Constellation™ and Simulink™, the Complete Controls Software Platform

Combining UML-based software development and Simulink-based control system development into one end-to-end process

Introduction

Building complex control systems is a huge challenge. Sophisticated control algorithms must be carefully designed, simulated, tested, and then embedded into complex software that coordinates interactions with many other subsystems. The special needs of real-time systems—sequencing, incremental test, online operator interfaces, and live debugging—compound the problem. Engineers, software programmers, managers, and test personnel must cooperate to design, develop, test, integrate and maintain the system. The development spans a long, complex process, from initial prototyping to final system implementation and testing. It’s a true “end-to-end” challenge.

Of course, many types of systems require team effort. Complex control-system teams, however, face a unique challenge: integrating sophisticated behavioral designs into a complex software system. This is quite difficult. In most organizations, different groups, acting nearly independently, develop the controls and the software. Integrating these two camps and their respective development processes has become one of the great barriers to the implementation of aerospace, machine control, and other complex systems.

Why is this? What makes it so difficult to integrate the control design process with the modern, model-based software development process?

To answer these questions, we must first understand both the challenges and the teams.

Challenges and Teams

Control-algorithm designers construct dynamic system models, develop algorithms, and simulate the proposed designs. Control engineers prefer a straightforward, concrete approach. Control engineering tools speak the mathematical language of numeric algorithms and provide fast modeling and numeric simulation.

Software developers, however, provide the infrastructure that glues the system together. They develop specialized modules, such as network communication, device drivers, and user interfaces. Their primary challenges are abstract structuring and practical system organization. Software developers use programming tools that provide coding flexibility, architectural design, code management, debugging, and reuse. Their design tools, presently primarily based on the UML standard, use abstract techniques that emphasize object-oriented design and structure.

Controls engineers are not usually trained in the advanced techniques of software development and software developers rarely have the engineering background for control-systems design. Starting from scratch with abstract modeling languages, the “modern” software design often ends up poorly adapted to control. The path from a blank-sheet software architecture to a working control system is simply too long to succeed.
The Problem

Controls engineers develop the algorithms and then “toss the design over the fence” to the development team. The software team rewrites all the algorithms from scratch, just to fit the computational structure. The project takes forever. System integration takes longer than forever. Rework, such as revising the control algorithms, is painful. It’s hard to reuse code, algorithms, or even the framework. The tools each group uses don’t talk to each other. The whole system development process is risky and slow.

So what can we do? How can complex software systems incorporate sophisticated control designs? How can we merge the tools used in the two development processes?

The Solution

The solution is simple: provide tools that work the way the teams think. Let control engineers develop with state-of-the-art dynamic modeling and algorithm design tools. Then integrate the control system design into a targeted, proven software architecture with complete system-development tools to accelerate implementation and testing. The result will be a software platform that satisfies both engineers and programmers and streamlines the system development process.

RTI has provided the first complete software platform for control systems by integrating Simulink with the Constellation software platform.

Simulink, by The MathWorks, is the tool for developing and simulating control systems. Simulink excels at dynamic systems modeling. It provides a powerful modeling and simulation environment. With MATLAB™, Simulink also provides detailed controls design and analysis.

Constellation is the software platform designed explicitly for complex control systems. It deals with complex and intelligent subsystems, wraps legacy code, and easily incorporates complex, custom modules. It integrates UML specification, code generation, code re-use, test tools, and other development tools into one tool-chain. It can handle large applications developed by teams of programmers, engineers, architects, and managers. Constellation is a clean, object-oriented platform that works in the real world.

Together, Constellation and Simulink address both controls and complex system software. They take the design team all the way from systems prototyping and controls analysis through simulation and test to final application code and system integration. Constellation with Simulink is the complete software platform that streamlines complex system development.

Simulink and Constellation

Simulink is a dynamic-system modeling tool. Constellation is an application framework and software-development environment for dynamic systems. While the Constellation developer tool and Simulink have superficial similarities, they really address different problems. Simulink excels at simulating physical systems and developing control algorithms, while Constellation excels at developing and managing all the software for complex systems that include control subsystems. Together, Simulink and Constellation enable complex system development.

Let’s examine them a little more closely…

Simulink

Simulink is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two. Simulink also supports multi-rate systems, i.e., systems that have different parts updated at different frequencies.

For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click-and-drag mouse operations. With this interface, you can draw control block diagrams just as you would with pencil and paper (or as most textbooks depict them). This is vastly superior to previous simulation packages that required you to formulate differential equations and difference equations in a language or program. Simulink includes a comprehensive block library of sinks, sources, linear and nonlinear components,
mathematical functions, and connectors. You can also customize and create your own blocks.

The Simulink environment also includes an optional state-diagram editor known as Stateflow. Stateflow describes logic sequencing in control systems, and supports flow-diagram notation for common code structures such as for loops and if-then-else constructs. You can use flow diagram notation and state transition notation seamlessly in the same Stateflow diagram.

Simulink is a superb simulation tool. After you define a model, you can simulate it with a choice of high-fidelity integration methods. Simulink includes model analysis tools, including linearization and trimming tools. Simulink supports both batch and interactive simulation. Batch simulation uses MATLAB’s command-line interface; it’s very useful for Monte Carlo simulations or parameter sweeps. In interactive mode, you can see the simulation results while the simulation is running. In addition, you can change parameters and immediately see what happens, for easy “what if” exploration. You can also export the simulation results to the MATLAB workspace for post-processing and visualization.

Finally, Simulink has a code-generation option called the Real-Time Workshop™ (RTW). With RTW, you can generate code from your diagrams, compile it, and download it to run on various real-time targets. All told, Simulink/Stateflow/RTW is a well-designed, complete controls development environment. It’s elegant, powerful, and amazingly useful.

However…

Simulink is not a software platform. It doesn’t support any object-oriented or higher-level programming constructs. For example, it supports containment, but not inheritance or abstract interfaces. Fundamentally, Simulink targets engineers who do not wish to write code.

Simulink blocks are connected only by simple vectors of numeric data. It presents no way to define and reuse generic object or system interfaces. For instance, there’s no way to specify a functional interface to a block; you cannot export or import methods. For that matter, there’s no facility to invoke methods and get a return value; routines execute when they get a data signal due to their position in the block diagram itself.

While there is a facility to create simple extensions with “S-function” files, this facility is
not designed for complex programming. S-functions have a limited, defined interface to external blocks. They cannot control or influence threads or scheduling, pass and return arguments, or present an API to the rest of the system. It’s not practical to implement significant new software functionality (like a network protocol) or wrap complex legacy code or libraries in S-functions.

Simulink does not attempt to be a software development environment. Simulink does not, for instance, maintain user code, help you build and link your own libraries, or manage dependencies with other libraries. There is no easy way to develop, test, share, and maintain components compiled from source code. The fundamental unit of expression in Simulink is a diagram, not a source tree.

This “programming free” philosophy pervades the system. For instance, the RTW Code Generator is powerful, but it’s a one-way process. After generating code with RTW, you can edit the code or edit the diagram, but you can’t do both. The generated code is monolithic C code. Any diagram changes require full code generation and building. With Constellation developer, you can change out just one block and do a quick link.

In summary, Simulink solves an important part of the problem. But it does not solve the entire problem. In most large control systems, the mathematics is a small part of the challenge. You could not, for instance, develop an entire hardware-in-the-loop, distributed avionics simulator, with dozens of programmers and thousands of components, with Simulink.

You can with…

**Constellation**

Once the control system has been designed in Simulink, an enormous software development and system integration process still remains. It includes elements such as network communication, device drivers, external user interfaces, legacy code, and interfaces to other design tools. Constellation manages this process; it is a complete software development environment and framework, specifically designed for complex, intelligent control systems.

Constellation encompasses all the tools and ‘glue’ required to build applications. You can develop complex and intelligent subsystems, wrap legacy code, and integrate custom modules. It integrates the four most-used UML standard diagrams to help you specify your system and elaborate your requirements. It can manage large applications developed by teams of programmers, controls engineers, and systems engineers. Constellation is a comprehensive software platform.

Constellation uses the UML diagrams to help specify the system and elaborate requirements.
What is a software platform? A software platform includes a software framework and tools to develop applications based on the framework.

Constellation integrates many tools. These include graphical tools for UML diagrams, data monitoring, performance testing, memory analysis, source-level debugging, function/method tracing, live state-machine animation, scenario capture and replay, and test scripting. Some of these tools are from RTI and others are best-in-class tools from other vendors.

At the core of the Constellation toolset is the Constellation developer tool. It is the “IDE”. It provides the graphical diagrams for developing applications. It is the launching place for all the integrated third-party tools. Developer has a complete compilation and build system, code and diagram level debugging, and a customizable integration with revision-control software tools.

Constellation is great for run-time test and verification. You can peek and poke at any component, data, or logic in the running system. All the same run-time facilities are available with any target platform, whether it is a local Windows®, Unix or Linux® simulation or a remote real-time application running VxWorks™. This makes the task of debugging, testing and integrating the system much easier and faster.

Constellation includes a framework that specifically targets control and simulation systems. A framework is a combination of pre-written software that accelerates application development. It includes an object hierarchy, a pre-defined structural framework, and pre-written libraries that form the basic program structure. Teams use the framework to coordinate their efforts, build and test software and hardware, and deliver a working application. If you select the right framework for your application, you start the project halfway done.

Using a pre-built, proven structure is certainly important. However, using an application framework provides other benefits as well. For instance, the framework enables reusable software components.

Constellation includes an integrated component repository. Team members use the repositories to share work. The Constellation developer tool is the graphical environment for developing, managing and using the components. Programmers write new components and store them in the repository. Engineers put together complex control applications with reusable components in the repositories. The repository-centric view of the world provides a reuse mindset that pervades the entire project. The real power of an off-the-shelf framework is that using an accepted structure enables you to leverage targeted tools and powerful run-time services.

Constellation is the complete software platform with a comprehensive toolset and field-proven framework for control systems.

**Simulink is an Integral Part of Constellation**

RTI has carefully integrated Simulink and RTW into Constellation. Simulink diagrams coexist as blocks in Constellation developer diagrams. Most Simulink diagrams are fully integrated, including discrete and continuous dynamic blocks and Stateflow diagrams. To edit a Simulink diagram, just open it from the Constellation developer tool. Changes to the Simulink block, such as changes to the number and type of the input and output ports, automatically flow through to the Constellation component without any user coding. Many Simulink diagrams, and even multiple instances of the same diagram, can be included in a single Constellation system.

With this tight integration, you can easily use Simulink to develop your physical models, prototype the system and develop your control algorithms. You can then include those models in any Constellation system. In fact, you may never have to look at any source code at all. The integration is seamless; you use each tool for what it does best.
A Typical Process
Constellation with Simulink fits naturally into your team’s process. It manages each of the steps, from system analysis and prototyping through test and integration to maintenance and upgrades.

Constellation’s top-down, bottom-up methodology makes it easy to combine complex software with sophisticated controls. You start development with a single block that represents the entire system. Next, you break the system—from the top down—into functional or physical subsystems. When the functionality of subsystems is clear, you build the subsystems—bottom-up—from reusable components. The components can be diagrams, sophisticated custom code, or Simulink diagrams. The single Constellation developer IDE makes all these easy to access graphically.

Let’s examine the stages of a typical development process for a large aerospace system, and see how Constellation with Simulink helps.

Prototyping
Simulink makes it easy to test and validate an accurate model of the system. Using the collection of linear and nonlinear components, you can build and simulate models of plant, sensor, and actuator dynamics. With Constellation, the prototype can include the overall software structure, including interfaces between major functional modules. Many systems can be prototyped quickly by reusing previous code modules or selecting from RTI’s libraries.

System Specification
Once the prototype is complete, most large projects start with systems analysis and specification. Working from high-level requirements, systems engineering breaks the problem down into manageable pieces. Constellation’s UML diagrams help you capture your use cases and elaborate them with sequence diagrams and class diagrams. You begin to break the system down into subsystems and break the high-level requirements down and assign them to the subsystems. You can use a requirements tool to track and later trace these requirements.

System Design
During this process, Constellation allows you to develop well-specified interfaces between the subsystems. You then use the formally defined interfaces to communicate the design to the other teams. The handoff is graphical and seamless. Better yet, all interfaces are actual code specifications that can be enforced, verified, and maintained.

The controls team will use Simulink/MATLAB to design the control algorithms for their subsystem. With the plant model, you can test your closed-loop system in simulation to determine how well the design meets performance requirements.

Once you have a Simulink model that you’re satisfied with, the model is incorporated in your Constellation diagram. You can launch the Simulink diagram from the Constellation diagram. Multiple models are easy to support; just hook up the Simulink diagram inputs and outputs to other subsystems and components in the Constellation diagram.
Implementation

Once the system design is available, the teams need to implement the subsystems. The various teams can use Simulink and Constellation to develop the code. For those groups providing subsystem components designed with Simulink, you generate C code directly from the Simulink block diagram using RTW. The code is automatically integrated into your Constellation application.

Software component development in Constellation is very flexible. You encapsulate all your software in components. For instance, you can develop components to communicate over a network or some other communications channel (like telemetry and command on a satellite). You can write device drivers to interface to the hardware. You can also easily wrap legacy code and interfaces to pre-existing subsystems. You can build a subsystem from pre-existing low-level Constellation components. You can start with the components RTI ships with Constellation (including source) and build up your own in-house repositories too. The well-defined interfaces help ensure that everything works together.

Simulation and Testing

You can, of course, easily simulate the Simulink-designed components. Then, when you are ready to test with the full application, simply treat the model as a component within Constellation.

The larger system is also easy to simulate. Constellation supports real-time, hardware-in-the-loop (HIL) simulation with “mix and match” software and hardware subsystems. You can easily simulate with a software model of a sensor implemented as a Simulink plant today, then swap in a device driver implemented as a Constellation component and test against the actual hardware tomorrow.

Of course, the rich testing and debugging environment is critical at this stage.

Integration, Delivery, Maintenance

Finally, you need to deploy the software on the deliverable hardware and do final testing and integration. When you deliver the system, Constellation’s tools help you install and debug the application. All the debugging tools available in the development environment, including the command-line shell, data monitoring, and state-machine tracking, are available on the running system. You can use your familiar environment to iron out the last-minute site-integration bugs. The tools can even be used remotely, so you can help debug from the home office.

Constellation also makes it easy to maintain and upgrade the system in the field. To change a piece of the code, just swap out a component. The debug tools are always available. Maintenance is much easier with assurance that the actual system matches the Constellation graphical design and automatic documentation.

Conclusions

Constellation is a complete software platform for developing and integrating complex control systems. It excels at coordinating the efforts of teams developing complex software systems that include control systems.

Simulink is a high-fidelity control design tool. Simulink is the best tool for simulating physical subsystems and developing control algorithms and logic.

Constellation with Simulink helps teams design, create, and deploy complete systems faster. Constellation combines best-of-class tools for the UML-based software development process with Simulink’s best-in-class, control system development process. Constellation bridges the gap between the software and controls and streamlines the entire system development process.