CERTIFIED SOFTWARE FACTORY-

OPEN SOFTWARE TOOLSUITES, SAFE METHODOLOGIES AND SYSTEM ARCHITECTURES

J. U. Gärtner
Esterel Technologies GmbH
Otto- Hahn- Str. 13b
85521 Ottobrunn- Riemerling
Germany

1. INTRODUCTION

Over the last decades, the concept of Model- based design has emanated and is now the state- of- the art for most embedded applications. A large number of parallel concepts exists here. Those tools have evolved from pure specification and documentation tools to tool suites allowing design of executable specifications which in some case allow to automatically generate application code.

These tools can be grouped in several classes, including

- UML- based tools
- Simulation- centric proprietary tools
- Formal tools and methods
- Domain- specific SW tools

Two contradictory trends can be observed: Some tool providers follow the path to open standards (e.g. UML2), open interfaces and formats (Eclipse, XML) and thus enable the user to build his own environment tailored to his needs.
Other tool providers hope to be heavy- weighted enough to build their own community which is relying on a proprietary format. (for example Simulink, Statemate)

In safety- relevant systems design, the usage of software design tools is highly recommended. However, the industry trend to automatic code generation is facing some difficulties in this domain

- process integration
- safety requirements: code generation only pays off if code generator is trusted by certification bodies
- domain- specific solutions lack openness and momentum because they are only deployed in small niches

The need for a solution that combines certified automatic code generation with truly open tool architecture and interface concept is more then obvious.

We will discuss these topics on the example of SCADE, the **S**afety **C**ritical **A**pplication **D**evelopment **E**nvironment by Esterel Technologies.

SCADE provides a modelling environment from which code can automatically be generated, while with open and documented interfaces providing full and seamless integration capabilities into existing development flows and processes.

2. LAYERED ARCHITECTURE

Prerequisite for seamless integration in existing or new software design processes is an open, scaleable architecture of the tools.

When discussing the interface architecture of a core tool, which is intended to be able to provide a hub- like functionality in the flows inside a tool workbench, some requirements become soon obvious:

- Abstraction: the system needs to be layered in a way that on each level provides abstract and encapsulated information
- Openness: all relevant information must be readily accessible
- Standardization: the interfaces must be based on commonly accepted industry standards

This is achieved by implementing layered tool architecture. An open architecture outside layer provides abstract access to all the information, which is contained in the core.

Core and interface layer together provide the basis for the Certified Software Factory.

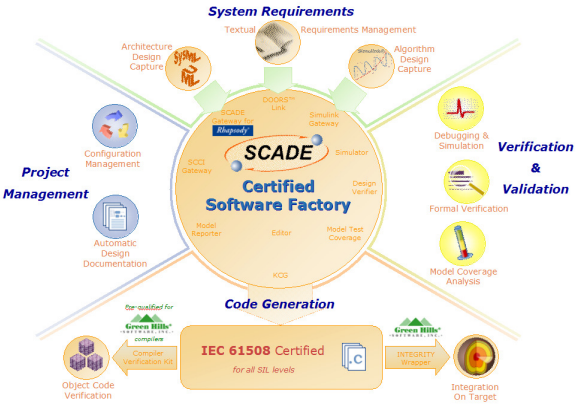

IMAGE 1. Layered architecture of the certified software factory

2.1. Interfaces based on open software architecture concept

Open software architecture interfaces rely on several concepts:

- Standard, openly documented file formats, equally readable by humans and machines
  - SCADE relies on standards such as XMI2, XML and ASAM-MCD2
- Standard, openly documented APIs (application programming interfaces)
  - SCADE provides TCL and C- based interfaces as well as an Eclipse- Plug-in.
- Models are stored as Meta-models so that they can be transformed to and from any other model format
  - SCADE stores the information in an UML-Metamodel

2. . . Example: SysML interface

When transforming models, one must take care to do meaningful translations conforming to the semantic properties of the underlying modelling languages.

Building such an interface requires analysis of the data formats as well as the semantics.

SysML	SCADE

• Overview
  • Semi-formal
  • Asynchronous
  • Object-oriented
  • Good for structural description
• Main construct
  • Classifier & Behaviour
• Dynamic (Instance/ link creation)
• Explicit & implicit flows (connectors or object references
	• Overview
	  • Formal
	  • Synchronous
	  • Functional with state
	  • Good for behavioural description
	• Main construct
	  • Node (close to UML behaviour)
	• Static (everything pre- instantiated)
	• Explicit flows only

IMAGE 2. SysML and SCADE semantic comparison

Deeper analysis shows that the SCADE and SysML notations are very complementary. The ideal pivot point for model transformations is the class/ node interface. When concentrating on this construct, the user gains a hybrid view on the overall model: Dynamic, object-oriented view on model architecture linked with static, instantiated and synchronous view on behaviour.

If such a model translator is additionally based on OSA (open software architecture) concepts and commonly accepted standards such as XMI2 and a meta- model approach, it can easily be built in a very generic way, allowing adaptations for all kinds of UML2/SysML dialects and specific profiles.

The SCADE Gateway to Rhapsody® is an instance of such an implementation.

2. .2. Example Requirements management interface

Requirements are usually formulated in textual form and stored either in a database or in text processing tools.

Requirements may be further refined and result in software or system design, CAD drawings or other format.

An open development platform must therefore provide a means to link requirements specifications (in whatever format) with designs and models, test cases or source code (in whatever format).

The SCADE requirements management gateway enables the user to link all his tools and data together and have instant and global understanding of the interdependencies and relationships.

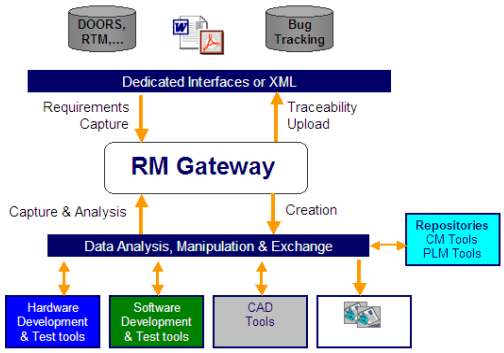

IMAGE 3. Requirements management gateway providing traceability throughout the entire lifecycle

2.2. The core of the certified software factory

At the centre of the software factory, there is a repository containing the information that describes the behaviour of the software.

On the one hand, this model complies with the notion of an UML- Metamodel, meaning that the contained information can readily be accessed through a standardized interface (script language or Eclipse).

On the other hand, the model must obey very strict requirements in order to comply with the requirements imposed by the standards that drive safety- relevant systems development: DO- 78B, IEC6 508- and –3, EN50 28.

High integrity levels imply formal models and unambiguous semantics that allow representing the typical features of embedded software systems: reactive systems with data flow, discrete states and concurrency, coupled with hard real- time constraints.

SCADE modelling language has evolved from LUSTRE, a formal, synchronous model description language.

The user interface provides the developer with a very intuitive view, based on block diagrams and state charts, tightly integrated. Powerful constructs for vectorization of flows and operators tackle even the most complex problems.

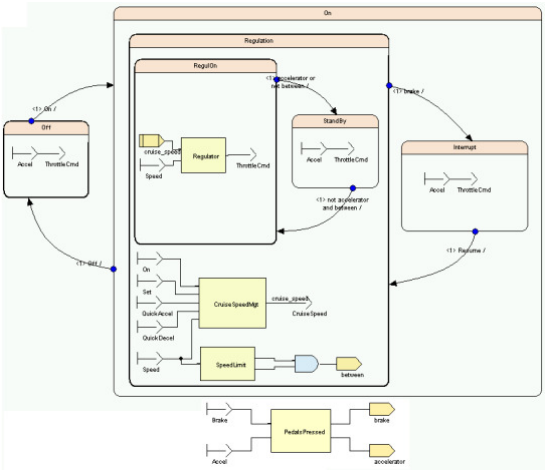

IMAGE 4. SCADE model representing a fully functional automotive cruise control application

This model is immediately executable for verification and validation purposes.
The development environment includes a powerful software-in- the- loop simulator with model- level debugging features.

Thanks to the formal nature of the model, it can also be examined by formal/ mathematical analysis and prove engines, such as the integral SAT- solver *Design Verifier,* which provides formal proof of functional safety properties.

The open software architecture makes this model fully accessible through the customer's specific tool suite and provides transformation engines to and from this environment.

It is also the basis for automatic generation of SDD documents (software design descriptions) and, more importantly, serves as direct input for certified code generation.

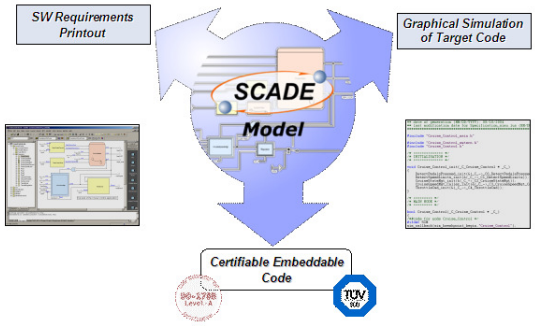

IMAGE 5. Formal model as the hub of the SW design process

Certified code generation ensures, that

- The code complies 100% with the model in the sense that the code fully and deterministically represents the behaviour described in the model
- The generated code complies with each and every objective and requirement imposed by the standards to which it has been qualified (DO-178B up to Level A) and certified (IEC61508, up to SIL 4)

For example, the generated C- code does not contain operations on pointers, no global variables, no indefinite loops, no dynamic memory allocation etc.

At the same time, it fulfils very stringent requirements related to memory usage and execution time.

It is absolutely comparable to highly optimized hand- written code.

Moreover, it is totally target- agnostic and therefore easily to be integrated on all platforms, from bare machine to complex distributed systems.

Certified code generation is a key to a completely defined process that covers all steps from requirements capture down to integration on target.

High quality of generated code and restriction to a very small subset of C allow also to verify correct compilation through compilation and automated verification of a representative model containing the complete generable subset of C in all its possible combinations and nested operators, resulting in a combined testing process which ensures and guarantees that each requirement is correctly designed, modelled, coded, and the integrated on the target hardware.

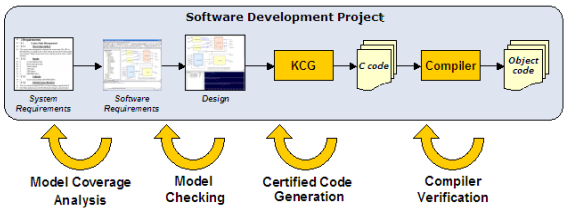

IMAGE 6. The combined testing process

3. SAFE SYSTEMS ARCHITECTURES

The tool suite and process outlined above ensures that no systematic errors can be introduced into the software design when transforming the system requirements relevant for software into an application.

Safe system architecture needs to also ensure that hazards or spontaneous, non- systematic errors on the hardware, sensors or from the environment will not affect safe operation of the system.

Various approaches exist to this problem, some of which include redundancy, dissimilarity and built- in tests.

All of them go beyond the scope of this paper, but share the same principle: A layered systems design which clearly separates the application from the hardware and usually incorporates a safe and certified operating system.

An open software development environment must directly support automatic integration of the generated code onto such safe HW/SW platforms.

SCADE provides such interface to several certified/ qualifiable operating systems such as GHS Integrity, Sysgo PikeOS or MicroC.