# Automated Model Compiler (AMC) based on Design Space Exploration Tool (DESERT)

*Sandeep Neema (sandeep.neema@vanderbilt.edu)*

This document provides a short summary on the scope and usage of the AMC tool-chain provided as part of the OTIF distribution. We give a short overview of the AMC problem, the DESERT tool, and then a synopsis of the AMC tool-chain. For additional details on the AMC or the DESERT tool, a list of references is included.

## Automated Model Compiler

An embedded automotive systems engineer is often faced with the challenge of composing models of vehicle system assemblies, from a set of sub-models for the purpose of simulation, verification, or synthesis. A typical embedded automotive example has 75-100 model components, each of which has 3-30 alternatives, and ~2000 signal inter-connects. In addition to the sheer scale of the problem, the system engineer also has to ensure the compatibility of inter-connected components and meet the overall design and performance requirements while performing the composition. Motivated by this enormous complexity, Ford engineers defined a challenge problem for an Automated Model Compiler in a paper published in the Proceedings of the Embedded Software Conference, 2001 [1] – "A *model compiler* is a tool that automatically composes a model from a set of sub-models and an architectural description of the arrangement of sub-models, ensuring full-connectivity of all control flow and data flow signals between sub-models, proper sequencing of sub-models, and compatibility of sub-models".

## Design Space Exploration Tool

DESERT is a meta-programmable tool for navigation and pruning of large design spaces using constraints. It provides a generic structured representation of design-spaces based on the concept of alternatives and parameters. With this capability one can represent a rich variety of problem domains such as product-line architectures, hardware-software co-design, and automated model compilation, to name a few. DESERT has an expressive constraint language based on a subset of OCL, which allows expressing compositional, resources, and performance (time, energy, size, weight, and cost) constraints. DESERT incorporates a powerful and highly scalable symbolic constraint solver based on Ordered Binary Decision Diagrams. The input and output interfaces of DESERT are XML based, and are accompanied with a programmatic API which renders the integration of DESERT into tool-chains convenient. For additional details on DESERT refer to [2] and [3].

## AMC Tool Chain

Leveraging the strength of DESERT, we have developed the AMC tool chain in response to the challenge problem referred above. Figure 1 shows the key components and the work flow in the AMC tool-chain. The key components of this tool-chain are as follows:

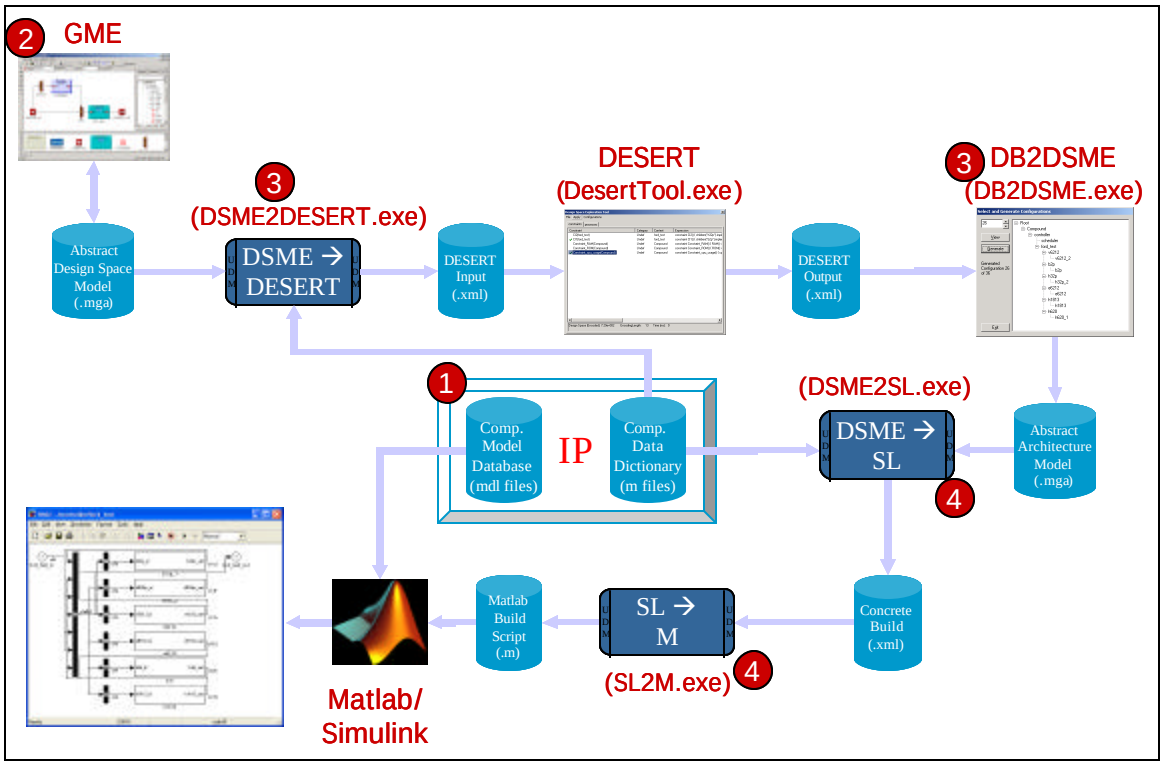


**Figure 1: AMC Tool Chain**

1. Matlab/Simulink and Component Repository: The repository contains simulation model for various automotive subsystems. The models are stored in .mdl (Model Definition Language) files, and there is an associated definition file, that defines the input and output signals, a number of performance parameter (e.g. CPU usage, RAM/ROM usage) and characterization information (green vs. fun-to-drive vehicle, etc.) for the Simulink/Stateflow models.
2. Design-Space Modeling Environment (GME): The challenge problem definition suggested the introduction of a high-level modeling language for the specification of target architectures. This language uses the abstracted components at its leaf nodes so as to allow modelers focusing on the appropriate level of abstraction. Therefore, we defined an SF-like design language with hierarchy and alternatives and instantiated it in our meta-programmable Graphical Model Editor (GME). Design space models capture the hierarchical composition of vehicle systems and capture design alternatives for subsystems. A primitive (leaf node) in this language represents a simulation model in the Matlab/Simulink Component Repository, and is linked to the simulation model through attributes that store model name, version number, and file name. There are additional placeholder attributes for performance, and characterization parameters that need not be filled by the user. The translator in the Component Abstraction tool parses the parameter and I/O definitions and populates the models with this information. The user can model design specifications (e.g. CPU_usage < 70%, RAM < 20Kbytes)

as constraints in the design space models. Consistency constraints (e.g. connected I/O should have matching data types) are automatically introduced in the models. The Design-Space Modeling Environment supports the specification of structural, and component compatibility constraints in OCL.

3. Design-Space Abstraction (DSME2DESERT and DB2DSME): DESERT uses a domain-independent meta-model, which separates its internal algorithms from domain-specific constructs. The Design-Space Abstraction component of the AMC tool-chain provides two-way model translation between the Design-Space Models and the DESERT's abstract design-space models. The two-way translation enables that acceptable point designs selected from the pruned design space by DESERT can be presented in the Design-Space Modeling Environment and can be translated further automatically for the Matlab/Simulink environment for detailed simulation studies.
4. De-abstraction and Assembly (DSME2SL and SL2M): This component of the AMC tool-chain elaborates the abstract high-level architectural model with Simulink model details and constructs the assembly model.

## References


[1] Butts K., et. al. "Usage Scenarios for an Automated Model Compiler," EMSOFT 2001

[2] Neema S., et. al. "Constraint-Based Design-Space Exploration and Model Synthesis," EMSOFT 2003

[3] Neema, S., "Design Space Representation and Management for Embedded Systems Synthesis," Technical Report, ISIS-01-203, February 2001.