Downscaling Generation, Verification and Validation of Software into the Range of Hours - The Benefits of Complete Formalisation and Automation

Abstract

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/

BSSE System and Software Engineering

BSSE System and Software Engineering

Downscaling Generation, Verification and Validation of Software into the Range of Hours - The Benefits of Complete Formalisation and Automation

Rainer Gerlich
BSSE System and Software Engineering

Auf dem Ruhbuehl 181
D-88090 Immenstaad, Germany

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/

Abstract: Usually, the process of software generation, verification and validation takes weeks, months, in some cases possibly years. By applying a completely automated development procedure the turn-around time required for an iteration can be scaled down into the range of hours. Such an iteration cycle starts with delivery of (preliminary) system engineering information and ends with reading of the evaluation report. Everything what happens between delivery of the information and result analysis is automated.

The high reduction of development time and related costs and effort is achieved by the use of a process model which allows complete automation of all of the required steps. This is different from the currently applied development procedures and process models which still require a significant manual contribution which prevent short turn-around cycles.

Moreover, due to the manually executed steps inconsistencies may occur or - vice versa - the effort to avoid or to remove detected inconstencies is rather high.

This is different for a completely automated process model which cares about such issues itself. The philosophy of the automated development environment called ISG (Instantaneous System and Software Generation) is to take the user inputs and to check them for correctness and completeness.

Having accepted such inputs the tool transforms them in executables, distributes the executables to the given CPU's of the network (which may include different platforms), executes them and establishes an evaluation report on the system's properties.

The generation process also considers instrumentation for analysis of the system's properties and automated test stimulation including stress testing and fault injection. The analysis report not only provides figures on the logical aspects like data flow and state coverage, but also performance and resource, sizing and timing aspects.

Due to the short turn-around time incremental development is possible. A user may start with very little information and then refine his view on the system getting always immediately full feedback on what he has defined. This allows to validate a system right from the beginning and continuously until the very end.

The achieved high degree of automation is based on formalisation. Firstly, the user inputs are subject of formal checks. Secondly, the system software is built according to formal construction rules. Thirdly, the generation steps are based on a formal process model which can be completely automated.

On this formalisation verification and validation techniques are based on and it is even possible to define application-independent checks of the properties. An important consequence is that the automated evaluation of system properties can give answers without being aksed for. This is an advantage compared with other verification approaches for which the engineer has to ask the right question. If he doesn't do it he never will be informed about potential problems.

The current application domains which are covered by ISG are (hard) real-time and/or distributed systems. Basically, by ISG the compete infrastructure of such systems is generated automatically, i.e. ISG provides a drawer into which the application-specific processing algorithms may be plugged-in as user-defined functions (UDF). At the beginning ISG provides (instrumented) stubs for the UDF's which are replaced incrementally by the real functions.

The recently executed projects showed that the automated generation of UDF's is also possible for a number of application areas.

The engineering inputs as provided by the user define the components of a system like processes and the network, the internals of the processes like states and expected inputs, the UDF's for processing of the inputs, and the outputs (if any) to be sent as response. Furthermore, the CPU needs to be given on which a process shall reside, also the expected consumption of CPU and the (estimated) amount of data to be sent. All the inputs are given in terms of literals, no code has to be written.

For a space project the user was familiar with spreadsheets. Therefore most of the inputs were defined by spreadsheets. However, it is possible to adapt the format of the user inputs to the user's needs.

Although a spreadsheet does not look very formal it is possible to formalise such inputs due to a meta-model which correlates the various types of entities. This is the same for other input formats even if they are based on plain text.

The space project consists of about 40 (real-time) processes distributed over two processors. It takes about 45 minutes to generate the complete system from the engineering inputs (spreadsheet), another 30 minutes to execute the system until a coverage of 100% of the input dowmain is achieved and another 5 minutes to establish the report. Recompilation of all files requires about 10 minutes. An assistant tool will make the decision automatically when to start from scratch again e.g. in case of a structural change or when just to do recompilation.

ISG is currently available for Sparc and Intel hardware, for Solaris, Linux and VxWorks operating systems. To move from one platform to the other just requires to change an option in a configuration file.

A (heterogeneous) distributed system is automatically installed at each node and started across the network.

The topology may be changed by redefining the mapping between processes and nodes (which represent a CPU from a logical point of view) and the nodes and the real CPU's. This is very efficient and allows to easily evaluate different network topologies.

The paper will also discuss problems and issues related to a completely automated installation and distribution procedure and its portability.

Keywords: automation, formalisation, incremental development, early system validation, distributed systems, process models