Purpose of DSP\textit{developer}

Provide an integrated environment of MathWorks tools and VisualDSP++ that supports the complete life cycle of an algorithm, from concept to an optimized, validated VCSE Component that can be incorporated into product software.

Integration of VCSE Component within total software project (VisualDSP++)

Sheet Dynamics, Ltd., Proprietary and Confidential
Use standard Simulink environment to develop algorithm. Algorithm behavior is simulated on desktop, inputs can be read from disk files, outputs can be saved to variety of output devices. Once algorithm is working (in this case the green Spectral Subtraction block) a prototype can be automatically generated to run on DSP hardware (including EZ-Kits).
Rapid Prototyping

Now I/O is replaced with ADC & DAC blocks specific to the selected target (e.g., an EZ-Kit). An executable program can be then automatically generated from the diagram.

DSPdeveloper control panel provide basic VisualDSP++ debugging functionality from within the Mathworks environment.

Algorithms can be tested on actual hardware. Interactive capability is provided to allow for monitoring internal signals.
Now to form a component based on algorithm a VCSE Component Subsystem is copied into the model.

Double clicking on the Component Subsystem will open the subsystem and a Dialog box. Within the subsystem an interface block is placed. The interface block can be used to define a VCSE interface.

The subsystem that implemented the algorithm is then placed into the Component Subsystem. IO connection points are automatically generated based on Interface blocks contained in subsystem.

The Dialog box allows VCSE Component documentation to be entered (e.g. Component name, etc.).

A packaged VCSE compliant Component can now be automatically generated by pressing the Build Component button.
Component Optimization

Source code can be placed into a VisualDSP++ project, complete with the necessary make file to build the packaged Component. The users can, if needed, optimize the component for speed, memory usage, etc.
Optimization/Validation

Now a means of validating the optimized Component is provided by automatically generating a test harness for the component.

This test harness allows for a Component that is executed on hardware to be incorporated within a Simulink Simulation.

This is very useful for validation of Component implementations. For example, an optimized Component’s operation can be compared to the original Simulink Component description as shown in the lower diagram.