SPEEDSIM.NET - AN OPEN .NET BASED SIMULATION SYSTEM

Wiedemann, Thomas [a], Wendt, Karsten [b]

[a] University of Applied Science Dresden
[b] DUALIS GmbH IT Solution

[a] wiedem@informatik.htw-dresden.de, [b] KWendt@dualis-it.de

ABSTRACT

As a result of the contradictory characteristics of existing simulation systems, there is a gap between component based simulators and programming language based simulators. This paper will discuss possible approaches and tool candidates for filling this gap.

The presented approach and the developed simulation system SpeedSim.Net are flexible and powerful options for defining future, complex simulation environments. The system can be used in a standalone mode inside a larger software system or together with a predefined GUI-framework like a traditional simulation system. All components are managed by tools of the Visual Studio, like the property editor or tree list dialog boxes, so experienced VS-developers can start modeling very fast.

Interested simulation experts can participate in the future development of the tool.

Keywords: .NET component based standalone

1. INTRODUCTION

The main algorithms and mathematical foundations of discrete simulation systems are well defined and efficient (Wiedewitsch and Heusmann 1995) (Wiedemann, T., 2002), (Zeigler, B. P. (1990). Nevertheless, the real application of simulation systems is still difficult (Kuljis and Paul 2000), because there is no optimal system. Especially the modelling comfort and the performance and software flexibility are contradictory, comparing different levels of existing types of discrete simulation tools (see colored triangles in fig. 1).

As a result of the contradictory characteristics of the simulation systems, there is a gap between component based simulators and programming language based simulators. This paper will discuss possible candidates for filling this gap. If such an optimal system could be defined, it should combine all the good characteristics of the different system classes (see right site of fig. 1).

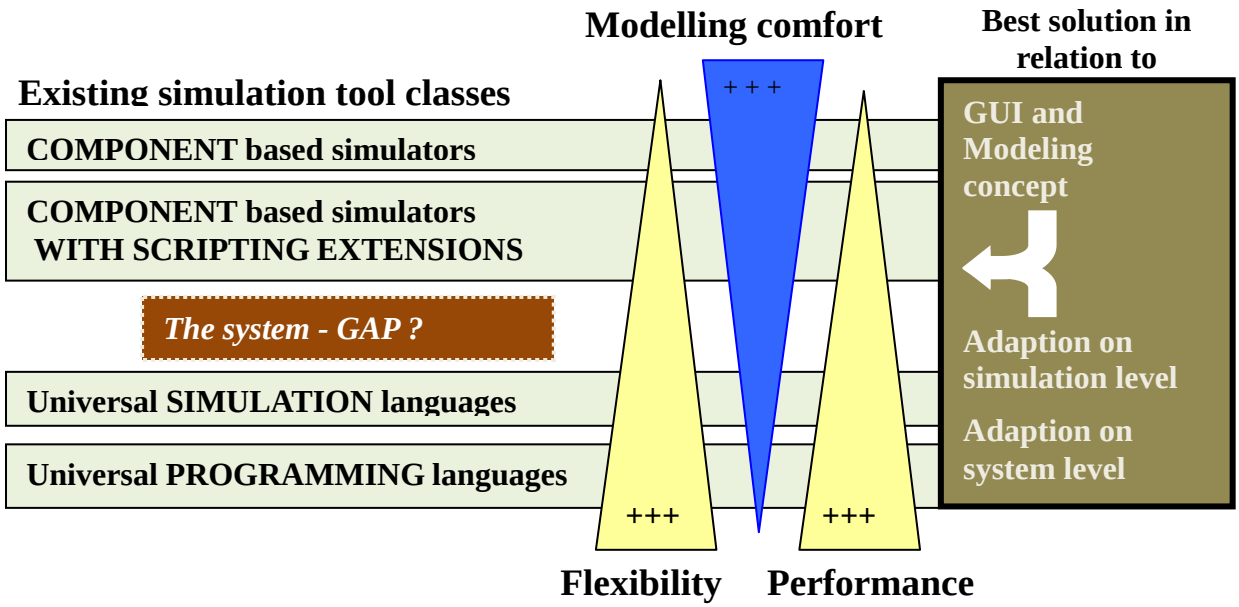

Figure 1: The actual system classes in discrete event simulation and their main characteristics

2. DEFINITION OF MAIN SYSTEM REQUEIREMENTS

Based on the discussed characteristics of existing discrete simulation systems, we define the following requirements for an optimal discrete simulation system:

1. Use a standard programming language and its integrated developing environment (IDE) for modeling and simulation control.
2. Provide a comfortable and affordable modeling environment similar to modern component based simulation systems.
3. Use the existing standard programming language(s) for scripting at the model level.
4. Provide high simulation performance for complex simulation scenarios and long optimization cycles.
5. Apply sophisticated software technologies for defining simulation components, basic simulation functions like random number generators and statistical modeling and analysis tools (autofit tools, empirical function definitions).
6. Use databases for storing all models and model input and output data.
7. Provide state-of-the-art interfaces for data import and export and for system integration and communication.

This “wish list” will never be complete and should add new software technologies like web-based standards such as HTML5 for user interfaces or web services for system integration and data exchange.

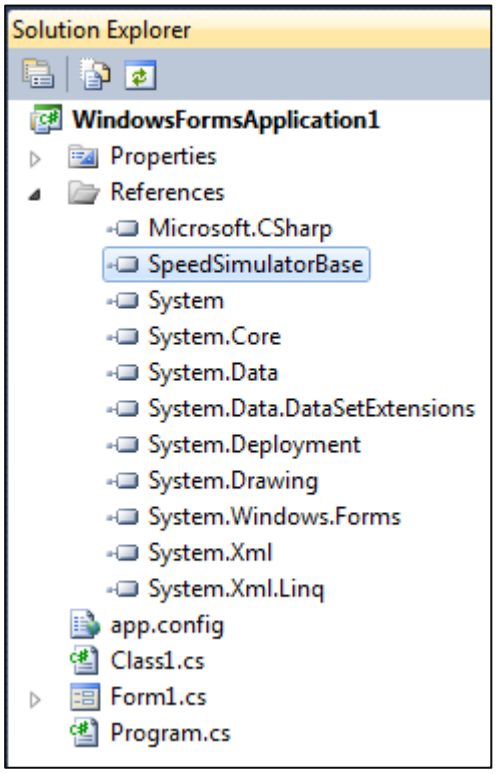

Figure 2 The software libraries

3. DISCUSSION OF IMPLEMENTATION OPTIONS

3.1. Selection of the IDE

Comparing all existing integrated developing environments, there are two main options: the Visual Studio by Microsoft (VS2013) and the Eclipse IDE from the Eclipse Foundation (Eclipse Foundation 2013). Both options have their advantages and disadvantages:

- Microsoft’s Visual Studio is a very complex environment based on the .NET-architecture and supports all relevant IT-technologies in the desktop and web area. The main problem is the focus on Microsoft operating systems, which limits the application area. But if the customer or end-user uses MS-OS, it is t a very good solution.
- The Eclipse IDE is based on Java and runs on any operating system, especially on Unix- and Linux-based systems. If a customer runs such OS, Eclipse is the only solution.

Both environments do not generate native code for the processor but some Intermediate code (MSIL / Java Byte Code). In general, this is a disadvantage under performance considerations, but the actual state of the art of code generators and the optimized code interpreters at the execution level allow execution rates at the level of native code. In some special cases the execution speed of the MSIL code was higher than the speed of native code. In conclusion, the intermediate code is no longer seen as a performance issue, but as a good option for migrating the compiled simulation model to different platforms.

The question of the overall development speed with the IDE´s was tested with two groups of students, working with Visual Studio and the Eclipse IDE on a standard Database oriented web-application. It was impressive that the Visual Studio group could finish the task in only 25% of the time and with a better end user quality than the Eclipse group. The main reason for this huge difference was the high quality of supporting solution assistants and a seamless integration in the Microsoft webserver environments. The Eclipse group finished the main Java programming in nearly the same time, but then lost a lot of time with configuration issues and run-time errors in the web server environment.

As a result of this development test and already existing .NET code at the partner software company DUALIS, the Visual Studio option was selected. In general, the main approach, which will be shown on the following pages, could also be migrated to the Eclipse IDE. In Eclipse, there would be the option for using Java for the GUI, but C++ should be used in the simulation core to get a good performance and to reuse the code form the Visual Studio version (Skeet 2013).

3.2. Selection of programming and language

Taking the selection of Visual studio (VS) into account and comparing all possible .NET programming languages for their usability, flexibility and user

acceptance, C# was the logical solution. The old C / C++ were used in an older version of the system, but they are no longer supported by the managed code system of Microsoft and there are increasing limitations and problems (Skeet 2013)..

3.3. Standalone user interface

Using the VS IDE as a basic modeling tool, a first simple .NET-like modeling is possible. All the simulation knowledge and functionality is provided by one DLL with a number of predefined simulation classes and additional supporting libraries (see fig. 2). The simulation model is created as a standard .NET-Windows form. The GUI-buttons call the simulation control methods and simulation results are displayed in the log field (see fig. 3).

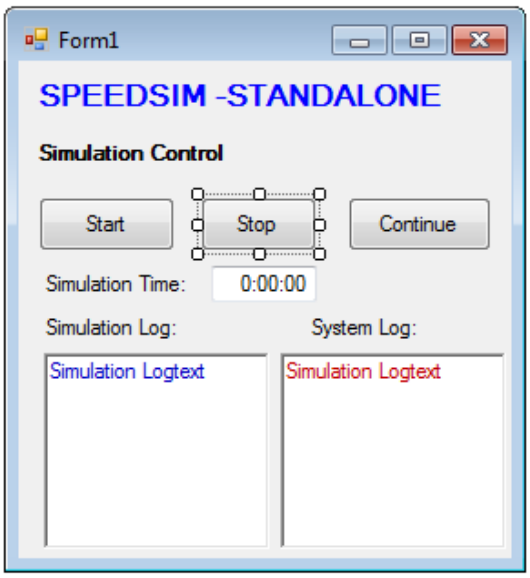

Figure 3 A simple Stand-alone simulation model in Visual Studio

The source code behind the form window of a very simple simulation model is shown in fig. 5. After defining the model, the simulation is started by calling the `_Model.ResumeSimulation();` method with the start button from the user interface.

The standalone modeling option is a good solution for small and quick simulation experiments. It takes about 15 minutes for experienced VS-developers to build a running .NET simulation model.

Further types of applications are highly integrated simulation models. The code of the model from fig. 3 could be migrated to any .NET application, like ERP-backend interfaces or business intelligence tools. By using web based technologies like web services, the input data for the simulation run and the simulation results can be distributed over the web.

3.4. A traditional desktop user interface

Simulation users without advanced programming knowledge will encounter problems with the programming orientated approach. For this group of users the system offers a second method of designing and manipulating models.

To meet specific simulation needs, an experienced VS-programmer will prepare a set of simulation components. The resulting component library is compiled as a DLL file and then a precompiled complex user interface – the SpeedSim.Net framework - is started. The simulation user will load the component library into the SpeedSim.Net framework. Afterwards, he can design and configure complex simulation models like in traditional simulation systems (see fig. 4)

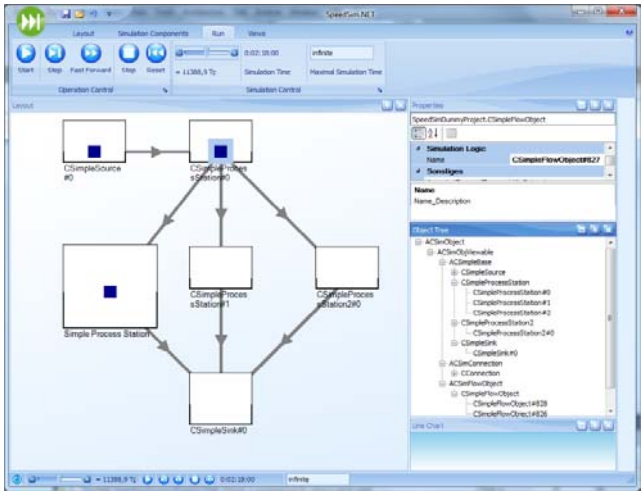

Fig. 4: The SpeedSim.Net user interface framework

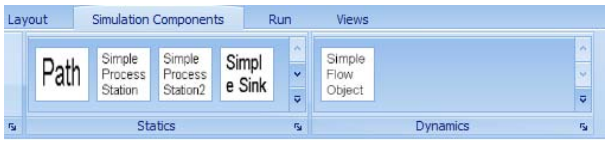

Fig. 5: The GUI-view of a loaded component library from the user component DLL

```
public Form1()
{  InitializeComponent();  // Init Form
   // Define Model for Simulation
model1=new SpeedSimulatorBase.Simulation.CSimModel();
     model1.Initialize();
     // Register SimObjects in Model
   sim1object1 = new mySimObject();
   sim1object1.Initialise(model1,true,false);
   sim1object2 = new mySimObject2();
   sim1object2.Initialise(model1,true,false);
    // Define Simulation
        simulation1 = new CSimulation();
        simulation1.Initialize(model1);#
         // Define Simulation Duration
           simulation1.MaxSimTime = 100;
           simulation1.SecoundsPerTick = 0.01;
 }
```

Figure 6: The source of a stand alone model

All model components can be managed by the well-known Visual Studio property editor (see upper dialog box of fig. 7). Newly designed methods of the simulation components are automatically part of the property list (see properties "MaxContent" in fig. 7).

During the simulation all dynamic objects like products or other moving objects are shown in the dynamic object tree list (see lower dialog box of fig. 7). The values of any dynamic object can be displayed and also changed during runtime.

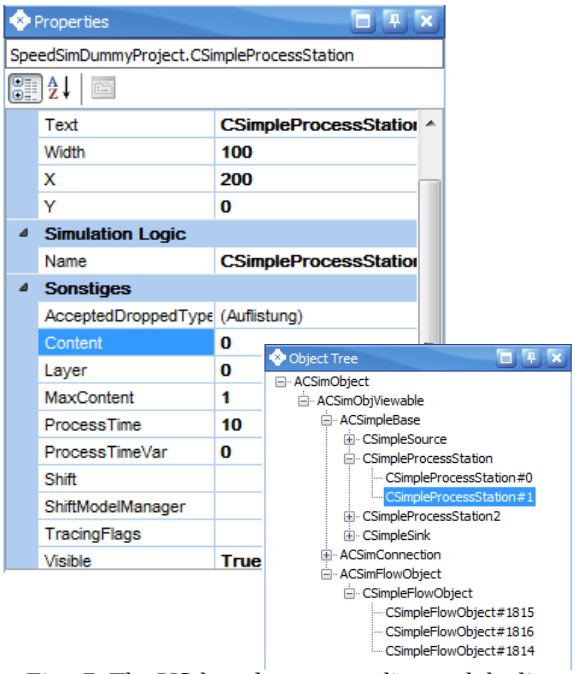

Fig. 7: The VS-based property editor and the list of dynamic object during simulation runtime

Results are written in external files and can be displayed also during run time in different chart windows, which are interactive (see fig. 8)

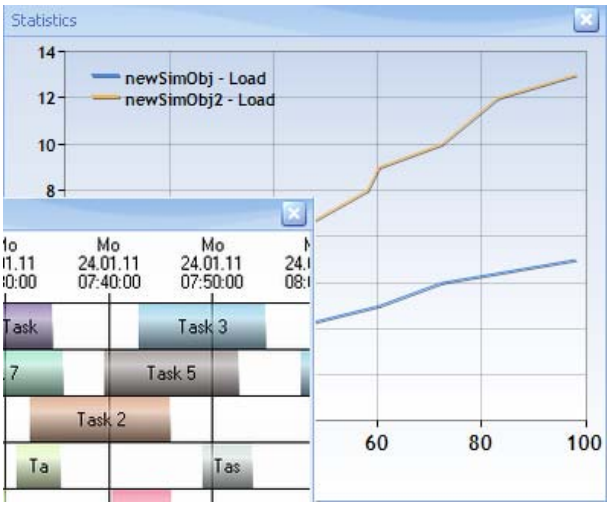

Fig. 8: Visualization of run-time statistics and results

4. THE DEVELOPERS VIEW

4.1 The SpeedSim.Net OBJECT model

The full object model of the developed simulator is shown in fig. 8. Beside the core modules for building the simulation model, there are a lot of other modules for simulation control and interfacing (see fig. 9).

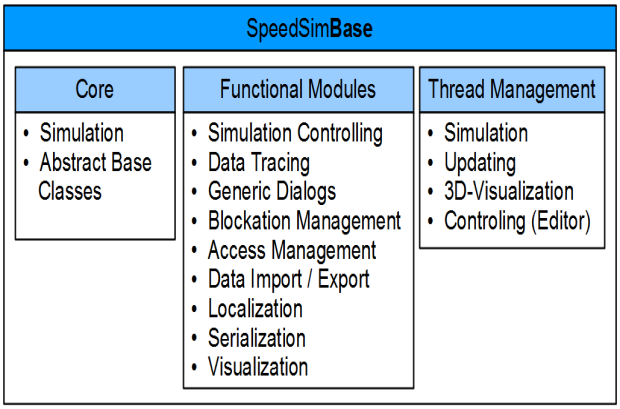

Fig. 9: The S SpeedSim.Net Object Model and basic system architecture (from (Wendt 2013))

This basic system architecture is concentrated in a single DLL. By uploading this DLL to a Visual Studio environment (VS), the VS becomes a simulation environment.
Customer specific changes to the basic algorithm could be added in two ways:

- Single changes of parameter values can be done with the SpeedSim.Net editor according to fig. 7.
- Much more complex and long-term modifications can be concentrated in customer specific libraries (see Fig. 10). It will be possible to build up a system of application specific libraries e.g. for logistics, IT-network or traffic models.

It should be mentioned again, that all changes of model or simulation algorithm will be done with one of the .NET-programming languages and the resulting project will be still a standard .NET-project! This allows the integration of all existing and future software technologies from Microsoft, like Web Services, SOAP--and database interfaces to Data Mining and Business Intelligence tools.

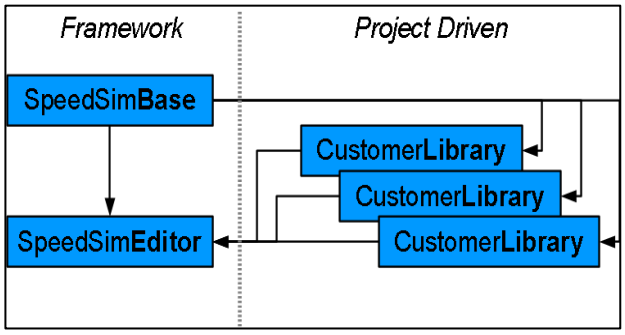

Fig. 10: Framework and customer driven project linraries (from Wendt, 2013)

The simulation object life cycle is shown in fig. 11.
After the setup phase the object state is scheduled between the active states, like “Action” or “Event handler”, and the delay state, where the object is waiting in the future event list of the simulation control for the correct simulation time to be activated again.
It is very important that all object states and their corresponding event handlers can be changed by the core or model developer. Again, changes of the algorithms could be saved in the model specific project or if useful in general also in the customer specific libraries.

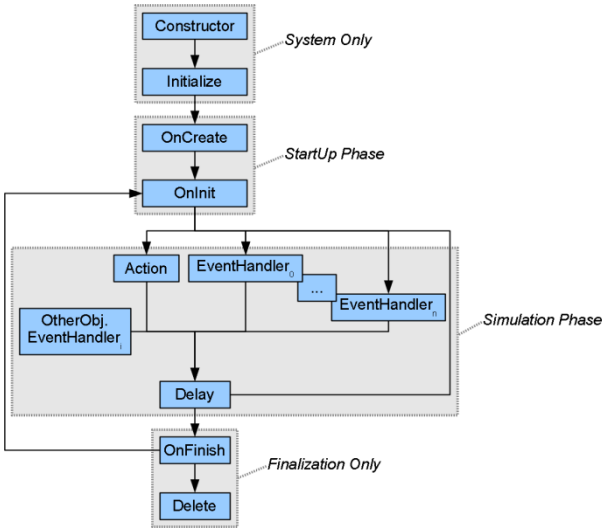

Fig. 11: The simulation object life cycle (Wendt 2013))

4.2 Optimization of visualization and animation

The SpeedSim.Net framework for non-programmers supports a multi-threaded module architecture and will also provide 3D visualization options in the future.
In order to optimize simulation speed, not all simulation events are displayed in the front end view.
Time measurements by the authors show that the time needed for displaying a 5 digit number on the screen is about 1000..2000 longer than the time for any single calculation for getting this number.
In conclusion, there should also be the option to switch off nearly all visualization options in order to achieve maximum speed. The decision of the visualization level is not coded in the simulation core, but is defined on a multi-level control structure (see Fig. 12.) The different levels of this message chain can also be changed for non- desktop purposes.
If the simulation core is running as a standalone program with no visible windows (or only an icon in the system tray), all user input and simulation output could be managed by web based technologies or web services.
Therefore, a remote simulation service is possible, which could be integrated in larger IT-environments like ERP- or business intelligence solutions.

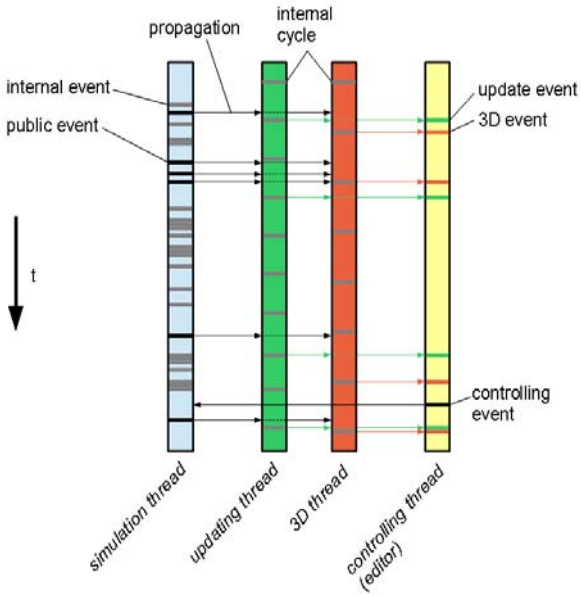

Fig. 12: Thread- and visualization Management

5. ACTUAL STATE AND DEVELOPMENT

The shown modeling options – standalone and framework based, are fully operational and could be used by experienced customers.

The system was already introduced in a master course “Discrete events simulation” at the University of Applied Sciences Dresden. The simulation with SpeedSim.Net was introduced to computer science students with one 90 minutes lecture and one double hour of student exercises. After some small, predefined example projects the students were able to successfully model and simulate a model with 6 model objects.

The actual work focusses on defining complex libraries for customer oriented production models and other typical application areas such as network and transport simulations.

Interested simulation experts can participate in the future development of the tool. Although the development is not yet finished, the shown approach and the existing tools could fill the gap in the actual hierarchy of simulation tools (see fig. 13).

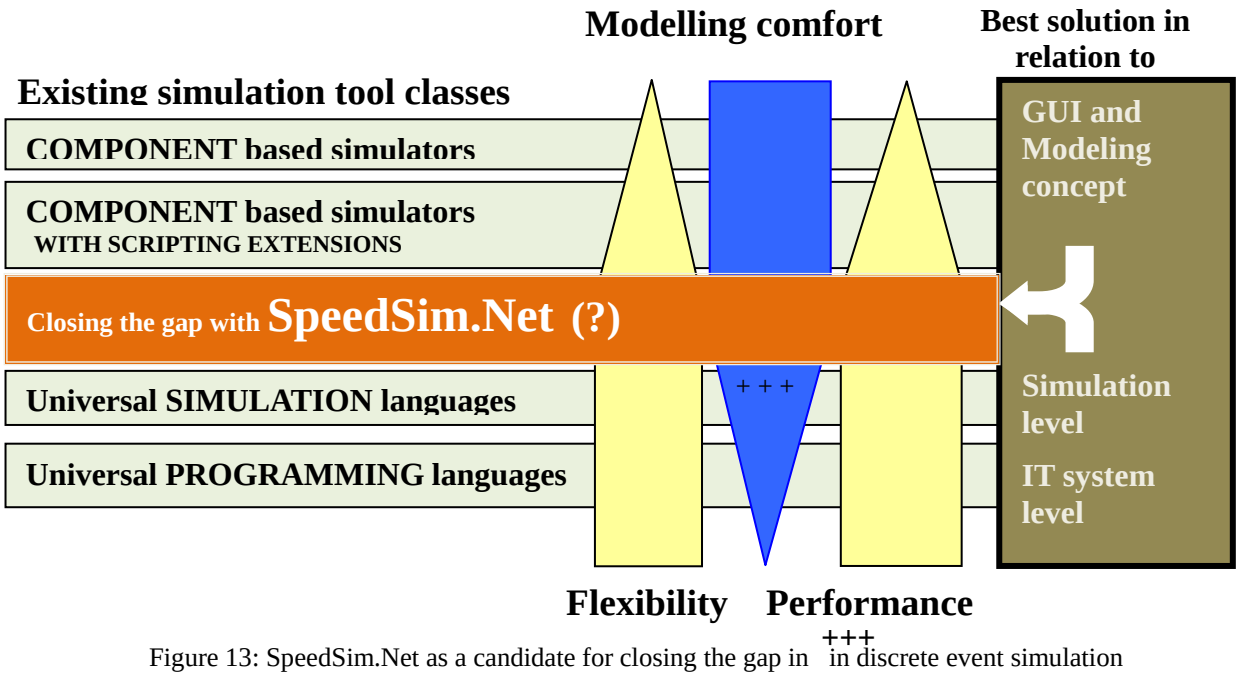

Figure 13: SpeedSim.Net as a candidate for closing the gap in in discrete event simulation

REFERENCES

Kuljis, Jasna and Ray J. Paul, 2000: A Review of web based simulation: whiter we wander?, *Proceedings of the 2000 Winter Simulation Conference*, Orlando Florida, page 1872-1881

Schriber T.,Brunner D., Smith J.: Inside Discrete Event Simulation Software: How It Works and Why It Matters. Proceedings of the 2013 Winter Simulation Conference, Pages 424-438

Skeet, Jon: C# in Depth. Manning Publications; 3rd revised edition. 2013

Wendt, K. (2013) : SPEEDSIM Documentation Version 1.3. Dualis IT Solutions GmbH Dresden, 2013

Wiedemann, T., 2002. Next generation simulation environments founded on open source software and XML-based standard interfaces, *Proceedings of the 2002 Winter Simulation Conference*

Wiedewitsch J. and Heusmann J. 1995. "Future Directions of Modeling and Simulation in the Department of Defense", Proceedings of the SCSC'95, Ottawa, Ontario, Canada, July 34-26, 1995

Zeigler, B. P. . Object-oriented simulation with hierarchical, modular models. Academic Press, Boston. 1990

AUTHOR BIOGRAPHIES

THOMAS WIEDEMANN is a professor at the Department of Computer Science at the University of Applied Science Dresden (HTWD). He has finished a study at the Technical University Sofia and a Ph.D. study at the Humboldt-University of Berlin. His research interests include simulation methodology, tools and environments in distributed simulation and manufacturing processes, intranet solutions and database applications.
Email : wiedem@informatik.htw-dresden.de

KARSTEN WENDT, born 1980 in Dresden; 2007 diploma in Information System Technology at the TU Dresden; 2007, is -now a scientist and PhD student at the Chair of Highly-Parallel VLSI-Systems and Neuromorphic Circuits, Faculty of Electrical Engineering, TU Dresden, implementing operating systems and multi-criteria cluster algorithms for large-scale neuromorphic hardware; 2008-now senior developer at the Dualis GmbH IT Solution Dresden, responsible for Speedsim.Net
Email : [b] KWendt@dualis-it.de