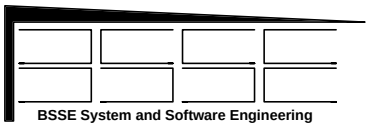
Dr. Rainer Gerlich



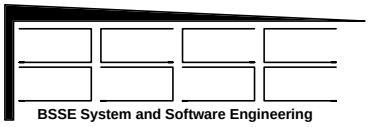
ISG and Software Quality

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/



Dr. Rainer Gerlich

BSSE System and Software Engineering

Experience with a Process Model and a Toolset for a Fully Automated Software Lifecycle and Its Impact on Quality Control

1. INTRODUCTION

In a number of industrial areas like automotive industry quality has been improved and costs have been reduced when moving from handcrafted to automated production. In the area of software development we have little of such automation, but still a large number of manual development steps. This is because the current tools only concentrate on the support of the conventional steps like speciifcation, design, coding, testing and integration of the lifecycle rather than on an automation of the whole lifecycle. They do local, but no global optimisation of the development process.

Currently, a system's architecture and its components are defined individually for each project, which makes it difficult to optimise the development process towards higher quality and efficiency. Even in case of object-oriented development the reuse is limited to classes and methods. More general rules for systematic construction of software are not supported, which e.g. could cover structural differences on nested hierarchy levels.

The approach for "Instantaneous Software Generation and Validation" (ISG) overcomes the weakness of the currently applied software development process described above. It introduces standardised interfaces and components and clear rules for construction of software. Its current application domains are distributed and/or real-time systems, including embedded and client-server systems.

This paper discusses the aspects of ISG regarding built-in quality, quality improvement and quality control.

ISG generates a skeleton of a system which covers behaviour, performance, communication, topology and fault tolerance and provides drawers into which the functional software can be plugged-in by standard interfaces. This functional software can either be automatically generated by ISG as well, seperately be developed by a conventional approach, or it may already exist like in case of legacy software. In latter case ISG provides a solution to efficiently port existing software to modern platforms by automated generation of a new infrastructure into which the existing functional software can be plugged-in.

ISG is an outcome of activities on improvements of software development methodology initiated in 1992 by the European Space Agency. Since that time the approach was continuously enhanced due to the feedback from a number of projects, e.g. in the area of space technology, air traffic control and telecommunication. Based on this experience the concept of ISG was defined and the decision for implementation of a toolset was made by end of 1998.

Although just recently defined and implemented by a toolset, ISG has been successfully applied to a space application which is the Material Science Laboratory of the European Space Agency planned to fly on-board of the International Space Station in 2001. This is a rather complex embedded application consisting of 38 process types and 48 process instances distributed over two processors, and about 350 states, and accepting about 150 external commands.

ISG expects as inputs a formal definition of the system processes and the communication channels (including the type of fault-tolerance), the behaviour (by Finite State Machines, FSM) and data exchange between the components, and the performance constraints (timeout, deadline, range of

expected CPU consumption). These inputs are expressed by literals and figures. From these inputs ISG automatically generates the software, the compilation and the run-time environment. It provides utilities for automated testing including stress testing, fault injection, automated report generation and evaluation.

No code needs to be written in a programming language for generation of the ISG part of a distributed (real-time) system. Current experience shows that not only the skeleton which represents behaviour and (real-time) communication can automatically be generated, but also data processing software like needed for data acquisition, limit checking, data calibration, telemetry handling and command processing.

In fact, ISG takes advantage of formalisation, without compromising, but increasing the efficiency. Formalisation is a pre-condition for automation, (1) the automation of the development process and (2) the automation of quality control of software. And (2) is a consequence of (1).

2. IMPACT ON QUALITY CONTROL

Automated generation of a complete system environment from high-level engineering information helps to improve quality because the generation process itself and the output from the automatically generated system are subject of quality control, and not the generated software itself.

In the conventional approach the complete system is constructed by human beings partially supported by tools, therefore the quality control must concentrate on all of the mostly manually generated software. No part of the system can be considered as already verified. Even in case of reuse there remains a risk that the reused part will not work as expected in another environment as the Ariane 5 accident showed some years ago. This raises the need for intensive manual testing and quality control.

In case of complete automation of the generation process this process can be verified and certified like in automotive industry or other areas. The ISG process model ensures that (1) the outcome of the process will be compliant with the agreed quality requirements and (2) the deviations can be detected. Due to the reuse of the "generation and validation process" all effort on improvements can be concentrated on the software and organisation involved in the generation process. This reduces the costs and gives a perspective on a deterministic increase of quality. Another important consequence is the significant reduction of time-to-market.

In view of (1) the reuse of construction rules ensures that a system is built properly under well-known conditions. Improvement of quality concentrates on the process model and not on the (manual) implementation steps.

Concerning (2) the environment automatically generated by ISG provides the means to control the quality by supporting

1. formal checks at pre-run-time on

   behaviour/FSM, data exchange, distribution, exception handling
2. implementing checks at run-time on
   a. compliance of received inputs
   b. monitoring of timeouts and deadlines
   c. occured exceptions
3. automated analysis of coverage and performance figures at post-run-time.

The built-in checks (which can be switched off by a configuration file) are application-independent. They check the properties of a system which are defined by the engineer by criteria which have been brought into a normalised form like: "coverage of states and actions = 100% or not", "deadline met or not", "timeout occured or not".

For the system performance like CPU utilisation (on each node) or network utilisation the measured figures are presented and a criterion can be defined so that the evaluation utility can decide whether the requirement is fulfilled or not. If a criterion is not fulfilled, e.g. in case of a coverage less than 100%, all the non-covered states or actions are documented. More information is provided on resource consumption, like maximum number of elements used in a buffer, number of execution of actions, which allow an engineer to tune the system and to demostrate that the constraints on the sizing and timing budgets are or shall be met.

Another means to document and control the quality is the automated set-up of (black-box) testing based on the specification of the inputs and the Finite State Machines. ISG supports automated stimulation of each of a system's processes for a given user-defind load. Just by adding the name of the process to a list, the respective process is automatically stimulated. The same way a process can be excluded from stimulation while all other process remain subject of testing. Similarly, fault injection can be initiated. This allows to easily test the system under stress and fault conditions.

Hence, a system can be executed and tested according to

1. its operational profile
   as defined by its behaviour and the possible inputs,
2. a stress testing profile
   for which e.g. all processes are automatically stimulated at high input rates
3. a fault injection profile
   for which certain conditions for fault injection (like loss of data) are defined.

Of course, modes (2) and (3) can also be combined.

The stress testing and fault injection profiles will document how the system behaves under exceptional conditions. E.g. it may be shown how the system behaves if the probability to loose data is increased. It was observed that there is a sudden decrease of the coverage figures at a certain fault injection probability. This documents the robustness and fault-tolerant properties of a system independently of its specific shape.

Hence, like for the operational case ISG also provides for stress testing and fault injection an application-independent method to measure and control the properties and the quality of a system.

The generic approach for evaluation of a system's properties allows engineers (or an automaton), which are not involved in the system definition process, to identify deviations from the expected properties and quality. Therefore implementation of an application-dependent oracle is not required, if any is known at all.

Regarding ISO 9000, only the processes for system generation and quality control would be a matter of certification. Obviously, ISG is completely in-line with the idea of ISO 9000, the certification of the development and quality assessment processes. As the processes are executed without human intervention and the results are automatically documented this makes it easy to prove the compliance with ISO 9000 certification requirements: a system is built in accordance to well-known rules and it is controlling and documenting itself the properties of the generated system.

The certification of the ISG process should be a matter of future activities in this area.

3. CONCLUSIONS

First results obtained from projects which were using the completely automated lifecycle show that the automation of the software development process opens the possibility to automate quality control as well.

This control of quality not only addresses principal formal checks as applied in the past e.g. by tools supporting semi-formal or formal checks, but all checks related to the implementation of a system in its environment. Such enhanced checks analyse e.g. the usage of resources and system services on the target. They help to close the gap between the formal, theoretical checks outside the target environment, just saying "the system can correctly work from a theoretical point of view" and the real target environment which decides at the end.

Currently, it has been observed that a number of checks can be brought into a "normal" form which is application-independent and allows for automated evaluation. Nevertheless, there is still a lot of work to do to bring all needed checks in such a form. But the feedback obtained from the first implemented automated checks indicates that it is worthwhile to continue this work.

The work on ISG was funded in part by the ESPRIT project 25514 "CRISYS" (Critical Instrumentation and Control Systems). In this project BSSE is partner of Schneider Electronique (prime, France), Elf (France), Verilog (France), University of Grenoble (France), CEA (France), Prover Technology (Sweden), GMD (Germany), Siemens ElectroCom (Germany).

OBJECTIVES

- formalisation of the development process by use of inputs and rules at high abstraction level defining behaviour, communication, topology and fault tolerance
- allow to update and change the engineering inputs until the "last minute" without creating a lot of effort, costs and delay in schedule
- when the user inputs are accepted ISG guarantees that they are complete and correct and the system is executable
- when the automatically generated report does not indicate an anomaly or exception the generated system complies with its specification
- higher quality by automation of the software production process and automated testing, stress testing and error injection
- control and documentation of quality by automated built-in checks and reporting facilities on behavioural and performance properties which are application-independent and reusable to a major part
- reduction of development risks by immediately available coverage figures and timing, sizing and network budgets
- higher degree of reusability by program construction rules as generalisation of class concept and inheritance
- higher stability against changes due to decoupling of behaviour, performance constraints, topology and fault tolerance