SIEMENS

Ingenuity for life

Capital Software Designer

Streamlining model-based software engineering

Benefits

- Create enriched architecture models to drive the design process
- Connect to your implementation tool of choice to develop code
- Integrate, verify and validate your code to deliver onboard software quality

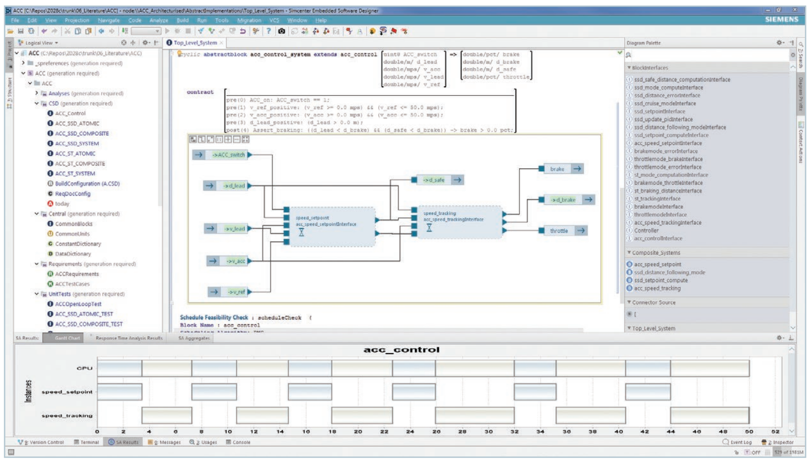
Summary

Capital Software Designer software connects the elements of your embedded software design processes and helps you avoid rework costs by detecting errors early in the process. An architecture-centric approach allows you to define, enrich, analyze and simulate onboard software designs.

The solution is open so you can implement onboard software designs in external development environments with the template export functions. Its external development support covers the C language for hand coding and the Simulink® environment interface for model-based development.

The Capital Software Designer contract-based architecture enables you to drive testing, verification and closed-loop simulation, even when following mixed external implementation paradigms. Capital Software Designer is interoperable with other validation tools and platforms, thus providing digital continuity in scattered process and tool landscapes.

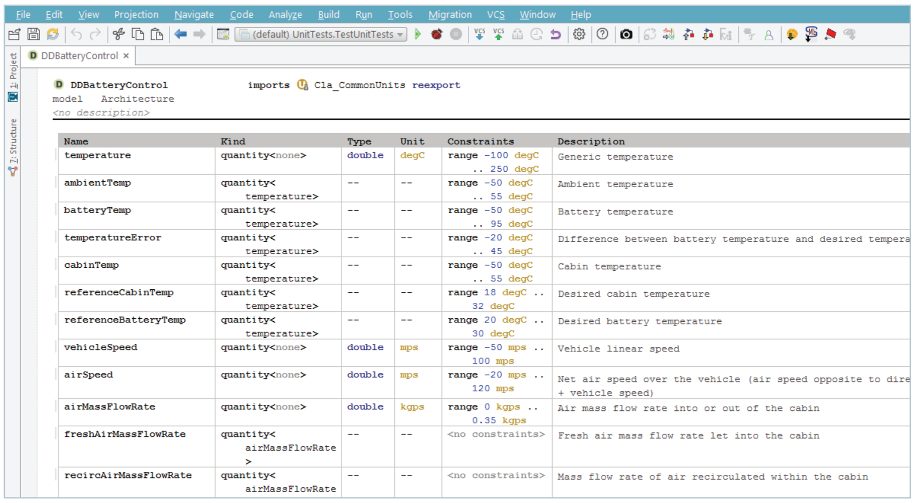
Capital Software Designer is tightly integrated with other Simcenter solutions, as well as Polarion ALM™ software. This provides a connected environment for embedded system and software architecting, engineering, design, validation as well as multi-physics simulation, application testing and lifecycle management.



```
cyclic abstractblock acc_control_system extends acc_control [uint8 ACC_switch     ] => [double/pct/ brake   ]
                                                            [double/m/ d_lead      ]    [double/m/ d_brake   ]
                                                            [double/mps/ v_acc     ]    [double/m/ d_safe    ]
                                                            [double/mps/ v_lead    ]    [double/pct/ throttle]
                                                            [double/mps/ v_ref     ]
contract [
         pre(0) ACC_on: ACC_switch == 1;
         pre(1) v_ref_positive: (v_ref >= 0.0 mps) && (v_ref <= 50.0 mps);
         pre(2) v_acc_positive: (v_acc >= 0.0 mps) && (v_acc <= 50.0 mps);
         pre(3) d_lead_positive: (d_lead > 0.0 m);
         post(4) Assert_braking: ((d_lead < d_brake) && (d_safe < d_brake)) -> brake > 0.0 pct;
         ]
```

siemens.com/swdesigner

Capital Software Designer



DDBatteryControl imports Cla_CommonUnits reexport
model Architecture
<no description>

Name	Kind	Type	Unit	Constraints	Description
temperature	quantity<none>	double	degC	range -100 degC .. 250 degC	Generic temperature
ambientTemp	quantity< temperature>	--	--	range -50 degC .. 55 degC	Ambient temperature
batteryTemp	quantity< temperature>	--	--	range -50 degC .. 95 degC	Battery temperature
temperatureError	quantity< temperature>	--	--	range -20 degC .. 45 degC	Difference between battery temperature and desired tempera
cabinTemp	quantity< temperature>	--	--	range -50 degC .. 55 degC	Cabin temperature
referenceCabinTemp	quantity< temperature>	--	--	range 18 degC .. 32 degC	Desired cabin temperature
referenceBatteryTemp	quantity< temperature>	--	--	range 20 degC .. 30 degC	Desired battery temperature
vehicleSpeed	quantity<none>	double	mps	range -50 mps .. 100 mps	Vehicle linear speed
airSpeed	quantity<none>	double	mps	range -20 mps .. 120 mps	Net air speed over the vehicle (air speed opposite to dire + vehicle speed)
airMassFlowRate	quantity<none>	double	kgps	range 0 kgps .. 0.35 kgps	Air mass flow rate into or out of the cabin
freshAirMassFlowRate	quantity< airMassFlowRate >	--	--	<no constraints>	Fresh air mass flow rate let into the cabin
recircAirMassFlowRate	quantity< airMassFlowRate	--	--	<no constraints>	Mass flow rate of air recirculated within the cabin

Streamlining model-based development
Create a holistic software architecture that acts as single source of truth across the software development lifecycle.

Capture and enrich software architecture

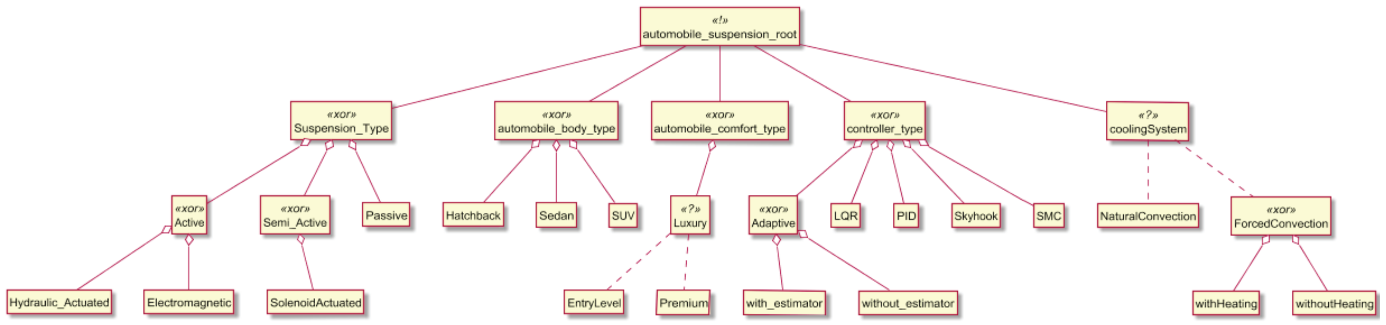
- Use data-flow modeling to support abstraction and re-use
- Enrich software architectures with contract and timing needs
- Identify precise type system and physical units
- Import and merge the architecture created in Systems Modeling Language (SysML), Autosar and Architecture Analysis and Design Language (AADL)
- Use intuitive graphical modeling based on unlimited nesting of data flows
- Re-use test case definitions to help clarify requirements and functional specifications

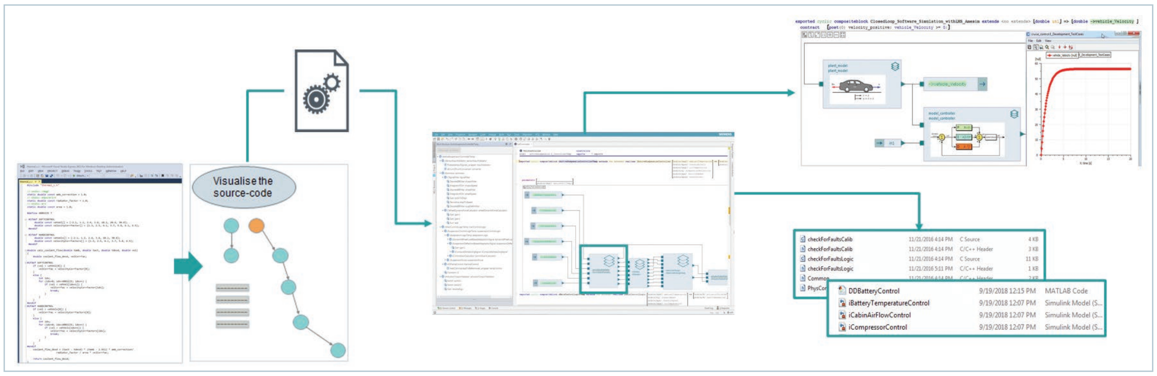
Analyze software architectures

- Analyze architecture for consistency, completeness and early detection of specification errors enabled by formal methods
- Schedule feasibility analysis to front-load software-hardware co-development

Manage design data

- Use detailed specification of elements in the data dictionary to specify data types, physical units and range constraints for signals and parameters
- Capture the product line variability in the parametric data by using constant groups





checkForFaultsCalib	11/21/2016 4:14 PM	C Source	4 KB
checkForFaultsCalib	11/21/2016 4:14 PM	C/C++ Header	3 KB
checkForFaultsLogic	11/21/2016 4:14 PM	C Source	11 KB
checkForFaultsLogic	11/21/2016 3:11 PM	C/C++ Header	1 KB

DDBatteryControl	9/19/2018 12:15 PM	MATLAB Code
iBatteryTemperatureControl	9/19/2018 12:07 PM	Simulink Model (S...
iCabinAirFlowControl	9/19/2018 12:07 PM	Simulink Model (S...
iCompressorControl	9/19/2018 12:07 PM	Simulink Model (S...

Master product line variability

- Support variability in embedded software product lines with a formal feature model and a rich set of product modeling variant options
- Find inconsistencies in feature models, product configuration instances and implementations

Prepare and control implementation

- Support external implementation in C and the Simulink environment as well as internal implementation
- Provide a central place to configure implementation choices per block
- Export C code and header files that contain the correct function interfaces and rich comments describing contracts and data elements; or Simulink implementation templates with correct port interfaces
- Support iterative processes with frequent interface changes

Integrate software implementations

- Integrate legacy software and convert it to a model layer
- Benefit from automatic software integration and reconciliation of external implementations coming from hand-coding or auto-code generation from Simulink, enabled by a contract-based approach
- Verify the implementation against pre- and postconditions

Manage software configurations

- Take advantage of built-in integration with configuration management tools for distributed development
- Access all operations of version control systems, such as branch, commit and merge in the integrated development environment (IDE)

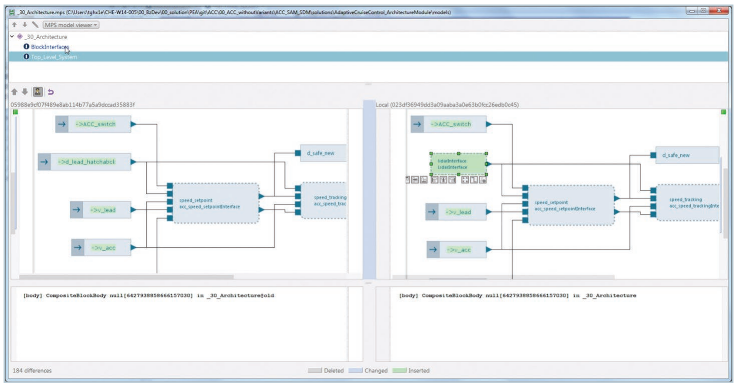
Trace requirements

- Benefit from bi-directional traceability between requirements in Polarion ALM and implementation artifacts in Capital Software Designer
- Use many-to-many links and single-click navigation to Polarion ALM web interface

Continuous quality improvement

Test and simulate

- Use the option of creating and managing table-based or complex logic-driven tests, setting up and launching test execution
- Establish enterprise-wide test collaboration thanks to a tight integration with Polarion ALM
- Run batch mode closed-loop



```
[body] CompositeBlockBody null[6427938856666157030] in _30_Architecture
```

simulation with plant models developed in Simcenter Amesim™ software and Simulink

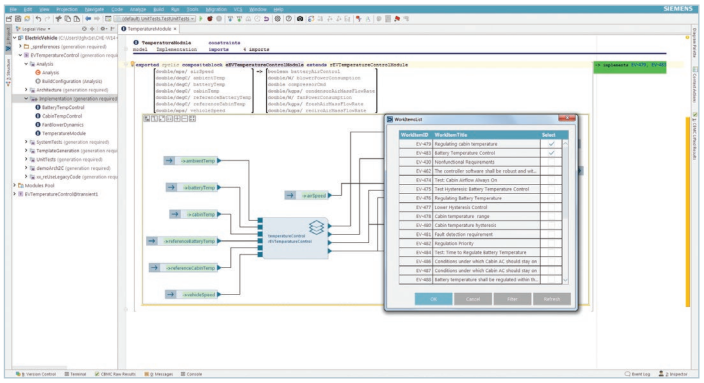
- Export Functional Mock-up Units (FMUs) with choice of compiler (GCC/VCC) and operating platform (32/64 bits) for connecting to Simcenter System Synthesis and any Functional Mock-up Interface (FMI) compliant system simulation tools

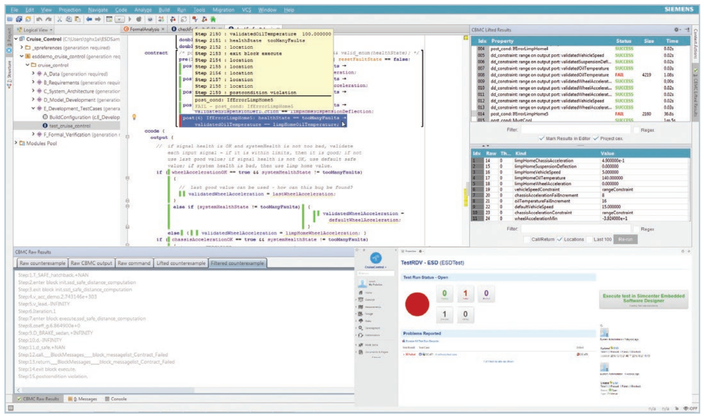
Verify using CBMC

- Verify key runtime properties; division-by-zero, array-out-of-bounds, memory leak, arithmetic overflow and pointer dereferencing
- Verify safety properties expressed in the form of contracts, assertions and data dictionary constraints
- Analyze dead and unreachable code
- Use automatic counter example production at the level of model and line of code
- Increase coverage with code coverage measurement and automatic generation of test vectors

Prepare target deployment of application code

- Analyze worst-case execution time (WCET) independently of the target platform
- Estimate program and data memory and compare the old and new result after a change





Siemens Digital Industries Software
www.sw.siemens.com

Americas	+1 314 264 8499
Europe	+44 (0) 1276 413200
Asia-Pacific	+852 2230 3333