VIRTUAL PROTOTYPING AS A MECHANISM FOR SIMULATION-BASED DESIGN

Roger A. Dougal, Blake Langland, Antonello Monti
University of South Carolina
Columbia, SC 29208

Abstract

The concepts of *simulation based design* and *simulation as specification* require support from appropriate computing tools. The Virtual Test Bed software continues to evolve to provide this support. Features of the software that support early-stage top-down collaborative design, in-situ and ex-situ execution of subsystem models at independent time steps, and variable levels of resolution are described here.

1. Introduction

Simulation-based design requires more than just "a simulation tool." Depending on the complexity of the system, the design process may require many tools, many ways to link those tools together, and accessibility of those tools to many people. Ideally, the toolset should support the entire process, not just independent steps in the process that must then be manually woven together[1]. The Virtual Test Bed software[1] attempts to fill a void in the engineering toolset by being an integrated toolset and also by serving as a development platform for new tools. Several papers presented at this conference address very specific features or capabilities of the software. The purpose of this paper is to provide a high-level overview of the VTB environment and to describe how it endeavors to support the mission of simulation-based design and the use of models to specify a system.

Most of the discussion presented here applies to the version of the software that we call VTB Pro. This is the next generation of the software that is expected to become widely available in the fall of 2007, and which particularly attempts to address enterprise-level simulation requirements. VTB Pro is founded on VTB 2003, which continues to serve as a test and development platform for new capabilities, and which is freely distributed from the web site listed in ref [2]. The most significant difference between VTB 2003 and VTB Pro, is that the latter does not yet incorporate the realtime capabilities that are supported by the VTB 2003 environment.

2. System Engineering: Simulation Requirements

Large systems are generally designed using a top-down philosophy that decomposes the large system into smaller subsystems. These subsystems may be further decomposed (often even successively decomposed) along both disciplinary and functional lines, until work tasks ultimately become manageably small. Along the way, these work tasks are assigned to appropriate technical resources (people, vendors, partners, etc.) A simulation-based procurement process should account for these steps in the system decomposition and the resulting breakdown of work amongst the design team.

After the system is decomposed, the resulting subsystems are generally designed in isolation, but then reviewed together at predetermined times set by the design review schedule. The requirements for each subsystem are documented, and the interfaces to other subsystems are defined. Although this process allows a very complex problem to be segmented into smaller solvable pieces, it also tends to prevent engineers from understanding the capabilities and limitations of each individual subsystem and the impact of design features of one subsystem on another subsystem. A simulation-based design process allows for decomposition of a complex problem but it also must support the understanding of the cross-linking process upon re-integration of the subsystems.

The developer of a model will typically view the model from the engineer's perspective with an emphasis on the fidelity of the simulation results. It is likely that the use of any models from an existing model database will not adequately capture or convey all the information needed at each phase of the design process from concept to manufacturing. It is therefore important that models be extensible so that they can be refined with additional information as the design evolves, thereby enabling the model to act as a complete specification of the design. The iterative design process requires that simulation models be annotated with additional information that, although not directly associated with simulation, are nevertheless imperative in a complete specification.

The simulation environment that supports the design process should be flexible enough to disaggregate a complex system into any number of smaller pieces and conversely to aggregate independent objects into a complete representation of the system. It should, as much as possible, reduce the degree of complexity for the concept designer. This means that a graphical interface is essential, that energy couplings between system components should be automatically and transparently handled, that existing models should be reuseable, and that rapid iteration of the design cycle and incremental refinement of the system should be supported on a group-wise basis [3]. Additional essential features include means for

- Rapid definition of models.
- Rapid optimization of subsystems and systems.
- Rapid substitution of refined models for coarse models, and vice versa.
- Simulation across an often large range of time scales.

- Collaboration within and among large design teams that are often divided along disciplinary lines.
- Co-simulation between discipline-specific design tools.
- Extraction of reduced-order models and model parameters from high fidelity models.
- Methods to proceed directly from a simulation model that specifies a system to a specific hardware or software realization of the system.
- Reversible processes across levels of abstraction, since design is an iterative process.
- Processes that manage the work and information flow within the design organization.
- Data sharing between disparate applications via an open platform.
- Different types of solvers and tools to be defined and added to the extensible framework.
- Models that allow addition of new information as the design evolves, even if that new information is not necessarily relevant to the simulation per se.

Our purpose in this paper is to describe at a high level how VTB supports these activities. The reader is referred to other papers for more in-depth descriptions of individual capabilities.

3. VTB Support for System Engineering

3.1. Rapid definition of models.

Especially during the conceptual phase of design, a capability to quickly create simulation models is essential. Any models developed early in the conceptual phase will likely be minimally complex in order to simply convey the general idea, to reduce the time required to configure or reconfigure the conceptual system, and to rapidly execute many simulation experiments[4]. Rapid exploration of the design space with minimal time investment is crucial. As the various conceptual designs are examined, the engineer can quickly focus on the set of most-promising designs; those that will require additional elaboration and examination. Tools to quickly create and deploy models for use within the simulation and design environment are valuable.

VTB ships with such a tool, called Entity Builder, that allows rapid creation of models that capture specification information. This tool permits the engineer to create the specification for the component, to define configuration parameters, terminals or points of connectivity with other components, to define its symbolic or iconic representation, and to define the behavior of the component during simulation. VTB allows components to define their behavior at simulation time in one of two ways - either by the use of an interpreted language that is automatically compiled at run time, or by supplying an assembly that provides the necessary information for simulation. The modeling language provides a convenient and rapid way to prototype a model [5]. After the component has been created in this tool it can be deployed with the single push of a button and then used in the drag and drop environment of VTB.

In some situations a new component can be defined by merely repackaging existing components in some pre-arranged configuration with an established set of parameters. VTB ships with a tool called Module Builder that allows engineers to quickly define new components by assembling and configuring existing components into reusable objects. These components (modules) can then be used in the drag and drop environment.

3.2. Optimization of systems and subsystems

An effective design and simulation tool should provide a capability for engineers to extract and compartmentalize parts of the design for purposes of futher development, definition, and optimization. Subsequent to those improvements, there must be mechanisms to re-integrate the elaborated design elements into the larger system model in order to then address any remaining optimization issues that span disciplines, multiple subsystems, or the system as a whole.

VTB permits the system design to be broken into as many subsystems as necessary in order to drive the entire design to completion. These subsystems can be simulated and optimized separately or coupled together and examined at a more global level. The points of tangency between the subsystems are defined by the use of subsystem connectors. These objects act more or less like plugs that can be connected and disconnected with a mouse click. Subsystem connectors act as natural points of separation between subsystems, and can even be used to break out pieces of a single subsystem. In this manner a system can be broken down into a set of modular subsystems that can be designed and tested independently, yet quickly coupled together for simulation and optimization at a more global level.

3.3. Model complexity

During the conceptual phase of a design many possible solutions must be explored as quickly as possible, so that the search space can become focused on a small set of promising designs. These early-stage conceptual designs often make use of idealized representations of system components; great complexity is not needed in the model, since much is unknown at this point in the design. As the design is explored further, the level of detail in the design increases and the complexity of the model likewise increases The simulation and design environment should provide a capability for the engineer to use models of varying levels of fidelity based upon the current maturity of the system design.

In VTB components have just as much complexity as is assigned to them by the developer The degree of complexity may indicate to the system designer the amount of realism incorporated into the model at simulation time. Generally, the

higher the complexity levels the more sophisticated and accurate a model may be - typically at the cost of a longer simulation execution time and an increase in the time required to fully specify the model's parameters. It is often the case, when moving from a lesser to a more complex model, that the system engineer will be required to supply more data either in the form of model parameters or possibly via coupling the component to existing or new components. For instance, a power converter with a complexity level of 1.0 may not incorporate any thermal characteristics, but a power converter with a complexity level of 2.0 might. When the designer replaces the simpler converter model with the more elaborate model, the designer will also have to specify thermal couplings to other components, or default boundary conditions that apply otherwise. In VTB, the user can examine a set of related components and choose the component with the most appropriate complexity level for the current phase of the design [6].

As the complexities of components increase, the execution time of a simulation also increases. Thus it is useful if the engineer can specify the level of sophistication needed in the simulation model at run time. In VTB under certain conditions the simulation engine for a model may be swapped out at run time with another simulation engine. This is a complex operation (at the software level) that requires that the simulation engines being swapped have intimate knowledge of each other and that they know how to handle the current and past values for all state equations. The advantages to such an operation are that the user can determine the level he or she is interested in achieving during a simulation and can manage this with respect to time constraints.

3.4. Multi-rate simulation

As system complexity rises, generally so does the simulation execution time; at some point the simulation time beomes cumbersome. While many strategies exist to deal with this, the proliferation of multicore processors and cluster computing make distributed computing an increasingly attractive solution.

Distributed computing has performance advantages over single-processor computing, and is conceptually consistent with a simulation-based-design philosophy in the sense that independent subsystems are "owned" by those who are responsible for them, and should be independently computable, either with highly-abstracted boundary conditions when the subsystem is tested outside of the larger system context, or with boundary conditions that result from direct interaction with the larger system. Distributed computing faces challenges due to the strongly-coupled nature of electrical networks but several methods support decomposition of large systems into smaller subsystems. These subsystems may have different dynamics but where the system as a whole would have its time step determined by the subsystem with the fastest dynamics, for both performance and strategic reasons, these independent subsystems should be computable with independent time steps. Providing the ability to simulate subsystems at time steps appropriate to their individual dynamics can provide us with greatly reduced run times at the cost of a miniml loss in the simulation's overall accuracy. Demonstration cases of independent time stepping in subsystems have been developed, and a more general capability for this is under development [7].

3.5. Collaboration

As the complexity of the system increases so does the need for engineering teams with specialized skills and expertise. Each engineering team is responsible for satisfying the design requirements of one or more subsystems. The ability to decompose the problem into smaller solvable pieces is critical to parallelizing the effort and making rapid progress on the overall system design. There are several consequences, both good and bad, to the decomposition of a system. Subsystems, once disaggregated, can be designed in relative isolation, which permits rapid evaluation of a subsystem but may not fully allow the engineers to understand the constraints and complexities that arise upon connection to other subsystems. Documentation of the requirements and interfaces for the various subsystems can help alleviate interface problems but documents themselves do not offer facile ways of handling change. In addition, other requirements or interests may not be completely known and understood in advance such as production, maintenance, and usability issues, especially during the conceptual phase. In a typical design environment subsystems tend to be viewed as systems unto themselves without interdependencies. Artificial barriers to inter-team communication can be erected during the decomposition process leading engineering teams to completely miss the impact of certain design changes. A design and simulation tool should address these issues so as to permit an environment that fosters true collaboration among all team members, including those not directly involved in the engineering process. Such an environment would allow all stakeholders the ability to reconcile design tradeoffs in an optimal way across the system as a whole[8].

VTB permits a complex system to be decomposed into subunits, but also provides an environment in which to resolve the problems introduced by this very decomposition. Even though the larger problem may have been broken down into its constituent parts, still, all team members are continuously informed of changes to the design of other subsystems with which they must interoperate. This permits everyone involved with the project to review the current state of the design at a level most appropriate for them, from management, to engineering, to manufacturing, to the end users of the final product. Issues and concerns can be raised and addressed sooner, reducing costs and development time. When a change is made to the design of any one subsystem, the side effects to other subsystems become immediately apparent to everyone involved. Optimizations can therefore cross various disciplines such as thermal, mechanical, and electrical, as well as subsystems, thereby preventing too narrow a focus.

In today's global economy it is more likely than not that the physical location of team members will be geographically dispersed either because the organization has a global

presence with expertise in various locations or because certain parts of the design are outsourced or subcontracted to third parties. In these situations the lack of an integrated and cooperative environment can seriously impede progress. VTB provides a single repository for the entire system design and expresses that design in an open format so as to permit it to be digested by other applications. The design is captured in an industry standard XML (Extensible Markup Language) format which permits any third party application the ability to intercept, extract ,or modify the contents of the design at any point in the design cycle. Data expressed in this XML format is easily translated to and from application specific formats via any one of the many free XML parsers available. In this integrated environment the design activities for all subsystems can move forward concurrently.

VTB provides an environment in which the system can be viewed as a collection of interconnected, interdependent, and reconfigurable subsystems. In this environment all users have an immediate understanding of the intent of the subsystem design, its limitations, how it is to be operated and how it should be tested. Introduction of change is understood more completely and at an earlier time in the design process reducing the overall costs associated with design change. Optimization, rather than occurring just on a local level, can be at levels above a single subsystem. This leads to a design which delivers better quality at lower cost.

3.6. Co-Simulation

There are many simulation tools available and used in the design of complex systems, primarily because each addresses only very specific needs within the many disciplines involved. It is also unlikely that any one simulation and design tool will adequately meet the needs of all users. For this reason it is important that the design environment be capable of integrating at some level with other design and simulation tools.

VTB addresses these issues in two ways. First, by providing an open architecture, VTB permits the extraction of information from systems defined in VTB. This data is expressed in an XML format with an XSD (XML Schema Definition) that allows other programs to translate and consume the system definition. Second, VTB provides wrappers that permit components defined in other environments to be simulated within the VTB environment.

For example, wrappers permit integration with components defined in Matlab / Simulink (primarily used for controls). This can be accomplished via co-simulation through a COM interface as defined by Matlab or by generating native VTB - compliant components through the Simulink Importer tool provided by VTB[9][10].

Other wrappers have been developed for Simsmart's SmartCADE tool for fluid simulations. This wrapper permits the usage of fluid components defined within SmartCADE to be used directly in VTB in a co-simulation mode. Yet other wrappers have been developed, or are in development, for interactions with ACSL, ESL, and SPICE.

Components that support general communication with external applications are also available. These components permit the exchange of data with third party tools or custom applications through inter-process communication mechanisms such as sockets and UDP. There is also the capability to read data from and write data to well known formats such as Excel.

Integration with other applications is permitted through the well defined interfaces of VTB using COM or any .NET compliant language. These interfaces permit existing systems to be modified such as making topological changes, component configuration changes, and solver setting changes. In addition, the solver can be controlled externally. This permits another application the ability to step the VTB solver at whatever level of granularity is required [11].

3.7. Moving from high fidelity models to reduced-order models

The ability to move seamlessly from a complex component to a simplified version of the component can be useful when executing a simulation. In cases where the user performing the simulation is not as interested in the fine details of the analysis but would rather decrease the time required to perform the simulation, having the ability to execute with a reduced-order model is important.

VTB addresses this by ensuring that all components have metadata which fully describes their state and allows for a single component to have multiple simulation engines. These simulation engines can be swapped out during simulation time by the user. In such cases the internal state of the component is passed from one simulation engine to the next. In the case of moving from a higher-order to a reduced-order model the reduced-order model can easily transition by examining the state and either ignoring some information or integrating the state variables as appropriate. In the case of moving from a reduced-order model to a higher-order model the transition is not so simple. A higher-order model has more states and so the reduced-order model cannot provide adequate initialization information. In these cases the transition simulation engine might only permit switching to the more detailed model under certain conditions where the additional state information can be assumed. Currently in VTB there are only a few models which permit the user to switch simulation engines, as the operation can be quite complex, but general procedures for this are being developed [12].

3.8. Model as specification

As the design of a system moves from the conceptual phase to detailed design to manufacturing additional information will likely be required at the component level which may or may not be relevant to all parties involved. This information may not be relevant for simulation purposes, but may, for example, add to the overall understanding of the subsystem from others involved with manufacturing, maintenance, etc… The design environment should provide

the ability to annotate the models with such additional information as it is discovered.

VTB permits custom attributes to be created and associated with model types or instances. These attributes are strongly typed and given particular values when associated with instances of a model. This information is not used during the simulation of the subsystem but can be extracted and used by other tools. Queries can be made of the system as a whole or from a subsystem to retrieve these custom attributes for additional processing by some third party tool. A possible application would be the creation of a manufacturer and part number attribute and its association with a particular component. This information could later be extracted in the manufacturing phase to produce a complete parts list for assembly.

3.9. Processes management

A design and simulation environment should address some of the process management and work flow issues that arise when designing large systems. There are many artifacts created during the design process, and it is imperative that management have a complete picture of the design, know where all design documents reside, control access to such documents, and track changes to them. Without such a controlled process it is impossible to know the precise state of the system's design or to effectively manage the design process.

VTB helps manage the change process for design documents as well as offering security for these documents. Once a system is decomposed into its constituent subsystems, each subsystem is treated as a separate document with its own set of permissions. Groups or individual users are given permissions to view, modify, and execute each subsystem document. Read and write permissions are also granted at the system level. In this manner access to the design documents are restricted to users that are authorized to make changes or to view the document. For instance, a user without read permission at the system level is not authorized to view the system or any of its subsystems regardless of any permission granted at the subsystem level. Individual user permissions constitute the union of permissions granted to the specific user and any permission granted to the groups the user is a member of. As modifications are made to the design document, VTB records who is making the change and the date and time that the change is initiated. Although not currently implemented, an additional feature currently being integrated will provide the ability to track versions of the design document and permit rollbacks to earlier design save points.

In order to ensure that the document remains in a consistent state, users check out the document or subsystem they wish to make changes to and then, once the change is completed, they check the document back into the central repository. Once the document is checked in, all other users who have permission who care to be alerted of such changes are informed that their version of the document is out of date and it is automatically updated. While a document is checked out, no other users may make modifications to it. This ensures that the design document remains in a consistent state and that change modifications are not lost. The administrator has the ability to remove such a lock on the design document if deemed necessary. In this case the user who has the document checked out would lose all of his or her changes.

The functionality of VTB can be quite different depending on the particular role the user has been assigned in this environment. There are administrators who have access to and control of all design documents, data, and components that are defined in the repository. Administrators are responsible for assigning users to roles and can revoke or grant access to all documents. Users in the component developer role are allowed to create new components or modules for use by the system engineers in their designs. A component developer creates a new component and after it has been adequately tested makes it publicly available to system engineers for general use. System engineers would be the standard users who are permitted to create or modify systems and subsystems.

3.10. Open platform for data sharing

When working on the design of large and complex systems a number of tools will likely be needed. These tools are generally created by the vendors without the capability to expose their internal data and data structures in a way permitting integration with other applications. Significant time can be lost and errors introduced if the engineers are required to manually re-enter data into other applications. Compounding this problem is the fact that the design no longer resides in a single location, but bits and pieces of the design are spread out across many applications without the possibility of reassembly into a complete view of the entire system design.

VTB was created fully expecting that users would want to extract or insert information from systems and subsystems into VTB programmatically. The underlying data structure of the system design and the schema for understanding this data structure are both industry standard formats called XML and XSD respectively. Use of this standard permits the design documents to be analyzed, processed, or modified by any application simply by performing a transformation from the standard XML format to the custom format of the application doing the processing. An interface to VTB exists that permits other applications to extract, manipulate, and insert information directly into systems and subsystems defined within the VTB environment. Third party applications can simply link to the framework objects in VTB directly and perform programmatic queries as well as modifications of the system design programmatically. VTB supports a complete view of the current state of the system design by providing a single repository for all the design documents of a system.

3.11. Extensible framework

The architecture of VTB permits extensions to its solving capabilities. New solvers can be defined and plugged into the

existing VTB framework. Likewise, components that take advantage of these new solvers can also be added. A solver must be compliant with some basic requirements, as defined by interfaces in the VTB framework. Once a solver has implemented the basic interfaces, the solver can then be loaded and used within the VTB environment. A solver also defines the requirements of any components wishing to leverage its services. The solver defines the required behaviors, and component developers then implement these behaviors. Typically, the developer of the solver will also supply software modules that help facilitate the component developers' efforts to implement the newly required behaviors. In this manner VTB can be extended beyond basic simulation capabilities in such a way as to handle additional duties such as managing uncertainty and quantifying risk in a design **Error! Reference source not found.**.

4. Conclusions

We have presented an overview of the VTB simulation environment and some of its unique capabilities. The VTB environment addresses issues across all stages of the iterative design cycle from concept to production to maintenance. In particular VTB helps to address problems that arise in large scale system design involving multiple design teams from various disciplines. The VTB environment provides management, work flow, and collaboration services that provide an integrated view with immediate feedback on the state of the entire system design. VTB provides an open and flexible environment which permits integration with any third party application that chooses to provide or consume data in an XML format. Capabilities exist within VTB to rapidly define new models and to extend these definitions beyond the interests of system engineers so as to permit the model to become the specification for all interested parties. The extensible architecture of VTB makes it an ideal platform for adding new capabilities and performing research.

References

[1] S. Robinson, M. Pidd, "Provider and customer expectations of successful simulation projects", Journal of the Operational Research Society 49 (3) , pp. 200-209, 1998

[2] The software is available at http://vtb.engr.sc.edu. Numerous relevant conference presentations and papers are also cataloged on that web site.

[3] R. A. Dougal, "Design Tools for Electric Ship Systems", IEEE Electric Ship Technologies Symposium, pp. 8-11, Philadelphia, PA, July 2005.

[4] J. Walrath, K. S. Chatha, R. Vemuri, N. Narasimhan, V Srinivasan, "Performance Modeling nad Tradeoff Analysis During Rapid Prototyping", IEEE Proceedings of the 1996 International Conference on Application-Specific Systems, Architectures, and Processors, pp. 313-322, 1996

[5] R. Dougal, Solodovnik E., "Symbolic Modeling of Non-linear Devices for the Virtual Test Bed", IEEE Power Engineering Society General Meeting, pp. 1553-1555, San Francisco, CA, June 2005.

[6] A. Monti, R. Dougal, F. Ponci, "The Incremental Design Process for Power Electronic Building Blocks", IEEE Power Engineering Society Annual Meeting, June 2006

[7] V. B. Dmitriev–Zdorov, M. N. Maksimov, V. P. Popov, J. Bastos, A. Monti, R. Dougal, "Generalized Coupling Scheme for Distributed Simulations of Power Systems", Transactions of the Society for Modeling and Simulation International, March 2006

[8] S. Taylor, "NetMeeting: A Tool For Collaborative Simulation Modeling", International Journal of Simulation, Vol 1 No 1-2, pp. 59-68, 2005

[9] Monti A., Santi E., Dougal R., Riva M., "Rapid Prototyping of Digital Controls for Power Electronics", IEEE Trans. On Power Electronics, pp 915-923, May 2003

[10] Dougal R., Lovett T., Monti A., Santi E., "A Multilanguage Environment For Interactive Simulation And Development Of Controls For Power Electronics", IEEE Power Electronics Specialists Conf, Vancouver, pp 1725-1729, 2001.

[11] F. Ponci, A. Deshmukh, L. Cristaldi, R. Ottoboni, "Interface for Multi-agent Platform Systems", IEEE-Instrumentation and Measurement Technical Conference, Vol.3 pp. 2226-2230, Ottawa, Canada, May 2005.

[12] Wu, H., Dougal, R., "Dynamic multiresolution modeling of a power supercapacitor" North American Power Symp., pp 241-246, Ames, 2005.

[13] A.Monti, F.Ponci, T.Lovett. A Polynomial Chaos Theory Approach to Uncertainty in Electrical Engineering, IEEE Intelligent Systems Applications to Power Systems, pp. 534-539, Washington DC, 2005

Acknowledgements

This work was supported by the US Office of Naval Research under contract N00014-02-1-0623.