Interactive Visual Steering – Rapid Visual Prototyping of a Common Rail Injection System

Krešimir Matković, *Member, IEEE Computer Society*, Denis Gračanin, *Senior Member, IEEE*, Mario Jelović, and Helwig Hauser, *Member, IEEE*

Abstract— Interactive steering with visualization has been a common goal of the visualization research community for twenty years, but it is rarely ever realized in practice. In this paper we describe a successful realization of a tightly coupled steering loop, integrating new simulation technology and interactive visual analysis in a prototyping environment for automotive industry system design. Due to increasing pressure on car manufacturers to meet new emission regulations, to improve efficiency, and to reduce noise, both simulation and visualization are pushed to their limits. Automotive system components, such as the powertrain system or the injection system, have an increasing number of parameters, and new design approaches are required. It is no longer possible to optimize such a system solely based on experience or forward optimization. By coupling interactive visualization with the simulation back-end (computational steering), it is now possible to quickly prototype a new system, starting from a non-optimized initial prototype and the corresponding simulation model. The prototyping continues through the refinement of the simulation model, of the simulation parameters and through trial-and-error attempts to an optimized solution. The ability to early see the first results from a multidimensional simulation space — thousands of simulations are run for a multidimensional variety of input parameters — and to quickly go back into the simulation and request more runs in particular parameter regions of interest significantly improves the prototyping process and provides a deeper understanding of the system behavior. The excellent results which we achieved for the common rail injection system strongly suggest that our approach has a great potential of being generalized to other, similar scenarios.

Index Terms—Interactive computational steering, interactive visual analysis, simulation, common rail injection system.

1 Introduction and Related Work

Increasing complexity and a large number of control parameters make the design and understanding of complex systems (such as automotive engines) impossible without simulations. Strict emission rules and regulations force car manufacturers to design improved engines, in very short time [2]. To meet those requirements, car manufacturers use simulations as a cost-efficient, and often the only possible way to design systems with desired characteristics. They use many types of simulation, including Computational Fluid Dynamics (CFD).

In this paper, we describe results from a recent project where the need for interactive steering emerged. We used interactive visual analysis to support an interactive design process. In contrast to the usual, very time consuming 3D CFD simulation, 1D CFD that is alternatively used in injection system simulation can be computed very fast. It is possible to run tens of thousands of simulations for a large set of parameters. However, the brute force approach, where a simulation runs for all possible parameter combinations, is often not feasible. Instead, interactive simulation steering helped us to insure a reasonably short design time. A pure numerical optimization is sometimes too complex and a user often gets only the final results, without proper insight.

The background of this work was the task to design an injection system. We developed a steering framework to support this task. Our interdisciplinary project setup provided us with valuable feedback during the design process in terms of the usefulness of the proposed approach and suggested improvements. We started from a simple model and gradually made it more and more complex.

One of the important parts of the automotive engine system is the injection system. The piezoelectric stack actuator is the main component of the injection system model [5]. When an input voltage is applied, the electric field across the ceramic layers of the stack actuator induces a mechanical strain. The strain results in an elongation of the stack that exhibits the rate-independent hysteresis between the electric voltage (force) and mechanical strain (displacement).

We identified tasks that can be generalized to other problems and illustrated how we designed and tuned the model. The model complexity did not allow us to run all possible simulations at the beginning and to analyze the results. Such an approach would also result in numerous unnecessary simulation runs and would waste time and computational resources. Furthermore, we did not have a complete model at the beginning. It was gradually built as we gained insight during the design process. Our approach, the use of interactive visualization and coordinated multiple views as a steering mechanism for simulation, proved to be very efficient. In this paper we show how the tight integration of visualization and simulation can significantly improve an engineer's workflow as compared to decoupled systems. The excellent results which we achieved for the common rail injection system and the very positive feedback from domain experts strongly suggest that our approach has a great potential and can be generalized to other, similar scenarios.

The decoupling of simulation and analysis can present significant obstacles and make it very difficult to effectively manage large amounts of simulation data [18]. We should be able to interactively steer computations, change simulation parameters or representation and immediately see the simulation results. Computational steering and interactive visualization emerged in 1980s and 1990s as some of the most useful visualization paradigms for computational science [3].

Many simulations are computationally intensive and may require interpolation for sensitivity analysis and optimization. Sensitivity analysis and optimization require the domain expert to interpolate the observed simulation data. This interpolating function is a metamodel of the underlying simulation model which is treated as a black box. An example is the Kriging interpolator representing a global metamodel that covers the whole experimental area [19]. However, we can also iteratively refine the simulation model. That way we can refine both the simulation parameter values and the simulation model.

The simulation output data is often visualized using scientific visualization methods [6]. Using visualization and spatial tools to under-

- *Krešimir Matković is with VRVis Research Center, Vienna, E-mail: Matkovic@VRVis.at.*
- *Denis Gračanin is with Virginia Tech, E-mail: gracanin@vt.edu.*
- *Mario Jelović is with AVL AST, Zagreb , E-mail: mario.jelovic@avl.com.*
- *Helwig Hauser is with University of Bergen, E-mail: Helwig.Hauser@uib.no.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mailto:tvcg@computer.org.

stand complex systems is not a new idea. James C. Maxwell, one of the most important physicists of the nineteenth century [22], often used visual-spatial thinking. An excellent example of his approach is a construction of 3D clay model of a surface based on Willard Gibbs' work. We've come a long way since those early beginnings. However, thinking about the science is still at the core of scientific visualization [8]. The most important scientific visualization research problems include perceptual issues, human-computer interactions, global/local visualization, feature detection and visual abstraction, to name a few [8].

Computational steering integrates modeling, computation, data analysis, visualization, and data input components of a simulation [18]. However, integrating a simulation within a computational steering can be a very difficult problem. We need to address four facets of the problem [9]: control structures, data distribution, data presentation, and user interfaces. Since computational steering is a highly interactive process, the user interface is a critical component of a computational steering environment [16]. Kreylos et al. [12] describe a system for real-time interactive visualization of computational fluid dynamics (CFD) simulations that allows a user to place and manipulate visualization primitives during an ongoing simulation process. Vetter and Reed [23] described performance monitoring, control, and interactive steering of computational grids. Wenisch et al. [24] demonstrated computational steering of CFD simulations on distributed computers.

There is extensive literature about user interface and visualization of simulation data from an engineering perspective. Laramee et al. [13] use different flow visualization methods to show various aspects of the simulation data to support insight and visual analysis of the coolant flow through the cooling jacket of a car engine. Konyha et al. [11] use 3D icons to analyze simulation data of chain and belt drives. Matković et al. [15] describe a method for the analysis of a fuel injection system that provides a highly abstract view of the injection system.

Using multiple, interactively linked views of the same data set allows the user to productively combine the information gathered from the different views [7]. Doleisch et al. [4] use multiple linked views for analysis of CFD simulation data. More advanced multiple view visualization systems can be freely configured [17] and provide flexible coordination of views [20]. As the number of linked views and the amount of coordination increases, it may be necessary to visualize the visualization's structure and operation [21].

2 Computational/Interactive/Simulation Steering

We used our previously developed coordinated multiple views visualization tool ComVis [15] and extended its functionality to interface it with the simulation tool HYDSIM which is a part of the AVL Workspace [1]. In this way a steering framework has been established and used in the project.

The initial design goal for the visualization tool ComVis was rapid prototyping of new visualization techniques within the scientific context. As a consequence the tool was designed to be flexible in a way that it is easy to add new views and support new data types. The tool is intuitive to use and supports advanced interactions (multiple, iterative brushing). As a result, the tool is easy to use for domain experts from different domains (medical, engineering, etc.), and can use/read generally used data formats to provide access to existing data.

The simulation tool, HYDSIM, is a modular program for the dynamic analysis of hydraulic and hydro-mechanical systems. It is based on the theory of fluid dynamics (1D) and vibration of multi-body systems (2D). The user defines a model using 2D graph-like structures with icons and connecting elements. The defined model provides a general representation of the system topology. For each element (represented by an icon) the user can specify properties for the particular case. Once the user completes a definition of the model, the simulation provides output parameters values. In a typical workflow, a domain expert analyzes these results and, if necessary, modifies the simulation model and repeats the simulation until the desired results are achieved. Earlier we pursued an alternative approach to compute a very large set of simulations runs at once (offline) and analyze the results afterwards [10, 15]. Although this was a significant improvement

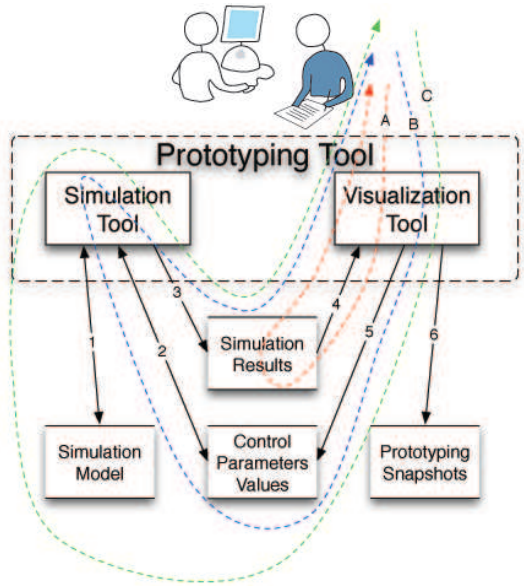

Fig. 1. An iterative approach to prototyping. A combination of the simulation and visualization tools and related data allows us to design at different levels of abstraction. We distinguish three levels of the interactive steering process depicted with loops A, B, and C. The first loop, loop A, is based on the available simulation results. We explore them, get insight and store results (snapshots). If this is not sufficient, new simulation results can be generated (loop B). The simulation model is still not changed, only parameters are being refined. Finally, it is also possible to change the simulation model (loop C).

compared to the traditional way, we still had to specify all combinations of input parameters in advance.

Our new framework makes it possible to define new simulations using the visualization tool. The visualization tool is used for the analysis and steering of the simulation. That makes it easy for the domain expert to generate new simulations and to refine or to filter the simulation dataset. Each simulation has a set of control (input) parameters and a set of output parameters that are computed for a given input. The main idea is to run many simulations with different control parameter settings (they are defined by lower limit, upper limit, and step size), and to use multiple, coordinated views to understand the model and to support the expert in injection system design.

We provide four basic operations: refining or coarsening some control parameters (changing the step); narrowing down the control parameter interval (changing boundaries); adding new control parameters; and removing some existing control parameters. If we represent data in tabular form, the basic operations correspond to adding and removing rows (refinement and filtering parameters) or adding or removing columns (adding or removing parameters).

The domain expert estimates the coarse boundaries of the parameters, runs a sufficient number of simulations and sees what parameter values make sense and what values are not allowed based on the output values. In the case of fuel injection systems, the injected fuel mass was one of the output parameters often used to identify parameter values that are not allowed. If there is not enough injected fuel or if there is too much injected fuel, the engine will not run properly.

We use an iterative approach (Figure 1). The domain expert uses the simulation tool to create the initial simulation model, specify the initial control parameter values and produce simulation results (Figure 1, steps 1, 2, and 3). Only a part of the injection system is modeled in detail while the rest is replaced with modelled "ideal" values. These ideal values became the target when we refined the model. The goal is to create a simulation model and to determine the control parameter values that produce the simulation results that are as close as possible to the idealized result. We repeat the process at three different levels.

The user first designs a very simple simulation model and sets the parameters. The user then extends the simulation model and provides the parameters for a more complex model that represents the second level. Finally, the user defines the complete simulation model that makes it possible to go back and forth between different levels (as it is often the case during prototyping) and to change already tuned parameters.

The domain expert carries out the interactive steering process at three different levels. The first level of iterative prototyping focuses on the already generated simulation results. The expert uses the visualization tool to investigate the simulation results and, by extensive use of brushing and linking, can get insight and create first reports (snapshots) about the current prototype results (Figure 1, steps 4 and 6).

If the current simulation results are not sufficient, the expert can proceed to the second level. The second level of iterative prototyping involves refining the control parameter values (Figure 1, step 5), generating new simulation results using the current simulation model and then returning to the first level of prototyping. The expert can do it in an interactive way and request new simulation results from the simulation tool. As new simulation results are computed, the data in the visualization tool is automatically updated. During this process visual analysis can proceed and benefit from better data resolution.

Based on the insight from the data, the expert may decide to refine the simulation model. At the third level of iterative prototyping the expert uses the simulation tool to update the simulation model and then returns to the second level of prototyping. As this is a larger step which makes it necessary to change internal data representation, the expert has to wait until initial setup is completed and the first results for new model are ready. This can take a few minutes. Once the first set of simulation results is computed and the visualization tool updates the internal structure, the whole process becomes interactive again. The simulation results are uploaded on the fly as they are computed.

In our implementation we always define the model using the HYDSIM tool. The HYDSIM creates simulation definition files and runs the simulations. As simulations are computed, output files are created, one directory for each simulation run. Our visualization tool, ComVis, reads the first simulation results, builds the internal data model, and visual analysis starts. The visualization tool checks for new output files and loads them when they are available. ComVis offers a possibility to specify new simulation parameters, as well. If the user requests new simulations from the visualization, ComVis creates HYDSIM input files and starts HYDSIM. HYDSIM generates new output files which are then automatically loaded into the visualization tool. Model changes are done in HYDSIM, and in these cases ComVis has to recalculate internal data structures which usually takes a while (minutes). Once the new model is created, first simulations are computed, the internal data structures needed for visualization are created, and the process continues in a usual way.

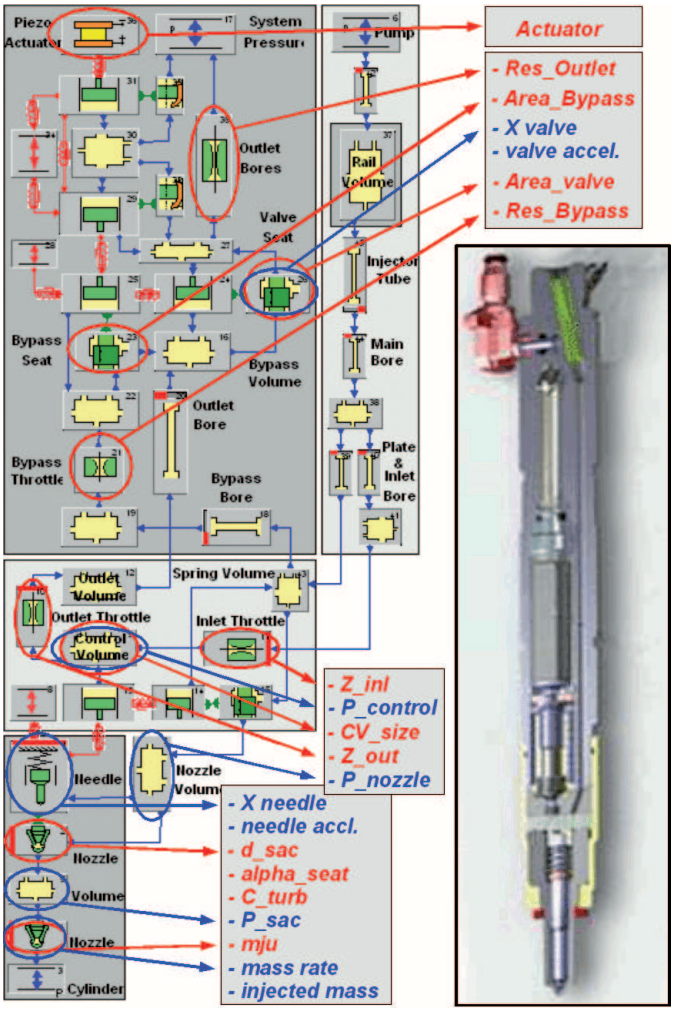

Fig. 2. The final injection simulation model and the four main blocks. The blocks represent logical grouping that has no influence on the model topology. An actual injector, used for each cylinder in a car engine, is shown on the right. The control parameters are depicted in red, and the output parameters in blue. The exact description of the parameters is too long for this caption and can be found in the main text, instead.

3 Rapid Visual Prototyping and the Design of a Common Rail Injection System

We used the developed prototyping tool to design a common rail injection system. We selected this task for two reasons, i.e., the availability of fast simulation algorithms and the importance of injection in an overall Diesel engine efficiency and emission characteristic.

There are many (often conflicting) goals of a Diesel engine design, including high power, good fuel efficiency, meeting emission regulations, low noise levels, and drivability [14]. The fuel injection system is the key Diesel engine component to achieve the goals. The common rail injection system has several attractive characteristics: injection pressure and quantity can be controlled with a high degree of flexibility, multiple fuel injections are possible within one injection cycle and the time and duration of the injections can be controlled precisely by the engine control unit based on the engine speed and load. These characteristics are key factors in meeting current and future (very demanding) emission regulations.

The common rail injection system consists of two parts, one hydro-mechanical and one electronic. The hydro-mechanical part determines the simulation model, while the electronic part determines the actuator control parameters. Our goal is to design both parts by iteratively adjusting the simulation model and the control parameter values.

We start by providing the end result of the design process (Figure 2). The reason for that is to provide the context and the basic expertise for the system design, something that experts already have. The main assembly components of the injector are the piezo actuator with the hydraulic amplifier (*Block III* from figure 2), the control valve with a certain control volume (*Block II*) and the nozzle (*Block I*). The piezo actuator governs the motion of the control valve. *Block IV* is the fuel supply from the common rail, which is not analyzed in this work.

Once the simulation model is created, the expert has to set up the control parameters (the parameters listed in red in Figure 2). For each set of the control parameters the output parameters are computed (the parameters listed in blue in Figure 2). All of the control parameters in our case are scalar values, and all of the output parameters are time series data. An additional control parameter is the actuator (the topmost element in the Figure 2) behavior. We model the actuator curve depicted in Figure 3 using a set of scalars, determining the start and duration of the pilot and main injection, their maximum amplitudes, and opening and closing times.

The model has 11 control parameters and setting them is the main task of the injection system design. The actuator curve parameters are

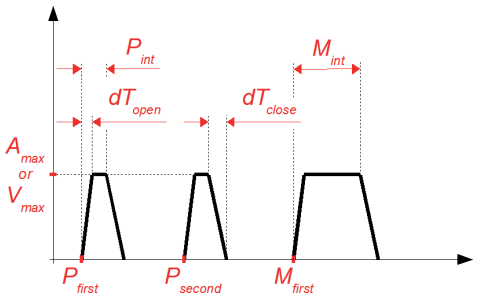

Fig. 3. The actuator, the top most element in the model in Figure 2, is modeled using these parameters. This is the only set of control parameters that will be changed during the engine operation. Depending on the operation point (speed and load), the electric managing unit (EMU) will select the shape of actuator curves.

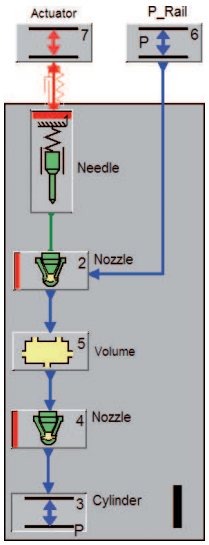

Fig. 4. A simplified model used in the first iteration. We use only one section and the rest is represented by an ideal actuator. Once the simplified model is tuned, we gradually extend it to include the rest of the model, as shown in Figure 2.

also set, but they vary, based on the crankshaft load and speed, during the actual engine operation. The electronic control unit (ECU) of a car engine controls these curves. ECU has a lookup map of all possible curves and selects a curve based on the current crankshaft load and speed. If a car runs downhill at a certain speed, the crankshaft load can even be negative, so the actuator curves are chosen accordingly. On the other hand, for a high crankshaft load and a certain speed, the actuator curves and resulting injection have completely different shapes. However, a detailed discussion of injection curve modeling is out of the scope of this paper.

One design option [10, 15] is to run the simulation for all possible combinations of parameters and to explore the system. If we use ten values per control parameter, there are 10^{11} possible combinations of control parameter values. Since we can run about ten simulations per minute (for the model in Figure 2), we would need 10^{10} minutes, or more then 19,000 years to complete all simulation runs. It is clear that such simulation time, even on a large cluster, is not feasible. Instead, we start with a simplified model, use interactive visualization to drill down the control parameter space, and once the initial control parameters are fixed, the simulation model is extended.

3.1 First Model

We start with a simplified model (Figure 4), *Block I* from Figure 2 with the actuator directly added on the top. This is supposed to be a direct actuator with simple characteristics. We tune the nozzle first.

Table 1. Control Parameters for the first case.

Parameter	Min	Max	Step
d_sac	0.7	0.9	0.05
alpha_seat	40	65	5
c_turb	0.8	1.0	0.05
mju	0.7	0.9	0.05

Table 2. Target Parameters for the first case.

Parameter	Target range (mg)
Pilot injected mass	2 – 2.5
Main injected mass	17 – 22

We use only four control parameters (Table 1). It takes about 12 minutes to calculate 750 cases (60 simulations per minute for this simpler model). After this setup time, we explore the first data set to achieve a certain amount of injected fuel during the pilot and main injection. The target values were set according to Table 2.

We compute the calculated output parameter *injected mass* as a function of time. It is a cumulative mass over time. The value at the end corresponds to the totaly injected mass during injection. As we are interested in the injected mass after the first pilot injection and after the main injection, we aggregated the injected mass curves so to have the injected mass after first pilot and the total injected mass. We brush now the scatter plot depicting these two aggregated parameters. Figure 5 a shows this case.

At the same time, other coordinated views show the control parameters (Figure 5a, scatter plots in the first column) causing the targeted injected mass. We can clearly see that the wanted injected mass is possible only for some combinations of nozzle diameter (*d_sac*) and angle of needle seat (*alpha_seat*), and for all combinations of flow discharge coefficient (*mju*) and turbulent flow coefficient (*c_turb*). We refine the selection by selecting high pressure (as far as possible for the given target) and the desired needle acceleration. Figure 5b illustrates the selections (brushes 2 and 3). Note the zoomed-in scatter plot of accelerations which helped in the selection.

The allowed control parameter space has narrowed significantly (Figure 5b, scatter plots in the first column), and the expert can now select the first parameters (Table 3). Note that this is a quite coarse estimation. The parameters are fine-tuned at a later stage. However, even this coarse case shows which input ranges make no sense.

3.2 Second Model

We refine the simplified model (Figure 4) to create a more detailed model containing *Block II*. The actuator with the control valve is placed on top of *Block II* now. This actuator is described in Figure 3. We tune this part using the fixed control parameters for *Block I*. We are interested in the control volume size (*CV_size*) and in the inlet/outlet throttle flow resistance (*Z_inl* and *Z_out*). 1,100 simulations are computed (20 simulations per minute since the simulation time changes with the model complexity). For the same target values as in the first case, we set the control volume size to ten. Figure 6 shows the parallel

Table 3. Control Parameters selected for First Model.

Parameter	Target range (mg)
d_sac	0.75
alpha_seat	50
c_turb	0.9
mju	0.7

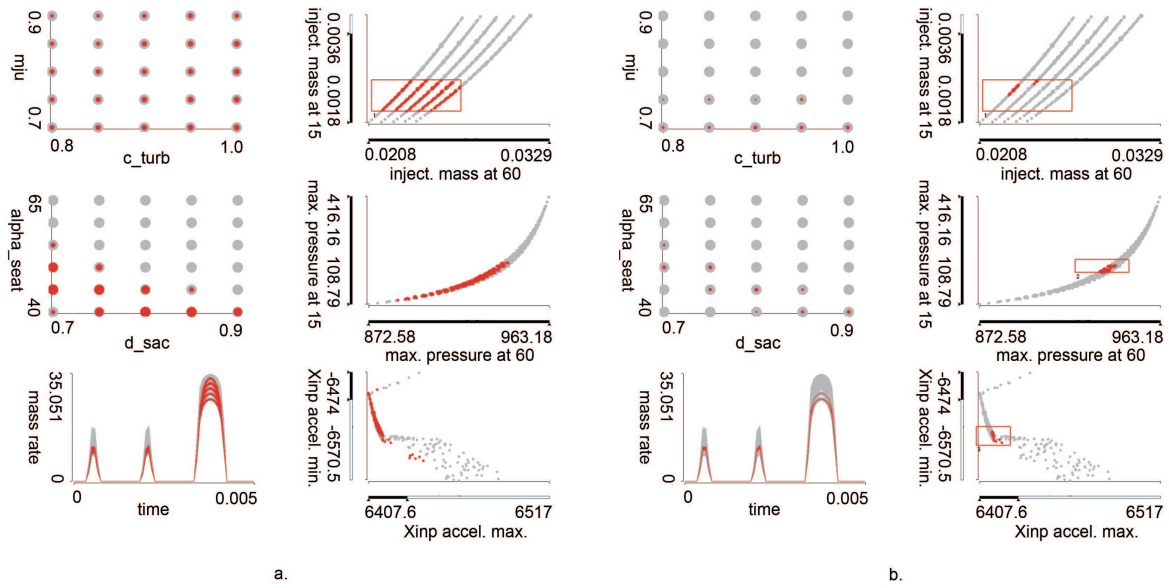

Fig. 5. Multiple coordinated views show control and output parameters. **a)** We brush target values for the injected fuel mass. All combinations of *c_turb* and *mju* can produce the desired output (second scetterplot) while only some combinations of *d_sac* and *alpha_seat* are possible. **b)** Further refinement of targets using additional brushes for pressure and acceleration helps us to narrow possible parameters and to estimate input parameters for the first step.

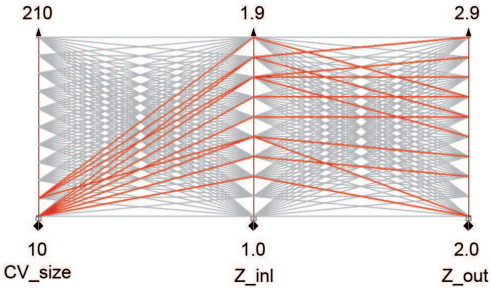

Fig. 6. Second Model: The parallel coordinates view of the control parameters for the target values. Note that only low *CV_size* values are possible, while there are many combinations of *Z_inl* and *Z_out* which allow the desired output.

coordinates view of the control parameters for the target values.

If we look at the mass rate curve view now, a new, interesting phenomenon can be observed (Figure 7). Note the width of curves at injections. There are some curves starting later and some starting on time. The actuator on the top is fixed, i.e., it has always the same input curve (Figure 3). This means that some of the parameter combinations can cause an injection delay. Injection delay is the time period between the injection reaction and the actuator action. In our case this is the difference between the start of the pilot (or main) injection as depicted in the curve view and the actuator starting times (P_first or M_first). This is a surprising finding, since we did not expect that this delay would show up at this stage of modeling.

The injection delay is an unwanted behavior and we have to be sure that it does not happen in the final model. Compared to the first, simplified model, the control volume which is placed just above needle top is not directly connected to the actuator any more. It is working in close correlation with two orifices (inlet and outlet) that supply the volume with fuel and drain it. We have to be sure that delay is as small as possible at this stage of modeling.

The correlation between the mass flow rate through the nozzle (*mass_rate*) curves and the control volume size (*CV_size*) is easy to detect. Simple brushing shows that the larger the control volume size is, the narrower (less injected fuel) the curves are. Figure 7 depicts the curves selection for the maximum values of *CV_size*. The inlet and outlet flow resistance (*Z_inl* and *Z_out*) are more challenging and to investigate this problem we refine the model. New limits and step sizes are selected using the visualization tool and new simulations are initiated. The control volume size is set to the minimum, $CV\_size = 10$, and new flow resistance parameters *Z_inl* and *Z_out* are set (Table 4). The simulation tool is started and the visualization gets updated as new simulations are computed. The visualization tool checks if new data is available and automatically loads it. Since we do not change the model in this case, the update of the internal data and its representation is straightforward. During this process we continue the visual analysis and exploration. In this particular case, approximately 1,680 new simulations are computed, and iteratively loaded. Of course, the user can stop the simulation or request another refinement at any time.

We used the multiple view setup to observe what is happening. The target values for the injected mass, pressure in the SAC volume *P_sac*, and the needle acceleration are set, and flow resistance parameters show a linear dependency. Figure 8 shows the flow resistance parameters on the top and the mass rate curves in the bottom. Note the much denser parameter space due to the refinement. After a detailed exploration, we are able to remove the influence of the parameters on the delay. We understand what is going on, the delay turns out to be logical, and the parameters are set to 1.6 and 2.6 for *Z_inl* and *Z_out*.

3.3 Third Model

We are ready for the final step now where we further extend the model. It now corresponds to the model in Figure 2. Note that the control parameters for *Block I* and *Block II* are set (but they can be changed) and we tune the last part now. There are four parameters in the last block:

Table 4. Refined Control Parameters for Second Model in order to investigate injection delay.

Parameter	Range	Step
Z_inl	1,0 - 2.0	0.025
Z_out	2.0 - 3.0	0.025

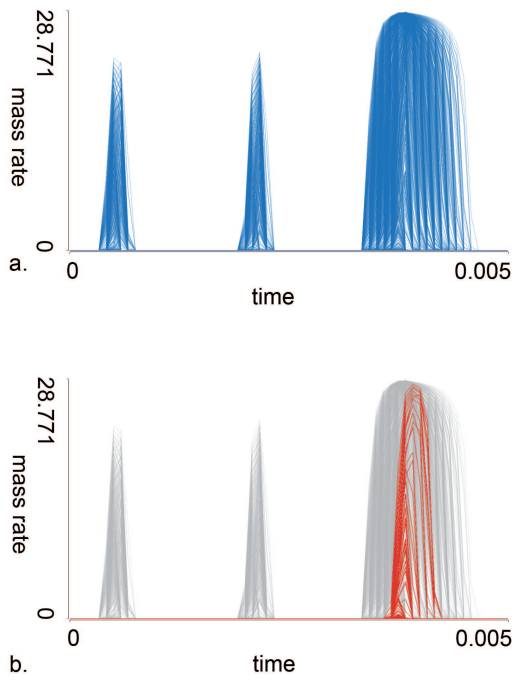

Fig. 7. *Mass_rate* curves for the second model. **a)** Many curves of different widths, showing significant delays for some parameter combinations. **b)** The selection corresponds to the maximum values of *CV_size*, such curves would result in insufficient fuel mass. Note also that pilot injections are completely missing in this selection.

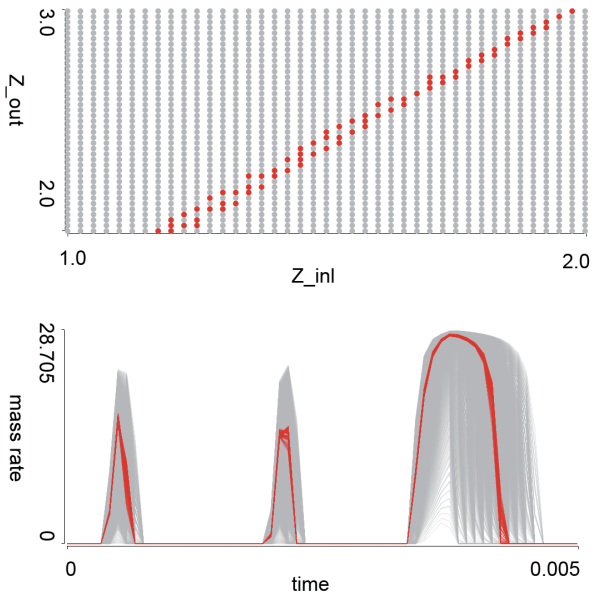

Fig. 8. The scatterplot shows refined input parameters. We have selected desired output parameters (on views which are not displayed in the image) and the selection is shown in red. The *mass_rate* curves have a desired shape now. The linear correlation of *Z_inl* and *Z_out* is unexpected.

the bypass flow resistance (*Res_Bypass*), the outlet flow resistance (*Res_Outlet*), the effective flow area at the bypass seat (*Area_Bypass*), and the effective flow area at the valve seat (*Area_Valve*).

Due to the model complexity we now calculate approximately ten simulations per minute. 900 parameter variations are set and we start

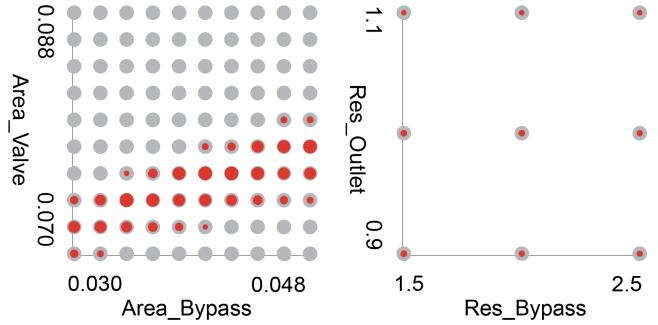

Fig. 9. Two scatterplots showing the control parameters for the final model. We further refine the *Area_Bypass* and *Area_Valve* parameters. The red points correspond to selected desired output parameters.

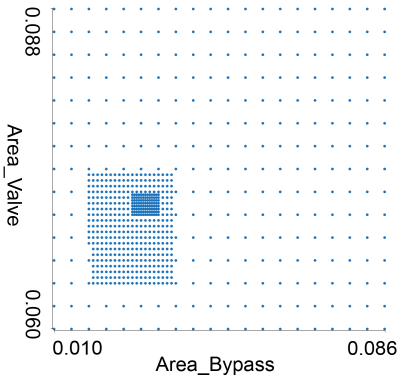

Fig. 10. Combinations of the control parameters as created during the iterations. We have a coarse mesh of parameter values at the beginning, and we refine them twice during the process. All combinations of the control parameters are shown here, we can hide unwanted iterations during the analysis, if necessary.

the visual exploration. In contrast to the simple parameter refinement, the step change here represents a model change. Internal data structures in the visualization tool have to be changed. This is considered to be a larger step and the user has to wait a few minutes for the first results. Once the simulation software computes the initial results, parameter refinement can be done on the fly. During parameter refinement we continue the visual analysis and the data is automatically updated. The target values for the injected fuel mass (pilot and main) remain the same. The actuator is still fixed.

Figure 9 shows the parameters after the target values were selected. Two parameters have no significant influence, the target values can be achieved with all possible combinations of the flow resistances *Res_Bypass* and *Res_Outlet*. We set the values to 2.0 and 1.0, respectively. The other two parameters show a far more interesting behavior. A wide range of parameter values are initially investigated. It is successively refined as we realized where we need more information.

Figure 10 shows the parameters as computed at the end. We use two iterations, we refine the parameters once and then refine a subrange of parameters once more. This represents parameter refinement, and the data in the visualization tool is updated as new simulations are computed. During computation we continue the visual analysis. Output values are used to steer the refinement. Based on the output values we decide where to refine input parameters.

Figure 11 shows an example of the output parameter values as they are computed in various steps. Resulting outputs from various iterations are highlighted in the figure in order to illustrate results from various iterations.

Figure 12 shows the target injected mass, the corresponding pressure, control parameters, and mass rate curves. Note the scatterplot on

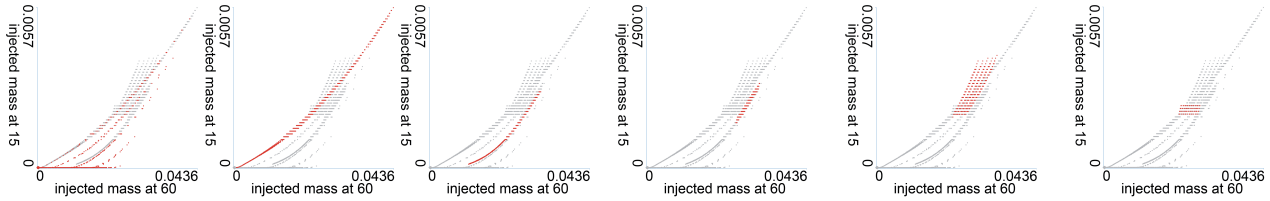

Fig. 11. Six scatterplots showing output parameters as computed during six iterations of simulation steering. We can see there are many scattered values in the beginning. We then used interactive brushing in other views to get insight on how these output parameters are changing. Finally we can identify the desirable area, and the simulation results are refined in that area.

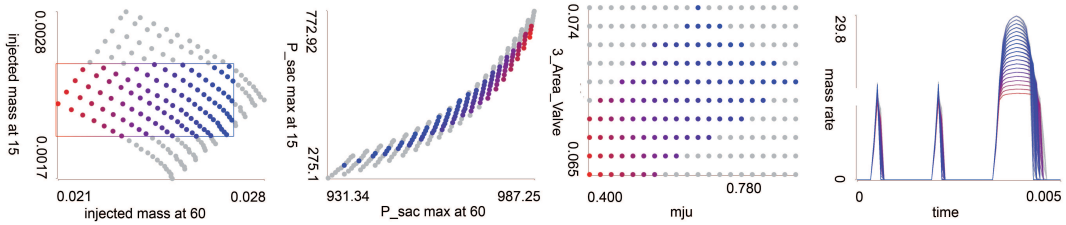

Fig. 12. The final model. The target injected mass on the left is defined using a quite narrow range here. The corresponding pressure, input parameters and mass rate curves are shown. Everything seemed to be correct in this simulation model.

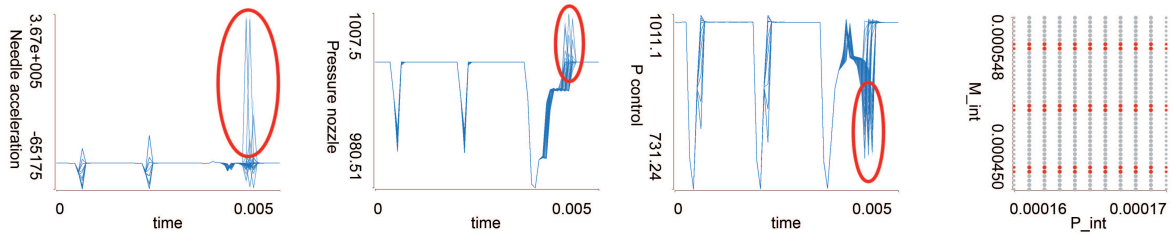

Fig. 13. With the final model fixed, the actuator curves are varied. Quite surprisingly, there are unwanted peaks in the output curves for multiple parameters. Red ellipses show these peaks. It is not intuitively clear why and when they occur, and we explored it in more details. The assumption that all parameters are set and that actuator variations are just a routine fails and we had to go several steps back and run new simulations. Corresponding values of *M_int* are shown in the scatter plot on far right. Note the puzzling oscillating behavior.

the far left showing the same data (also in Figure 11) as the scatterplot from the very first model (Figure 2). The scatterplot in Figure 12 is zoomed in and shows the data from the final iteration only. We are far off the target values at the beginning and by successive refinement and simulation steering we achieve a finer granularity around the target area.

3.4 Final Model

We now set the final control parameters. Any of the data points of the injected mass shown in Figure 12 can be selected. They all result in a desired behavior. We have to select one set, however, as they can not be changed later. Due to the wanted pressure and needle acceleration, we select the effective flow area at the valve seat — *Area_Valve* and the flow discharge coefficient — *mju* to be 0.071 and 0.54, respectively. Our injector now is set.

The actuator on the top used to drive the injection in a real setup was fixed up to now. As stated before, the ECU of the engine will change the actuator during operation. Our parameters, on the other hand, remain the same. They cannot be changed at runtime.

The last task is to check if the parameters also yield satisfying output for various actuator curves. We vary the actuator settings and again several times refine the parameters. Eventually we are satisfied and want to see all the response curves for all the combinations of actuator parameters (1,600 combinations are chosen).

To our great surprise, we see that some curves exhibit a very unusual behavior. The curve views in figure 13 show the response curves for *Needle_acceleration*, and pressures in the nozzle and control volumes, *P_nozle* and *P_control*, with the undesired peaks marked with red ellipses. Those peaks indicate system oscillations at specific points. Any oscillation in system is dangerous and undesired. The amplitude of any oscillation may rise above system limits for some unknown situations.

Now we have to find the reason for these oscillations in order to predict and avoid such a behavior. Furthermore, especially for the fuel injection system, any kind of secondary oscillations may open the nozzle at the wrong time and lead to fuel inflow in the combustion chamber and an undesirable combustion process.

To investigate the oscillations further, we isolate the peaks using a line brush in the curve view. The tool allows to simply draw a line across the curves, and all curves crossing the line will be selected. The composite brushing functionality is supported as well.

The scatter plot in Figure 13 shows pilot and main injection intervals, *P_Int* and *M_Int*, with the peaks selected. An unexpected and very interesting finding is the pattern at which peaks appear at 3 *M_Int* values. It shows oscillating behavior in the parameter space. We can easily skip those values, and program the ECU not to use these parameter values.

However, puzzled by this discovery, we want to investigate this phenomenon further. We go back one more time. The parameters with most influence up to now: *mju*, *Area_valve*, *Area_Bypass* and *M_int* are varied once more. Undesirable peaks are present again, but the control parameters are chosen to be far away from the settings which caused them. The previously set parameters are changed, and the in-

Table 5. Final control parameters. The *Block IV* parameter values are not analyzed at this stage (listed for the sake of completeness).

Param.	Name	Description	Final Value
I_1	d_sac	Sac diameter	$0.75mm$
I_2	$alpha\_seat$	Needle seat angle	$50degrees$
I_3	c_turb	Turbulent coefficient	0.9
I_4	mju	Flow discharge coefficient at nozzle holes	0.6
II_1	CV_size	Size of control volume	$10mm^3$
II_2	Z_inl	Inlet flow resistance in control volume	1.6
II_3	Z_out	Outlet flow resistance out of control vol.	2.6
III_1	Res_Bypass	Flow resistance through bypass	2.0
III_2	Res_Outlet	Flow resistance through outlet	1.0
III_3	$Area\_Bypass$	Bypass effective area	$0.032mm^2$
III_4	$Area\_Valve$	Valve effective area	$0.07mm^2$
IV_1	HPP_Length	Length of high pressure pipe (fixed)	$300mm$
IV_2	RV_Size	Common Rail volume size (fixed)	$30cm^3$

jector is finally set. Table 5 shows the final values of the control parameters.

4 Conclusion

The coupling of interactive visualization with the simulation back-end facilitates fast prototyping of a system under development. We start from a non-optimized initial prototype and the corresponding simulation model and through an iterative process, going back and forth between different levels of abstraction, we refine the simulation model and design a system that meets the requirements. In doing so we are significantly reducing the number of simulation runs.

The brute force approach requires to run simulations for all possible combinations of the control parameter values which is not computationally feasible. The described approach requires only several thousands simulation runs to find a design that meets the requirements.

The interdisciplinary setup of this project allowed to develop this steering solution in the context of a real-world problem. It was very rewarding to see how the tool facilitated new discoveries (Section 3.4), even quite surprising ones. The discoveries provided much better insight and allowed us to anticipate and address oscillation problems and thus create a much better design. Engineers still only seldom use interactive visualization and usually analyze simulation results using static 2D charts, depicting few simulation runs simultaneously in most cases. They also use automatic optimization methods, but our approach offers completely new view and insights.

The three levels of iteration (simulation data, control parameters values, simulation model) provide different levels of interactivity. While viewing the simulation data is done in real-time, changing the simulation model introduces a noticeable delay. However, since we can go back and forth between different levels, instead for waiting for the simulation model update to propagate to the simulation data, we use the simulation data level and continue our analysis until new simulation data are generated. This approach is rather general and applicable to a wide range of design problems.

We plan to explore a variety of design problems and related solutions to identify some design patterns. We will further improve the interactivity of the developed tool. Some semi-automatic support for drill-down, possibly involving approaches to (semi-) automatically detect a region which seems to be out of the range of interest will be researched as well. Finally, we will explore a collaborative, multi-user version of the tool to "share" the design process among several experts.

Acknowledgements

Parts of this work have been done through the scope of applied and basic research at the VRVis Research Center (funded by an Austrian governmental research program called Kplus, http://www.kplus.at/) and at the Center for HCI at Virginia Tech (http://www.hci.vt.edu/).

References

[1] AVL List GmbH, http://www.avl.com [Last accessed on 30 June 2008].

[2] F. Boecking, U. Dohle, J. HammBoeckinger, and S. Kampmann. Passenger car common rail systems for future emissions standards. *MTZ*, 66(7–8):552–557, 2005.

[3] J. X. Chen, D. Rine, and H. D. Simon. Advancing interactive visualization and computational steering. *IEEE Computational Science & Engineering*, 3(4):13–17, 1996.

[4] H. Doleisch, M. Mayer, M. Gasser, P. Priesching, and H. Hauser. Interactive feature specification for simulation data on time-varying grids. In *Proc. of the Simulation and Visualization 2005*, pages 291–304, 2005.

[5] M. Goldfarb and N. Celanovic. A lumped parameter electromechanical model for describing the nonlinear behavior of piezoelectric actuators. *ASME Journal of Dynamic Systems, Measurements, and Control*, 119:478–485, Sept. 1997.

[6] H. Hagen, A. Ebert, R. H. van Lengen, and G. Scheuermann. Scientific visualization: methods and applications. In *Proc. of the 19th Spring Conf. on Computer Graphics*, pages 23–33, 2003.

[7] C. Henze. Feature detection in linked derived spaces In *Proc. of the IEEE Visualization 1998*, pages 87–94, 1998.

[8] C. Johnson. Top scientific visualization research problems. *Computer Graphics and Applications, IEEE*, 24(4):13–17, 2004.

[9] C. Johnson, S. G. Parker, C. Hansen, G. L. Kindlmann, and Y. Livnat. Interactive simulation and visualization. *IEEE Computer*, 32(12):59–65, 1999.

[10] Z. Konyha, K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual analysis of families of function graphs. *IEEE Trans. on Visualization and Computer Graphics*, 12(6):1373–1385, Nov./Dec. 2006.

[11] Z. Konyha, K. Matković, and H. Hauser. Interactive 3D Visualization Of Rigid Body Systems. In *Proc. of the IEEE Visualization 2003*, pages 539–546, 2003.

[12] O. Kreylos, A. M. Tesdall, B. Hamann, J. K. Hunter, and K. I. Joy. Interactive Visualization and Steering of CFD Simulations In *Proc. of the Eighth Eurographics Workshop on Virtual Environments*, 2002.

[13] R. S. Laramee, C. Garth, H. Doleisch, J. Schneider, H. Hauser, and H. Hagen. Visual Analysis and Exploration of Fluid Flow in a Cooling Jacket. In *Proc. of the IEEE Visualization 2005*, pages 623–630, 2005.

[14] B. Mahr. Future and potential of diesel injection systems. In *Proceedings of the Thiesel 2002 Conference on Thermo- and Fluid-Dynamic Processes in Diesel Engines*, Sept. 2002.

[15] K. Matković, M. Jelović, J. Jurić, Z. Konyha, and D. Gračanin. Interactive visual analysis and exploration of injection systems simulations. In *Proc. of the IEEE Visualization 2005*, pages 391–398, 2005.

[16] J. D. Mulder, J. J. van Wijk, and R. van Liere. A survey of computational steering environments. *Future Generation Computer Systems*, 15(1):119–129, 1999.

[17] C. North and B. Shneiderman. Snap-together visualization: a user interface for coordinating visualizations via relational schemata. In *Proc. of the Working Conf. on Adv. Vis. Interfaces*, pages 128–135, 2000.

[18] S. G. Parker, C. J. Johnson, and D. Beazley. Computational steering: Software systems and strategies. *IEEE Computational Science & Engineering*, 4(4):50–59, 1997.

[19] W. C. M. van Beers and J. P. C. Kleijnen. Kriging interpolation in simulation: a survey. In *Proc. of the 36th Winter Simulation Conf.*, pages 113–121, 2004.

[20] C. Weaver. Building highly-coordinated visualizations in improvise. In *Proc. of the IEEE Symp. on Inf. Visualization 2004*, pages 159–166, 2004.

[21] C. Weaver. Visualizing coordination in situ. In *Proc. of the IEEE Symp. on Inf. Visualization 2005*, pages 165–172, 2005.

[22] T. G. West. Images and reversals: James Clerk Maxwell, working in wet clay. *ACM SIGGRAPH Computer Graphics*, 33(1):15–17, 1999.

[23] J. S. Vetter and D. A. Reed. Real-time Performance Monitoring, Adaptive Control and Interactive Steering of Computational Grids. *International Journal of High Performance Computing Applications*, 14:357–366, 2000.

[24] P. Wenisch and C. van Treeck and A. Borrmann and E. Rank and O. Wenisch. Computational Steering on Distributed Systems: Indoor Comfort Simulations as a Case Study of Interactive CFD on Supercomputers. *Int. Journal of Parallel, Emergent and Distributed Systems*, 22(4):275–291, 2007.