ISG

Instantaneous Software Generation

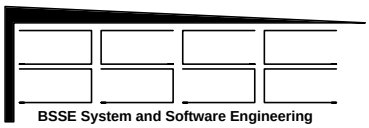
- A Brief Introduction and Survey -

Rainer Gerlich

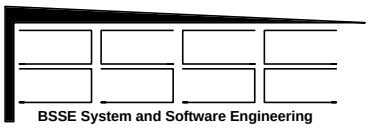
BSSE System and Software Engineering

Auf dem Ruhbuehl 181
D-88090 Immenstaad

Phone: +49/7545/91.12.58
Mobile: +49/171/80.20.659
Fax: +49/7545/91.12.40
e-mail: gerlich@t-online.de
www: http://home.t-online.de/home/gerlich/



V 03.08.2000



Instantaneous System and Software Generation (ISG)[1]

- A Brief Introduction and Survey -

Rainer Gerlich
BSSE System and Software Engineering

Auf dem Ruhbuehl 181, D-88090 Immenstaad
Phone: +49/7545/91.12.58 Mobile: +49/171/80.20.659 Fax: +49/7545/91.12.40
e-mail: gerlich@t-online.de internet: http://home.t-online.de/home/gerlich/

ISG PURPOSE

The purpose of ISG ("Instantaneous System and Software Generation") is process improvement. ISG aims to introduce full automation into the development and maintenance process in order to make it less risky and more efficient and to shorten the time-to-market.

ISG STATUS

ISG has already been applied successfully in the area of distributed and real-time software systems (embedded systems). The goal of this paper is to highlight the major features of ISG from a general point of view and to discuss the relevance of the ISG approach for application domains other than embedded systems.

ISG provides a degree of automation which is well known from other engineering disciplines like mass production of cars, electronic parts and equipment, but until now is missing in the area of software development.

STATE-OF-THE-ART

State-of-the-art software development is still based on a lot of handcrafted work which needs to close the gaps which are left by the tools, or to modify the output of a tool such that it can be processed by another tool. Consequently, tools can cover only a small part of the development process and the savings which can be achieved are rather low compared to the required investments, also regarding training in methodologies and tools..

An example: tools for test automation exist but the inputs must be prepared manually and the evaluation of test results is done manually, too. Therefore only a small part of the complete testing process is covered by automation and a large part still has to be done manually.

Moreover, the tool support remains on an informal level only, supporting mainly documentation of existing ideas, but giving no real feedback from practice. Currently, no such tool guarantees that the processed and accepted inputs can be successfully converted into an executable system. Therefore the risks still exist up to a rather late phase when testing and integration is started.

Also, current methodologies are still oriented towards manual processing. E.g. in the object-oriented technology a lot of manual changes are needed to specialise an existing class for its use for another application, while ISG writes all its classes automatically without any human intervention.

THE ISG IDEA OF AUTOMATION

The idea of ISG is to automate the complete lifecycle based on a minimum of inputs for specification and design.

The ISG framework which implements the process model can be compared with a spreadsheet which processes some inputs, derives figures from the basic inputs and presents them in a graphical manner. Whenever a part of the input is changed, the dependent figures and graphics are updated immediately.

Similarly, ISG approaches the development process. It identifies for a certain application domain the minimum inputs, the steps needed to produce the desired software output and the dependencies between inputs and the required steps. Consequently, whenever an input changes all the software is automatically changed accordingly.

This capability allows for incremental development and continuous maintenance right from the beginning.

In addition, the automated generation allows for instrumentation by which the properties of the software can automatically be checked and presented in either textual or graphical form.

THE MOTIVATION

The driver for ISG was risk reduction: to get the capability to confirm early in the lifecycle that specification and design are feasible and the system to be developed will have the desired properties - from a behavioural, functional and performance point of view. This step is called "Early System Validation"

Soon it became clear that this goal can only be achieved when the development process is automated, otherwise the overhead would be too high, possibly much higher than caused by the late changes required in case of the conventional approach.

A number of projects were executed until the principles of automation were sufficiently identified and the first project could be run based on a fully automated process model.

ABOUT INNOVATION

Automation and incremental development are the major corner stones of innovation related to ISG. Automation is a pre-condition for efficient incremental development.

Incremental development requires a different understanding of specification, requirements and design than it is state-of-the-art. To be efficient, a coherent transition from specification to design must be possible, this is what is currently considered as impossible by most software engineers. However, ISG guides an engineer to smoothly navigate between specification and design and leads to "Executable Specifications" and "Executable Design".

THE ORGANISATION

To apply ISG to a certain application domain following principal steps need to be executed:

- identification of the minimum engineering inputs
  from which the system can be generated
- the appropriate representation of such inputs
- the steps through which the system can be built
- the dependencies between the inputs and such steps.

The inputs need to be given in a formal manner so that they can be subject of checks. Consequently, this allows to guarantee the conversion of the inputs into an executable system. To make it easier for a user the degree of formaility should be hidden by embedding the formal inputs within natural language or another representation a user is familiar with.

Only once, for each application domain the organisation scheme has to be defined, then it can be reused and continuously be improved. Experience is already available in the area of distributed real-time systems and databases from an already completed space project.

IMPACT ON QUALITY

As a formal and automated process is applied to generate the software, the quality of the produced software is well defined, can be measured and controled by instrumentation and result evaluation. Last but not least the automated process model is fully reusable and can continuously be improved.

In fact, the rules which define the generation process are always exactly followed because an automaton does execute them. Also, an automaton does the evaluation of the results and the assessment of the system's properties. Hence, the control of quality is completely independent and repeatable and always the same for applications in the same domain.

ISG identifies the needed tests automatically from the user's inputs and stimulates the generated system automatically for nominal and stress testing and fault injection. The automated derivation of test cases allows to cover the full test domain or to identify that the system is not completely testable due to a huge number of test cases which cannot be executed within a given time. This improves quality because results are made available over all of a system's properties regarding behaviour, functionality and performance, and test case definition is not biased by an engineer's view on the system.

ISG AND THE CAPABILITY MATURITY MODEL

The "Capability Maturity Model" from the Carnegie-Mellon Software Engineering Institute defines five maturity levels of software development.

CMM level 3 ("Defined") is defined as:

The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organiszation. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

CMM level 4 ("Managed") is defined as:

Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

CMM level 5 ("Optimized") is defined as:

Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

To conclude, the idea of ISG perfectly fits with what is understood as "CMM level 5" implying the coverage of the needs of level 3 and 4 as well..

ACHIEVEMENTS

ISG has been successfully applied in the domain of distributed, real-time systems for a system planned to fly on the International Space Station in 2001/2002, of which the software is now completed. About 70% of the project's software are based on ISG, the remaining 30% are firmware which was excluded from the contract. The parts which are covered by ISG are: scheduling, message communication, command dispatching, exception handling, database handling, data processing and monitoring, telemetry handling, instrumentation and verification and validation.

A report on the properties of the system is available after about 90 minutes (on an UltraSparc I/143 or PC-200 / Linux) by generating the system from scratch and only taking the high-level system engineering inputs.

There are two more exercises going on in the course of the ESPRIT project 25514 "CRISYS" in the area of automation (mail sorting and distribution) and nuclear power plants (back-up power supply).

The applications are immediately available on a number of platforms: any possible combination of Unix (Solaris, Linux) and VxWorks with Intel and Sparc hardware. Parts of a systems even may run on different platforms. To generate a certain configuration just literals and figures have to be given or changed. More platforms can be supported.

OUTLOOK

It is planned to apply the ISG concept to larger database and client-server applications using SQL and Java and (extended, adapted) UML for user inputs.