Quality Assurance of Models for Autocoding

Ann Cass, Pierre Castori

SYNSPACE AG
Hardstrasse 11
CH - 4052 Basel
ac@synspace.com, pc@synspace.com

Abstract: Automatic Code Generation is an emerging technology that holds great promise for future generations of embedded systems. However, ACG raises many fundamental questions about model and code quality. A series of experiments promise to provide an enhanced approach to Software Quality Assurance in order to allow autocoding to become a trusted industrial solution for on board space software.

1 Introduction

In the recent years, the automatic generation of source code based on a model of the system or software (autocoding) has received increased attention from the embedded software community. In fact, modelling and autocoding are now among the main challenges for improving the development of safety-critical embedded software. Impressive benefits have been reported by the companies who have played a key role in the industrial use of this technology, such as Airbus, BMW and Eurocopter: Cost reduction up to 50%, shorter lead times through executable specifications, (almost) immediate availability of changes in the software, etc. [Ca02, EST, Ba03].

The space industry is taking note of these benefits and several projects have studied the technical feasibility of autocoding the control software that flies on board satellites [TAV99, Co00, OSA]. Modelling is currently used in the space industry for mission feasibility studies and for describing and simulating the behaviour of algorithms – in particular for Guidance, Navigation, and Control (GNC) systems. However, up until now, these models have rarely been used further in the development process. Autocoding opens these possibilities but requires a greater understanding and control on the quality of source models and generated code than "classical coding". In addition, modelling and automatic code generation have a strong impact on the development process and require new or modified practices. These difficulties need to be tackled and solved before autocoding becomes a 'standard' way of working in the industry. Such challenges are faced by other safety critical embedded domains as well such as aerospace and automotive [ET03, BM02, HS01].

The European Space Agency (ESA) has identified modelling and autocoding as cornerstones of its technology roadmap for the development of spacecraft over the next decade in the programme "European Harmonisation in On-Board Software" and has

launched a project "Automatic Code Generation"1 (ACG) to demonstrate the possibility of using autocoding for safety critical embedded systems.

A series of experiments will provide detailed insight into the impact of modelling and autocoding on the specification, design, coding and validation phases. These experiments (performed by space industrials and described in more detail in [CZ04, ZL05]) investigate the feasibility of ACG by autocoding parts of software already developed for previous space missions. The experimental situation is rich as the original software and the project validation facilities are still available for use and comparison. In addition, the history of the original mission projects (e.g. time and effort spent, changes to requirements, etc) is available and provides a solid baseline for observing differences in the autocoding development process. The experiments use some of the ACG tools leading in the market today (dSpace TargetLink, ESTEREL Technologies SCADE, Telelogic Tau G2). They will identify potential problems or risks with modelling and autocoding and support the definition of suitability criteria for autocoding environments to be used in the production of embedded systems. Finally, the project will provide constructive guidelines that enable the effective use of autocoding.

Meanwhile, SYNSPACE is responsible for the Quality Assurance and Safety aspects of the experiments. As modelling and code generation affect a variety of software engineering activities, the corresponding QA processes must also be adapted to ensure the fulfilment of quality requirements. Our focus in the study is thus twofold:

- To assess whether Automatic Code Generation can be successfully used for producing safety- or mission-critical software,
- To assess and advise what changes should occur to quality assurance processes (and therefore the relevant space standards) to incorporate ACG.

2 Adapting the QA Process for Modelling and Autocoding

Defining a consistent strategy for Quality Assurance in an ACG lifecycle represents a significant innovation, as ACG affects many areas of the software development process and raises fundamental issues about quality. New QA activities must be identified to ensure that quality requirements for the *product* and *process* are fulfilled. For the ACG project, SYNSPACE has identified a number of modifications to the traditional space QA process. These are represented in the form of changes to ECSS-Q-80 [EC03], the European space standard for software QA. All proposed changes remain to be confirmed during the experiments, but a few are described in this section.

In terms of *product* quality assurance, ACG raises fundamental questions about the importance of certain "classic" *code* quality attributes such as readability, complexity and modularity. In addition, the quality control of the autocode *model* takes on an important significance. Quality properties must be defined and verified for models that

[1] This work is supported by ESA/ESTEC Contract No. 18056/04/NL/JA. The ACG project consortium includes EADS ST, SciSys, BSSE, and SYNSPACE.

ensure the quality of the code generated. This leads to a new focus on measures of model quality and modelling standards, described in the sections below.

2.1 Measures of Model Quality

QA must establish the expected quality characteristics both of the final product and of intermediate work products (specifications, designs etc). For traditional manual development, a defined set of quality attributes exists for source code. Indeed the current quality standards applicable for the development of European space software [EC03] contain a number of requirements on attributes which represent code quality, but no such quality requirements are defined for models used for autocoding in industry. It is not even clear to what extent it is possible to define generic quality attributes that are independent of the specific modelling languages. In the study, a first examination is to be performed, seeing if certain measures associated with code quality (e.g. McCabe complexity) can also be adapted to apply for autocode models, or at least if certain measures can be defined to control the generation of code.

2.2 Modelling Standards

A common approach to ensuring higher quality autocode is to restrict the use of the modelling language. This has been observed as an effective means of improving performance by removing constructs that are not efficient for the target compiler [Co00]. Code size can be decreased by 30% using very simple rules such as declaring that only one instance of a process is authorised at execution time. The declaration of procedures as inline is another such possible rule [MO98]. This shows that modelling for autocode encompasses more than just defining a correct model of the system or software. Tailoring of the model and configuration of the tool used to produce code is also a key part. Therefore a complete set of rules specific to the modelling language and tool used (called a modelling standard) must be defined. In [Ba03], experiments made at the BMW Group show a reduction of 55% in size and 77% in execution time of the generated code just by using a set of 70 rules at the modelling level. [TAV99] considers that appropriate design rules might be needed to achieve a correct level of numerical accuracy. Some authors even believe that the main obstacle preventing widespread acceptance of ACG in safety-related projects is the lack of industry-wide definitions and modelling standards against which models can be verified [Fr03]. One notable exception is the MAAB Style Guide, a modelling standard in the automotive domain [MA01], which includes rules to prevent inefficient or unreadable code as well as errors in the code or with code generators and compilers. In the ACG experiments, modelling standards will be applied by the projects for the different modelling languages used. These standards will be refined based on the result of code quality analyses as well as validation testing.

2.3 QA of the Modelling and Autocoding Process

In terms of the quality assurance of the development *process*, ACG introduces a number of issues including the need to assess the translation between the model and the code and possibly the certification, qualification or fitness requirements of the tools composing the autocoding environment [OH00]. Other issues include ensuring traceability between

elements in models and software requirements, ensuring the configuration control of autocode models (i.e. tracking versions and changes made to models), and managing upgrades in the code generation tools.

A critical aspect of *process* assurance focuses on ensuring safety. During the original baseline projects, software safety analyses were performed placing certain constraints on the on the design of the software. This was particularly the case for the experiment Monitoring and Safety Unit of the Automated Transfer Vehicle. Used for managing the docking manoeuvre as it joins the International Space Station, this subsystem represents the highest possible level of safety critical space software.

The consequences of this original safety analysis were in the form of constraints: *requirements* added to the software requirements specifications to cover e.g. conditions for reporting bad health status, actions to be performed when some units report bad health status, new actions to be performed within precise time ranges or *design constraints* added to architectural and detailed designs: e.g. partitioned memory and redundancy.

Studies such as the SETTA project [SE02] have highlighted that decoupling can occur between the functional development process and the safety analysis process. The focus in the ACG experiments will be to see if these original design constraints are still relevant and are feasible to implement in the autocode models and autocoding development process.

3 QA in the ACG Experiments

Findings from the experiments will come from three main sources:
- Metrics from a quality model for autocode,
- Code analysis and reviews, and
- Lessons learned workshop.

SYNSPACE has developed a **quality model** for the experiments, based on a tailoring of an ISO 9126 conformant quality model developed for the space domain [SP02]. Of particular interest are the quality attributes associated with code functionality, reliability, maintenance and safety as well software development and system engineering effectiveness. Additional metrics are included to cover quality aspects of the autocode models (e.g. size, complexity) as discussed above. Overall project productivity and reactivity (time to change) will also be evaluated.

Two **code reviews** are planned during the implementation phase. These analyses of the generated code will be performed by SYNSPACE to provide insight on the quality of the generated code and the effectiveness of modelling standards applied. SYNSPACE will use the TICS [TIO] tool to verify that the generated code is in compliance with the MISRA C coding standards [MI98].

4 Conclusion

Automatic Code Generation is an emerging technology that holds great promise for future generations of embedded systems. However, ACG raises many questions about model and code quality. An enhanced and thoughtful approach to Software Quality Assurance is, therefore, required to allow autocoding to become a trusted industrial solution. As a practical result of the experiments, changes to standards and guidelines are proposed which can be widely used for mission- and safety-critical software.

5 References

[Ba03] Baumgartner, W., "Turbo für den C-Code", RealTimes 2-2003, BMW Group, 2003.

[BM02] Bostic, D., Milam, W. P., Wang, Y., Cook, J. A., "SmartVehicle Baseline Report – Embedded Software Analysis and Generation", Ford Motor Company, May 2002.

[Ca02] Camus, J. L., "Efficient Development of Avionics Software with DO-178B Safety Objectives", Esterel Technologies, 2002.

[Co00] Conquet, E., "SPACES: Software Production Using Automatic Code Generation for Embedded Systems", Process Improvement Experiment, Final Report, ESSI Project Number 27931, Astrium, May 2000.

[CZ04] Cormery, P., Zindy, G., Conquet, E., Automatic Code Generation in Space On-Board Applications: From R&D to Industrial Use, Proceedings of the DASIA 2004.

[EC03] ECSS-Q-80B, Product Assurance : Software Product Assurance, 10 October 2003.

[EST] Esterel Technologies, "Faster and Cheaper Level A Software – Certification with SCADE", Eurocopter.

[ET03] ETAS, "Generating Efficient Production Code", ETAS, ASCET case Study No 4, December 2003.

[Fr03] Frost G. (Ricardo Tarragon Ltd.), "Automatic code generation for safety-critical systems", MIRA New Technology 2003.

[HS01] Heintz, P., Siller, A., Waldmann, P., "ASCET-SD Traffics Trade – Safe rail vehicles thanks to ETAS tools", RealTimes 1-2001, 2001.

[MA01] The MathWorks Automotive Advisory Board (MAAB), "Controller Style Guidelines for Production Intent Using MATLAB, Simulink and Stateflow", Version 1.00, April 2001.

[MI98] MISRA-C:1998 Guidelines for the Use of the C Language in Vehicle Based Software, ISBN 0 9524156 9 0, April 1998.

[OH00] O'Halloran, C., DERA, "Issues for the Automatic Generation of Safety Critical Software", Proceedings of the Fifteenth IEEE International Conference on Automated Software Engineering (ASE'00), p. 277, September 11 - 15, 2000, Grenoble, France.

[OSA] O'Donnell, J. R., Stephen, D., Andrews, F. et. al., "Development and Testing of Automatically-Generated ACS Flight Software for the MAP Spacecraft", NASA Goddard Space Flight Center.

[SE02] The SETTA Consortium, "Systems Engineering for Time Triggered Architectures – Final Document", Deliverable D7.3, Version 1.0, April 2002.

[SP02] ESA/ESTEC SPEC/TN3 V3.4 Space Domain Specific Software Product Quality Models, Requirements and Related Evaluation Methods, February 2002.

[TAV99] Teraillon, J.L., Ankersen, F., Vardanega, T., Carranza, J.M., Automatic Code Generation and Space On-Board Software, Proceedings of the DASIA 1999 conference.

[TIO] TIOBE Coding Standards Framework. See www.tiobe.com

[ZL05] Zindy, G., Lacan, Ph. et. al, Automatic Code Generation: Issues and Solutions, Proceedings of the DASIA 2005 conference.