Softw Syst Model (2012) 11:151–152
DOI 10.1007/s10270-012-0244-1

EDITORIAL

Data warehouse concepts for model artifacts?

Robert France · Bernhard Rumpe

Published online: 3 April 2012

Manufacturing industries, in particular avionics and automotive industries, that have embraced model-driven system development approaches, are faced with the challenge of getting development tools that address different, but related aspects of the development process to interoperate. This challenge led to increased interest in and work on tool-integration approaches that enable the creation of "tool chains". These approaches are based on the assumption that it is best to use tools that are specialized for specific purposes. An appropriate way to integrate the tools is to provide facilities for transferring the outputs of one tool to another specialized tool. This typically requires transformation of data produced from one tool to a format suitable for input to other tools.

As an example, the development of software for a cyber-physical system (CPS) based on a communication bus (e.g., software found in cars) is based on a dominant bus architecture. We have seen in current development approaches any piece of data (signal) flowing between control devices that are connected with a bus needs a unique name (identifier). This enforces that signals be maintained in a centralized catalog to help ensure qualities such as consistency and completeness, even though they never flow on the same bus and do not interfere at all in the CPS and ruins flexible adaption of the CPS.

Furthermore, such a catalog of signals normally starts as a relative simple database. Practical experience shows that this signal database is typically augmented with additional information about the signals. This additional information includes, among other things, the source ECU (electronic control unit) and target ECUs, the logical function blocks involved in signal communication, timing information, and signal dependencies. It is also useful to add information that can be used to trace requirements and feature variants to their realizing signals. This allows developers to determine whether a signal is still necessary when requirements become obsolete or change. In turn, this enables optimization of signal-based bus layouts.

Extending the signal database with this additional information occurs often in practice and allows to connect various development activities and their artifacts. These extensions though run counter to the original notion of a "tool chain", where the output of one tool is fed into the input of the next tool. The signal database approach has characteristics of a common "data warehouse" that provides model data on which all tools operate on. This approach has advantages and risks. It is definitely an advantage to have integrated data structures that greatly reduce data redundancy, and in turn decreases the likelihood of data inconsistencies.

However, there are risks. First, there is only little experience in developing an integrated data model for specialized tools, some of which operate on models that are expressed in non-standardized notations. It may easily happen that the data model is incomplete or inaccurate. Second, tools can be expected to evolve relatively often and each tool may evolve at different rates. One big challenge is to co-evolve the data model so that it keeps pace with varying evolution rates for tools in a tool-chain. This also includes co-evolving the data in the "data warehouse", which is a problem on its own. Third, there is currently no practical approach to coping with different "meanings" of the model concepts in different views or contexts that are syntactically identified, i.e. have the same representation.

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

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

A key issue is the readiness of a development organization to develop and maintain an integrated data warehouse for model data. Can individual stakeholders see and adapt the views they typically use? Are the tools and the data model able to cope with uncertainty and the resulting incomplete or underspecified information? The ability to cope with uncertainty and incompleteness is very important, because not all information may be available in early phases to make informed design decisions. Being forced to take decisions on the system under development too early typically leads to less optimal solutions.

Redundancy-free integration of model data also means that the same functions and signals have the same name throughout the development process. This apparently simple requirement can be challenging to realize. It involves integrating the terminologies of many departments, including marketing, design, quality assurance, and product diagnostics departments. In automotives, e.g., this means that the early requirements developer suddenly defines the signal and feature names that finally appear in the diagnostics tester in the repair shops.

None of the above issues seems intractable, but there is a need for solutions that are practical and cost-effective. Will tool-chains based on integrated warehouse of model data (like a business data warehouse) take-off? Or will we see better forms of loosely coupled chains of modeling tools (e.g., an orchestrated set of modeling services in the cloud)? Or will approaches that are based on a single open tool environment that allows plug-ins (e.g., the Eclipse platform) become dominant? At this point it is unclear if any of the above approaches is technically superior. We look forward to receiving submissions that tackle the important problem of tool interoperability in model-driven development environments especially when looking at the newly arriving domain of cyber-physical systems that have an immanent need for integration of different viewpoints including, e.g., various mechanical, electrical, and even hydraulic models.