Process-oriented Approach to Verification in Engineering

Ekaterina Auer, Roger Cuypers and Wolfram Luther
INKO, University of Duisburg-Essen, Lotharstr. 63, Duisburg, Germany

Keywords: Numerical Verification Assessment, Validation, Uncertainty, Result Verification.

Abstract: Common verification and validation methodologies suggest using verification benchmarks in the cases where no theoretical proof of the model and algorithm accuracy is possible and interpreting the outcome of the process of interest on these. In this paper, we propose to shift the focus on the process itself. Helped by appropriate questionnaires, users are urged to subdivide their processes into smaller tasks and to classify the input/output data and algorithms, which allows us to assign a certain initial verification degree to the (sub-) process from a recently proposed four-tier hierarchy. After that, the initial degree can be improved as well as uncertainty quantified by with the help of a selection of specialized interoperable data types and tools. Besides, we address issues of software quality and user support by describing a comparison system for verified initial value problem solvers.

1 INTRODUCTION

The demands on the quality of products developed in engineering grow significantly every year. On the one hand, the market requires shortening of the production cycle and cost reduction, so that expensive prototypes become uncompetitive and computer simulations unavoidable. On the other hand, the need for ensuring and testing the quality of the future system is as high as ever, because there appear more and more areas where safety, reliability, and dependability analysis play a crucial role. Most prominent examples here are modeling and simulation for surgery, for nuclear power plants or for aeronautics, but the same is true for cell phone networks or for electronic commerce. The increased demand for validity or at least credibility of models and simulations lead to the appearance of a common research direction inside many application areas which addressed working out and formally defining guidelines for verification and validation (V&V) of corresponding systems.

This paper aims at systematizing information on V&V assessment and uncertainty treatment in engineering with the focus on computer-based modeling and simulation in biomechanics. Simultaneously, its goal is to enhance the usual methodology with the help of verified methods (Moore et al., 2009) where it makes sense. Such methods provide a computational proof that the outcome of a simulation is correct. Especially in the field of verification, these techniques can be employed for entire processes or their parts without the need to simplify them to known benchmarks or special cases. Moreover, the areas of model design and validation might profit from the ability of verified methods to handle bounded uncertainty and from verified sensitivity analysis. In the paper, we provide general verification guidelines from our point of view. An illustration for this is given in (Auer et al., 2012), where the V&V process for the hip prosthesis fitting is described.

There is a long tradition in the area of computational fluid dynamics of developing and testing methodologies and tools for verification and validation assessment. This process starts with precise definition of the involved concepts, in particular, verification and validation themselves. The American Institute of Aeronautics and Astronautics and the American Society of Mechanical Engineers adopted the following terminology (AIAA, 1998; ASME, 2006). *Verification* is the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model, whereas *validation* is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model. These definitions implicitly assume a certain notion of what a system is. Here and in the following, we postulate that it is possible to define and formalize the system to be engineered in a structured and hierarchical form. Particularly, the system should be describable in such a way as to allow for its later implementation on a computer.

Auer E., Cuypers R. and Luther W..
Process-oriented Approach to Verification in Engineering.
DOI: 10.5220/0004037905130518
In *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2012), pages 513-518
ISBN: 978-989-8565-22-8

Still more particularly, we assume that the system (or at least its parts) possesses a mathematical model. In the following, we use the terms *system*, *product*, and *problem to be engineered* interchangeably.

The known assessment methodologies usually concentrate on mathematical modeling exclusively. They rely on analytically solvable special cases and simplifications as well as benchmark examples, so that they do not provide a definitive step-by-step V&V procedure immediately applicable by an engineer. A major difference between our V&V understanding and the traditional one is the approach to verification advocated in this paper. Whereas common V&V methodologies suggest using verification benchmarks in the cases where no theoretical proof of the model and algorithm accuracy is possible and interpreting the outcome of the process of interest on these, we propose to consider the process itself. Helped by appropriate questionnaires, users are urged to subdivide their processes into smaller tasks and to classify the input/output data and algorithms, which allows us to assign a certain initial verification degree from the four-tier hierarchy developed in (Auer and Luther, 2009) to the (sub-)processes. After that, the initial degree can be improved as well as uncertainty quantified by using a selection of specialized data types (e.g., enhanced interval arithmetic) and corresponding algorithms. This step is not a part of traditional V&V guidelines.

The issue of interoperability with respect to data types, algorithms and hardware is of a great relevance. For instance, users should have the possibility to combine interval-based and statistical software (e.g., for the Monte Carlo method). On the one hand, they would be able to solve problems which cannot be treated with interval methods alone (e.g. if bounded system parameters have outliers of significant magnitude). One the other hand, they could compute the solution in several ways in parallel as well as compare and interpret the outcome of these different computational models.

The correlation between modeling/simulation and verification/validation activities is usually represented as a cycle (Schlesinger, 1979). The first stage of the cycle is to analyze the real world problem and to develop a conceptual and a formal model for it. The task of V&V assessment at this stage is qualification, which includes determining the types of uncertainty and methods for its treatment, relevant parameters and their ranges, and quality of input/output data. Besides, it might be profitable to carry out dependability analysis at the early design stages. For example, if we want to model human lower extremities, an inverted double pendulum would correspond to the conceptual model, whereas its equations of motion would constitute the formal (mathematical) model.

The next step is to implement the formal model. Here, verification is performed, the task of which is to make sure that the formalized model is implemented correctly in the sense established by requirements devised during qualification. This is the stage where we need to know the relationships between system elements (a structural system definition) and not solely their inputs and outputs (a functional system definition). In general, the activities fall into the categories of code verification (software quality analysis) and (numerical) result verification. The latter can be carried out in different ways. The usual methodology is to assess the accuracy by using formal mathematical proofs and analytical solutions. If that is not possible, simplifications, benchmark cases, and independent computations are employed. If we return to our biomechanical example once again, then the task here corresponds to either implementing the double pendulum using modeling and simulation software based on a certain theoretical principle (e.g. the multibody method) to obtain of its equations of motion or deriving them analytically. We suggest using methods with automatic result verification such as intervals or Taylor models (Berz, 1995) at this stage.

The third and probably most laborious step is validation. The task of V&V assessment here is to compare the outcomes of computerized model simulations or of a system prototype to reality, to establish the degree of similarity using a (non-deterministic) validation metric, and to address model fidelity. Although is it not necessary to know the relationships between system's elements to perform simple validation, this knowledge is helpful if uncertainty were to be included into the validation metric. In terms of a biomechanical example given above, it is necessary to compare the results of a computer simulation of the double-pendulum-based model to the experimental data gathered with the help of real patients.

At all the stages of the cycle, sensitivity analysis can be performed to identify the influence of parameters in the model formulation (Henninger and Reese, 2010). Before the validation stage, it helps to single out critical parameters. Performed after validation, it ascertains whether or not experimental results lie within initial estimates. Here, verified methods help to avoid elaborate Monte-Carlo or parameter variation techniques and provide a possibility to deal with whole ranges for parameters instead of concrete values.

Usually, developers have no absolute information about the system to be engineered and its data due to either the natural variability inherent to phys-

ical phenomena (aleatory, irreducible) or simply lack of knowledge (epistemic, reducible) (Ferson et al., 2003). That is, the whole modeling and simulation process is influenced by uncertainty. It is necessary to characterize model or parameter uncertainty at the stage of qualification, propagate it through the system and take into account discretization and rounding errors at the stage of verification, and do not leave it out while defining validation metrics at the last stage. The propagation can be formalized by defining mathematical or set theoretical operations along with logical expressions on uncertain numbers. Besides, higher level algorithms such as for solving initial value problems might become necessary. Generally, it was suggested to use probability theory based approaches for purely irreducible uncertainty and interval methods for purely reducible one. For combinations, such methods as probability boxes, Dempster-Shafer belief theory, and fuzzy set theory can be employed.

Interval methods offer a natural and comparatively simple instrument for specifying the overall imprecision in the outcome given the bounds on imprecision in inputs. It is simple because set theoretical or mathematical operations on intervals are easy to define, implement and interpret. To define logical operations such as $<$ is more difficult conceptually, because the set of interval numbers is not ordered in a natural way, although a partial order can be readily introduced. Besides, most known numerical types of algorithms were adapted to intervals. A disadvantage of fast and naive interval approaches is the already mentioned problem of overestimation, which is especially noticeable for multiple appearance of the same intervals (with large widths) in mathematical expressions. That means that interval methods cannot produce meaningful results for problems with parameters having known outliers of a considerable magnitude in principle. However, the dependency problem and the wrapping effect can be countervailed by enhanced verified methods such as affine arithmetic (de Figueiredo and Stolfi, 2004) or Taylor models.

The terminology defined above builds the basis for our further considerations. The paper is structured as follows. In Section 2, we provide detailed process-oriented verification guidelines supplemented by a questionnaire. An important aspect of verification stage in the V&V cycle is software quality analysis and the subsequent software choice. To facilitate this decision making process for the user in the area of dynamic simulations, we developed a comparison system for verified initial value problem (IVP) solvers, described in Section 3. Here, a special focus lies on validating the employed similarity measure for benchmark problems. Finally, conclusions are in Section 4.

2 PROCESS-ORIENTED VERIFICATION GUIDELINES

Here, we suggest an enhancement of the usual verification methodology in the form of process-oriented verification guidelines. As mentioned in the introduction, verification stage comprises numerical result verification and the software quality analysis supported by testing process. Software quality analysis is primarily directed toward establishing programming correctness in the source code and in the system and compiler software as opposed to numerical verification, the goal of which is to assess the correspondence between the outcomes of the computerized and formal model. Numerical result verification presented in, for example, (Oberkampf and Trucano, 2007), is mainly focused on classical mathematical procedures such as proofs for algorithms' error estimations, on analytically solvable special cases, and on benchmark examples. We suggest to perform the result verification step for the appropriately decomposed modeling and simulation process for the targeted product using methods with result verification. This can be complemented by uniform strategies for sensitivity analysis and uncertainty management at each step of the respective V&V cycle. The approach is illustrated in (Auer et al., 2012) for the computer-based hip prosthesis fitting process.

For the numerical verification stage of a complex modeling and simulation process to be successful, we propose to pursue the following four strategic goals during the whole production cycle for a certain piece of software (computerized model).

The *first goal* concerns designing the software for the problem under consideration in such a way as for it to be verifiable more easily later. It starts with an informal description of the workflow to structure the process in relevant sub-tasks. After that, application context and ontology should be identified. After choosing the model for the problem, it is necessary to define its features along with requirements for it. The requirements concern, for example, functionalities of the targeted software; expected performance and necessity for parallelization or real-time execution; GUI/front-end design and configuration along with user interaction (e.g. online or offline control facilities). For the later stage of verification, the following specifications are especially important:

- exchange formats, schemes for meta data, their integration into the software,
- data transfer, bandwidth, network, security, and
- documentation (reviews), repository.

Once the software is released, further requirements address integration of middleware; software config-

uration (settings) suitable for the task, calibration and data flashing; customizing and user training.

The *second goal* is to define guidelines for developing a computational model and its formal description for the system. The guidelines should concern system classification (technical, cognitive, or for information processing), knowledge representation (the system model and interaction logic), action logic (symbolic/algorithmic, functional/procedural, logical, Petri net, graph or grammar based), and hardware architecture (CPU/GPU, distributed system, etc.).

The *third goal* should focus on the verification extent and the congruence assessment between the computerized and formal models. Based on the information acquired while persuing the above two goals, we suggest assigning a certain degree to the modeling and simulation process from the four-tier numerical verification hierarchy. This degree reflects the ratings of all sub-processes and the kind of uncertainty assessment. The initial estimation can be improved using a selection of specialized data types and tools which are chosen in dependance of what kind of action logic was employed. For example, if the symbolic type was used, that is, actual expressions for the mathematical model are available, there exist verified tools in each important numerical algorithms area such as solving systems of equations, initial value problems, and optimization.

In this regard, interoperability with respect to data types, algorithms and used hardware is of a great relevance for developing highly verified software. Generally, modern programming languages support templates and implementation patterns which allow for adaptive use of real, floating point, interval, fuzzy, or stochastic data types and algorithms. In this way, it is possible to choose the kind of computation according to the task at hand, to perform different kinds of computations (e.g. interval or floating point) in parallel, and to combine them if it is not feasible to carry out the whole computation using just one of them.

The *fourth goal* is devoted to user support. Here, we suggest to employ recommender systems similar to those from the area of e-commerce. They should guide the user in questions of interactive task solver selection, of result reporting, and of visualization.

The V&V assessment should be regulated by questionnaires. They aid in organizing a complex process with respect to initial data sources and their accuracy as well as its parameter and result ranges. The questionnaires should require precise description of special functions, algorithms (e.g., as a UML activity diagram), and employed mathematical operations. In this way, they help to make process-wide decisions about adequate data types and arithmetic, software quality testing, numerical result quality measures, and performance issues. Below, we give an example of such a questionnaire which appeared in the scope of the recent project PROREOP[1].

In PROREOP, a consortium of engineers, biomechanicians, computer and medical scientists developed and evaluated new tools in training, planning and assessment of total hip replacement (THR), which was broadly discussed from the perspective of our working group in (Auer et al., 2011). The above mentioned questionnaire was distributed among the participants to identify accuracy issues of various algorithms used in the project. By describing the algorithms with respect to their data types, to initial and resulting data, and to functionality, each group that filled out the questionnaire contributed to the assessment of accuracy and correctness for the corresponding sub-process. The development of a requirement pool helped us to bring all project parts into a coherent software architecture. An important example is a superquadric fitting algorithm with parameters defining scaling, roundness, orientation, position and deformation (tapering, bending, cavity), which was ranked initially as belonging to Class 3 in the hierarchy and was assigned a Class 2 degree after parts of it had been verified (Auer et al., 2011).

We reproduce the questionnaire with some abridgements and improvements. Its filled-out version used in PROREOP for the reconstruction and fitting approach is given in (Auer et al., 2012). The first part is the *description of input data* with questions concerning its source, description form, possible pre-selection technology, the corresponding accuracy, and representation. The second part urges to characterize the involved *models* using their formulas along with the respective parameters and their ranges. This part is followed by the *description of the used algorithms* by their type (numeric/symbolic etc.), the degree of parallelization, the employed elementary operations along with their precision, the necessary subalgorithms, their detailed description, and their sensitivity. Finally, the *output data* should be reported on by specifying the corresponding data type and accuracy, the external requirements for high accuracy, and possibilities for information exchange.

3 COMPARISON OF VERIFIED IVP SOLVERS

Nowadays, there exist not just one but several tools

[1]Development of a new prognosis system to optimize patient-specific preoperative surgical planning for the human skeletal system

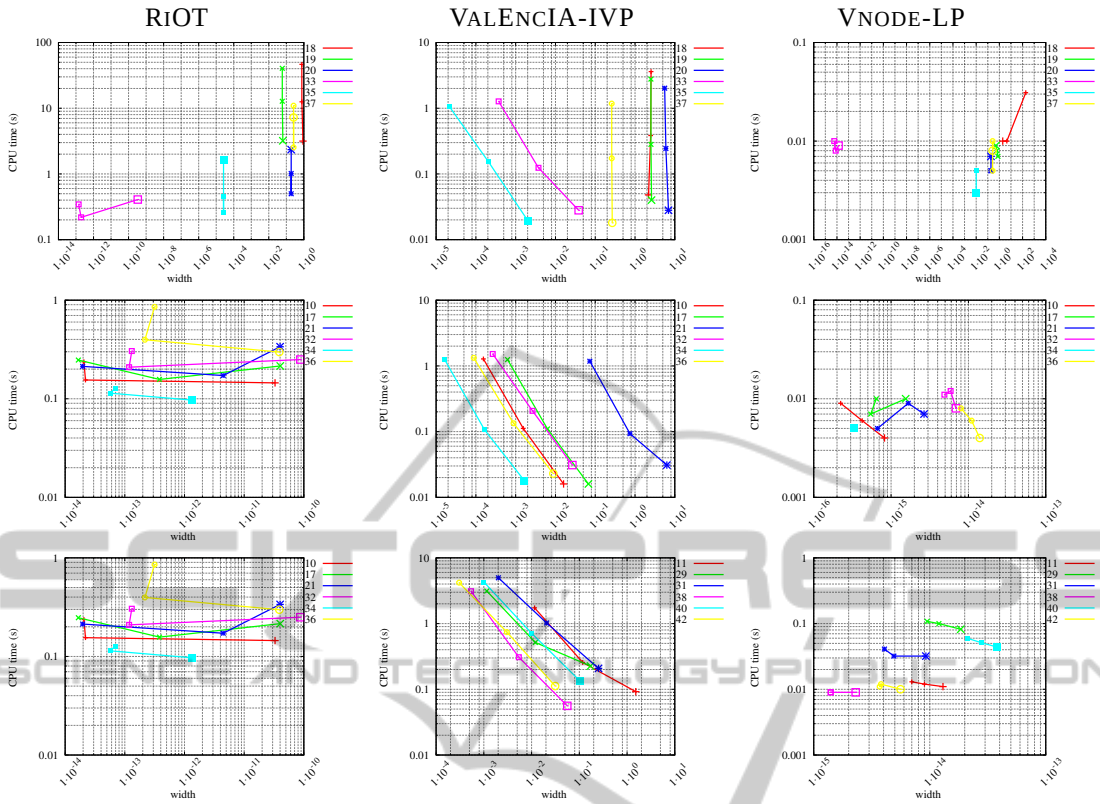

Figure 1: WPDs for three solvers and three classes of nonlinear problems: simple with uncertainty (top), simple without uncertainty (middle), moderate without uncertainty (bottom). Bigger symbols tag results obtained with the first parameter set.

to solve a certain task. Users have to make decisions about what program is the best for their problem. This choice is not simple since modern tools have many settings with which they can be finely tuned. However, the tuning usually requires some in-depth knowledge of underlying algorithms, for learning which an average user might have no time.

For example, a long-term involvement of out working group with verified modeling and simulation of dynamic systems revealed the need for comparison of verified initial value problem solvers. During the last decades, a number of programs have been developed for this purpose such as VNODE-LP, RIOT, VALENCIA-IVP and others. The goal of such software is to provide as tight enclosures as possible over a sufficiently long time span. Whether they succeed depends on different factors, for example, the type of the problem or the presence of explicitly uncertain parameters. Despite successful attempts at using them in practice, a typical engineer is not interested in them since their naive use often leads to too pessimistic results due to overestimation.Therefore, (preferably automatic) comparisons between tools are necessary so that users can choose the right one easily.

In (Auer and Rauh, 2012), we introduced an online framework for comparisons between verified initial value problem solvers. The conceptual basis is derived from the previous work on floating-point based solvers (Hull et al., 1972), (Mazzia and Iavernaro, 2003). However, verified solvers have to be compared differently. For example, since they always produce correct results, we do not need to access the number of incorrect solutions. On the other hand, the criterium of computing time gains in importance because verified solvers are in general slower than usual ones. To save user's time, we developed a recommender front-end (online soon) which suggests a solver and its appropriate settings for a given initial value problem without the need to test all the available solvers and settings with it. The recommendation is based on the similarity of the user's problem to one from our database for which full test results are available. The similarity is established semi-automatically using the problem classification described in detail in (Auer and Rauh, 2012). In this way, we contribute to the software quality analysis and user support at the verification stage of the V&V cycle (dynamic simulations).

Such comparisons and guidance are not possible without a representative set of examples. In (Auer and Rauh, 2012), we worked with a small database of

12 systems which represented the following classes of non-stiff problems: linear/nonlinear, in each of them simple/moderate/complex, in each of them with/without uncertainty. This classification makes sense since there exist theoretical results about differences in performance of solvers for linear and nonlinear problems. The presence of uncertainty and the dimension of the problem (which is one of the factors defining the numeric complexity) seem to play an important role. These claims can be corroborated empirically if the problem database is big enough.

As of now, there are over 45 examples in our database, mainly simple and moderate systems in both linear and nonlinear classes. In Figure 1, we show work-precision diagrams (WPD) for three available solvers (RIOT on the left, VALENCIA-IVP in the middle, VNODE-LP on the right) and three sets of parameters for each. We tested the solvers on all nonlinear problems from the classes simple/with uncertainty (top), simple/without uncertainty (middle), moderate/without uncertainty (bottom). For more information about the used problems (given by the number in the figure) and the conditions of the test, visit `vericomp.inf.uni-due.de`. The parameter sets we vary for the three tools are different in their nature. Therefore, the CPU time always increases from parameter set one to parameter set three for VALENCIA-IVP (the only solver without an automatic step size control) since we decrease the stepsize, which is not always true for the other two solvers where we modify the order of the Taylor expansion.

From the figure, we observe that all three solvers perform differently for problems with and without uncertainty (the top and middle WPDs). Whereas VNODE is best with respect to CPU times in both cases, it produces considerably less tight enclosures than RIOT for uncertain problems. For the same examples without uncertainty, the quality of enclosures of these two solvers is almost equal. There is also a difference between the classes of simple and moderate problems: for example, RIOT with the third set of settings is mostly better with respect to the enclosure width for the bottom figure, which does not hold for the middle one. The figures show that each class generates a distinct solver behavior, giving us a basis for employing the classification as the similarity measure while recommending a solver for a specific example.

4 CONCLUSIONS

We presented process-oriented guidelines for verification of biomechanical problems. In the scope of software quality analysis, we described an online system for comparison of verified initial value problem solvers. This provides broader user support in the area of numerical verification with (extended) interval methods, helping to raise awareness of verified tools in engineering.

REFERENCES

AIAA (1998). Guide for the Verification and Validation of Computational Fluid Dynamics Simulations. *American Institute of Aeronautics and Astronautics*.

ASME (2006). Guide for Verification and Validation in Computational Solid Mechanics. *American Society of Mechanical Engineers*, pages 1–15.

Auer, E., Chuev, A., Cuypers, R., Kiel, S., and Luther, W. (2011). Relevance of Accurate and Verified Numerical Algorithms for Verification and Validation in Biomechanics. In *EUROMECH Colloquium 511*, Ponta Delgada, Azores, Portugal.

Auer, E. and Luther, W. (2009). Numerical Verification Assessment in Computational Biomechanics. In *Proc. of Dagstuhl Seminar 08021*, LNCS, pages 145–160.

Auer, E., Luther, W., and Cuypers, R. (2012). Process-Oriented Verification in Biomechanics: A Case Study. In *Proc. of SUM 2012*, Marburg, Germany. Submitted.

Auer, E. and Rauh, A. (2012). VERICOMP: A System to Compare and Assess Verified IVP Solvers. *Computing*, 94(2):163–172.

Berz, M. (1995). Modern Map Methods for Charged Particle Optics. *Nuclear Instruments and Methods A363*, pages 100–104.

de Figueiredo, L. H. and Stolfi, J. (2004). Affine Arithmetic: Concepts and Applications. *Numerical Algorithms*, 34(1-4):147–158.

Ferson, S., Kreinovich, V., Ginzburg, L., Myers, D. S., and Sentz, K. (2003). Constructing Probability Boxes and Dempster-Shafer Structures. Technical Report SAND2002-4015, Sandia National Laboratories.

Henninger, H. and Reese, S. (2010). Validation of Computational Models in Biomechanics. In *Proc. of the Institution of Mechanical Engineers*, volume 224, pages 801–812.

Hull, T. E., Enright, W. H., Fellen, B. M., and Sedgwick, A. E. (1972). Comparing Numerical Methods for Ordinary Differential Equations. *SIAM Journal on Numerical Analysis*, 9(4):603–637.

Mazzia, F. and Iavernaro, F. (2003). Test Set for Initial Value Problem Solvers. Technical Report 40, Department of Mathematics, University of Bari, Italy.

Moore, R. E., Kearfott, R. B., and Cloud, M. J. (2009). *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia.

Oberkampf, W. and Trucano, T. (2007). Verification and Validation Benchmarks. *Nuclear Engineering and Design*, 238(3):716–743.

Schlesinger, S. (1979). Terminology for Model Credibility. *Simulation*, 32(3):103–104.