventive problem solving, together with a series of practical tools for helping engineers solve technical problems. Among these tools and theories, *The 40 Principles* describe common ways of improving technical systems. For example, if you want to increase the strength of a device, without adding too much extra weight to it, TRIZ tells you that you can use principle 1, segmentation, or principle 8, counterweight, or principle 15, dynamicity, or principle 40, composite materials. TRIZ is now used by a wealth of Fortune 500 companies in support of their innovation processes.

When Atshuller developed TRIZ, he could not think of software components. These objects just did not exist. But software are technical objects too. They are in fact some of the most complex technical objects produced by man. Many of the TRIZ tools and theory elements only relate to concrete objects (e.g. principle 11, prior counteraction, or principle 32, change the colour). But some of the principles can be applied to software. Among these we outline principles 1 and 15.

Principle 1, Segmentation, shows how systems evolve from an initial monolithic form into a set of independent parts, then eventually increasing the number of parts until each part becomes small enough that it cannot be identified anymore, such as in a powder. Further evolution based on this principle leads to similar functions obtained with liquids, gases or fields. Think of a bearing with balls suspension, replaced by microballs, then by gas suspension and finally by magnetic field.

Principle 15, Dynamicity, introduces flexibility and adaptation by allowing the characteristics of an object, of an environment, or of a process, to be altered in order to find an optimal performance at each stage of an operation. Think of a traffic light that adapts its period depending on the traffic.

The evolution of CAPE tools

The first CAPE software developed in the sixties and seventies e.g. FLOWTRAN were large monolithic systems. They remained as such until recently when developers started to cut those systems into smaller pieces that would fit together. Modularity, object-oriented programming, component software, n-tier architectures are the current paradigm for CAPE software development, and can be considered as the second stage of evolution. The CAPE-OPEN interoperability architecture, based on object orientation and middleware, is the best representative of this stage. The third stage will be the one of dynamicity, as the needs for self-adaptation become increasingly important, in order to match the increasing diversity in usage. Self-adaptation can be obtained using current software technologies, such as JINI or Enterprise Java Beans, that allow software components to discover their environment at runtime and to seamlessly integrate within these environments. But this is not enough. Another dimension of self-adaptation is that software should behave correctly when faced with new situations, new data, new usage modes.

Beyond CAPE-OPEN

Current architectures, even though they allow distributed computing on heterogeneous hardware platforms, share the same paradigm for control and co-ordination: a central piece of software controls and coordinates execution of all software modules and components that together constitute the model and the solving mechanism of a system. One example is the central piece of software that is usually called "simulation executive", or "COSE" in CAPE-OPEN architectures. Its tasks are numerous: it communicates with the user; it stores and retrieves data from files and databases; it manages simulation cases; it helps building the flowsheet and checks model topology; it attaches physical properties and thermodynamic systems to parts of the flowsheet; it manages the solving and optimisation algorithms; it launches and runs simulations; etc. All other modules (e.g. unit operations, thermodynamic calculation routines, solvers and optimisers, data reconciliation algorithms, chemical kinetics, unit conversion systems etc.) are under control of the simulation environment and communicate with it in a hierarchical manner, as disciplined soldiers will execute their assignments and report to their superiors.

COGENTS

Future CAPE tools will involve distributed architectures based on multi-agents technology where control and co-ordination are decentralized. Instead, each piece of software, each module, each component, generically called "agent", lives its own life, is able to negociate and co-ordinate with other components in order to solve problems such as process design, fault diagnosis, or supply chain management. The second part of the paper will show how CAPE-OPEN interfaces can be extended towards decentralised architecures of adaptive process modeling agents nicknamed "COGENTS" (CAPE-OPEN Agents). These adaptive cogents will implement the third stage of CAPE tools, following the evolution process defined by Altshuller in TRIZ.

References

TRIZ : The Right Solution at the Right Time. A Guide to Innovative Problem Solving. Yuri Salamatov, Insytec B.V., 1999

40 Principles : Triz Keys to Technical Innovation by Genrich Altshuller, Uri Fedoseev, Steven Rodman (Translator), Lev Shulyak (Translator)

www.colan.org CAPE-OPEN Laboratories Network web portal

B. Braunschweig, R. Gani (editors), Software Architectures and Tools for Computer Aided Process Engineering, Elsevier (in print).

Keywords

TRIZ, interoperability, CAPE-OPEN, Software Components, multi-agent systems, adaptation, cogents.