Softw Syst Model (2012) 11:303–304
DOI 10.1007/s10270-012-0251-2

EDITORIAL

Modeling to improve quality or efficiency? An automotive domain perspective

Robert France · Bernhard Rumpe

Published online: 30 May 2012

What added-value does modeling bring to the development process? An often stated value is the productivity improvement gained by using models to automatically generate code. Model-to-code generators provide automated support for bridging abstraction gaps, and thus relieve developers from routine, often tedious, decision-making and implementation activities. Developers using model-driven software development techniques should be able to deliver working software at a faster rate, and thus reduce time-to-market.

Model-to-code generation is particularly valuable when it supports synchronized evolution of generated code and models, as opposed to "one-shot generation" in which generated code that has been modified cannot be re-generated from models because the models do not reflect the changes made to the generated code. Maintaining fidelity between abstract models and the more detailed code as they evolve can lead to faster delivery of software extended with new or modified features.

However, in the automotive and similar embedded software industries, process efficiency may be less of an issue. Consider the case of a particular European automotive company that had a total turnover of 60 Billion Euros and earned 6 Billion Euros. Car development, production, sales, after-sales, and other costs for this company thus totaled roughly 54 Billion Euros. The company employs approximately 1,000 software developers and approximately another 1,200 software developers from component suppliers perform work for the company. If an engineer's average salary is 120,000 Euros (this is the average salary reported in a recent study), then software development manpower cost for this company is approximately 264 Million Euros. This is 0.45 % of the overall costs. While 264 Million sounds like a lot, it is relatively insignificant compared to other costs. For example, a software error that results in a recall of over one million cars can be very costly. Assuming a cost of 400 Euros per recall, the result would be over 400 Million Euros in cost. In such a scenario the cost of a recall easily exceeds the total software development manpower costs.

So what really counts in these cases? The answer should not be surprising: quality, quality, and quality. Time-to-market is second significant concern for these organizations. Efficiency probably only counts, when the availability of experienced developers to hire is low.

Quality concerns can very nicely be addressed by model-based development methods. Modeling techniques can be used to automatically synthesize and compose sub-models that capture different features of a distributed system. Composed models can be used to generate high quality code that works on appropriate middleware, but was developed independent of any technology stack and can therefore be reused rather unchanged. The big advantage of reuse is not the efficiency increase, but the reuse of quality that was already proven.

We therefore need appropriate tooling infrastructure for developers in the automotive and similar domains for quality and not that much for efficiency reasons. But we also need industries that have the courage to accept that the development process for software is significantly different from the traditional engineering process. Only then software will keep up with the desired quality and be there in time. Traditional engineering processes today significantly lack to address software specific characteristics, but appropriate combinations of software-aware development with

R. France
Colorado State University, Fort Collins, Colorado, USA
e-mail: france@cs.colostate.edu

B. Rumpe (✉)
RWTH Aachen University, Aachen, Germany
e-mail: Bernhard.Rumpe@sosym.org

the overall systems development are not yet established. To meet these challenges we need close collaborations between researchers in academia and practitioners in these industries and need to accept that computer science has to deal with applied research topics as well. We welcome submissions that describe industrial experience and innovative methods that resulted in advancing the state of the research and practice in model-driven software development.

We hope you enjoy reading this issue.