By using high-level domain specific languages for designing programs, for example in the hardware domain, the model-based approach promises a more transparent and efficient development process. The main task is to synthesize the program into the target language, e.g. a circuit, which is done by the compiler. Designing, understanding, and maintaining a compiler still remains a difficult task. Although the user has some degree of control over the compilation, compilers are usually black boxes. It is relatively hard to grasp beforehand how the final or intermediate results will look like. In addition, the results are not necessarily executable models or comprehensible in a human-readable way.

By leveraging the complete compilation workflow of chained model-to-model transformation systems to a meta level, both, the tool developer and the modeler, can benefit from an interactive development process. Combined with modern transient view and automatic layout technologies, creating and working with model-to-model transformation systems becomes transparent and less time consuming. We argue that modern development tools, such as model-based compilers, besides producing the correct result, can and should also guide the modeler to potential issues and provide means to understand what is happening during the transformations.

The KIELER\textsuperscript{1} SCCharts Editor is such a modular, open-source modeling suite, using the synchronous language SCCharts\textsuperscript{2} \cite{Smyth13} as main demonstrator, but can handle arbitrary many languages. The SCCharts statechart dialect is a successor of SyncCharts, which can be seen as a graphical version of Esterel. The editor supports high- and low-level language compilation \cite{Motika14}, automatic syntheses of intermediate results using a transient view framework \cite{Schneider13} with automatic layout via the Eclipse Layout Kernel (ELK)\textsuperscript{3}, and deployment to different platforms, both software and hardware \cite{Rybicki18}. The modular concept of the compiler framework \cite{Smyth18} allows for rapid application and prototype development. The concepts used to build the reference SCCharts compiler can be applied to various other languages or domains and are not bound to SCCharts.

![Fig. 1. Different life cycles of an SCCharts model in KIELER. The graphical views (here (1) the normalized model, (2) a controlflow graph representation, and (3) a generated circuit) are synthesized automatically from the textual model and the intermediate results of the transformation steps. The developer can select specific transformation steps (e.g. 1-3) and then inspect model element relations (orange arrows) between the selected intermediate results interactively. Since the compiler framework is fully modular and generic, arbitrary transformation steps can be performed.](image-url)