`des-ist`: a simulation framework to streamline event-based *in silico* trials

Max van der Kolk[1,3][0000−0001−5761−5776], Claire Miller[1][0000−0003−0758−9447], Raymond Padmos[1][0000−0001−7253−240X], Victor Azizi[2], and Alfons Hoekstra[1][0000−0002−3955−2449]

[1] Computational Science Laboratory, Informatics Institute, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, The Netherlands
[2] Netherlands eScience Center, Science Park 140, 1098 XG, Amsterdam, The Netherlands
[3] m.vanderkolk@uva.nl

Abstract. To popularise *in silico* trials for development of new medical devices, drugs, or treatment procedures, we present the modelling framework `des-ist` (Discrete Event Simulation framework for *In Silico* Trials). This framework supports discrete event-based simulations. Here, events are collected in an acyclic, directed graph, where each node corresponds to a component of the overall *in silico* trial. A simple API and data layout are proposed to easily couple numerous simulations by means of containerised environments, i.e. Docker and Singularity. An example *in silico* trial is highlighted studying treatment of acute ischemic stroke, as considered in the INSIST project.
The proposed framework enables straightforward coupling of the discrete models, reproducible outcomes by containerisation, and easy parallel execution by `GNU Parallel`. Furthermore, `des-ist` supports the user in creating, running, and analysing large numbers of virtual cohorts, automating repetitive user interactions. In future work, we aim to provide a tight integration with validation, verification and uncertainty quantification analyses, to enable sensitivity analysis of individual components of *in silico* trials and improve trust in the computational outcome to successfully augment classical medical trials and thereby enable faster development of treatment procedures.

Keywords: Event-based modelling · *in silico* trials · ischemic stroke.

1 Introduction

Research and development of new medical devices, drugs, or treatment procedures requires significant monetary and temporal resources [2]. Recent investigations estimate post-approval Research & Development (R&D) costs to average 985 million US dollars [21]. Regardless of careful planning, time-to-market [13] and R&D costs are increased by trial difficulties such as statistical uncertainties, trials lacking a clear understanding of their outcome, and other unforeseen side-effects caused by a newly proposed drug, device, or procedure [3].

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53

To counteract these challenges, researchers are adopting computational biomedicine to augment traditional *in vitro* and *in vivo* trials. These *in silico* experiments are enabled by recent developments in computational modelling in biomedicine [19]. The so-called *In Silico* Clinical Trials (ISCTs) [18] specifically consider cohorts of virtual patients representing highly specific population subsets and thereby enable (*in silico*) clinical trials considering rare diseases with reduced costs as development times decrease.

However, defining ISCTs is not without cost or complexity either [18]. Typically, accurate *in silico* simulations require detailed modelling across time and length-scales [6, 16] and close collaboration of experts—i.e. clinicians, experimentalists, and software developers—spanning multiple fields of research [4]. Additionally, the numerical models require strict scrutinisation with advanced validation, verification and uncertainty quantification (VVUQ) analyses to gain sufficient trust in their outcome [20, 22]. Finally, there are practical issues to address, e.g. interfacing multiple *in silico* models, reproducibility, and supporting various computational environments from personal workstations to large-scale cloud or High Performance Computing (HPC) environments [5].

In this work, we propose a simulation framework: `des-ist`, that addresses these difficulties in managing ISCTs. Specifically, it interfaces multiple *in silico* models using a predetermined data layout in combination with a simple, unified Application Programming Interface (API), which is enforced across each component of the *in silico* simulation pipeline. Furthermore, the simulation is split in a series of components, each captured in a containerised environment using `Docker` [11] or `Singularity` [9], to ensure reproducibility of the *in silico* trials. An additional benefit of these containerised environments is their ability to scale well towards cloud-based or HPC compute environments. Our application `des-ist` then orchestrates running these *in silico* trials, thereby enabling parallelism using `GNU Parallel` [17] and VVUQ analyses by interfacing `EasyVVUQ` [22].

To illustrate our proposed workflow, we present a numerical experiment from the *In Silico* clinical trials for acute Ischemic STroke (INSIST) project [8]. We show how `des-ist` supports the user in setting up and evaluating an ISCT pipeline. The example illustrates a detailed, event-based *in silico* pipeline studying acute ischemic stroke. The pipeline considers a containerised environment for each distinct event identified in the clinical trial, i.e. virtual patient generation, initial randomised clot placement, blood-flow and brain (re)perfusion analysis [7, 12], chemical and mechanical treatments using thrombolysis [15] and thrombectomy [10], and finally the NIHSS scoring of each virtual patient [1].

The paper continues in Section 2 with a discussion on the chosen data layout and API definition as adopted in `des-ist`. Next, we run a single *in silico* pipeline in Section 3 and close with brief discussion and conclusions in Section 4.

2 Methods

In general, ISCTs can be separated into three distinct phases: *i)* generating statistical representative cohorts of virtual patients, *ii)* the analysis of a simulation

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53

driven pipeline *per* virtual patient, and *iii)* the (post-)processing of statistical data as generated by the *in silico* trials. The first and last steps are typically driven by statistical models. These are carefully formulated to ensure they generate sets of virtual patients that accurately represent a subset of interest of the studied population. The second step, i.e. evaluating the *in silico* experiment per patient, differs strongly depending on the type of the ISCT and chosen computational environment. Here, we will specifically consider *in silico* trials represented by a sequence of discrete events, where each event is assumed to instantaneously change the system's state. The events are stored in `des-ist` using a directed, acyclic graph, as conceptually illustrated in Fig. 1. The nodes in the graph represent to discrete events and each holds a reference to the matching container that needs to be invoked for the specific time instance. The graph is traversed by `des-ist` following the directed edges and thereby covering all individual events of the *in silico* trial.

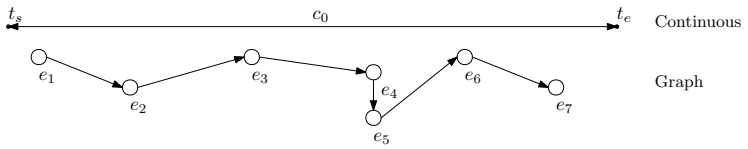

Fig. 1. A continuous time system c_0 is modelled in `des-ist` as a sequence of discrete events e_i, where each event is assumed to instantaneously change the system's state. The discrete events are stored in a directed, acyclic graph, implicitly storing the traversal order as well as a reference to the containerised simulation environment for each event e_i. Then, `des-ist` traverses the graph and invokes the simulation per event.

To combine various containers within a single ISCT, `des-ist` enforces an API and data layout requirements for each container. The API is written in Python—a programming language well suited for writing *glue* code [14]. The API contains three functions that need to be implemented for each container, i.e. `event`, `example`, and `test`. Most critically an implementation has to be written for the `event` call, which invokes the container's main simulation[4].

To ensure a container only has access to a given virtual patient's data, we enforce a specific data layout in combination with the presented API. Each trial has its own directory, e.g. `/trial`, containing a trial's configuration file and a subdirectory per virtual patient in the cohort, i.e. for patient i: `/trial/patient_i`. To ensure a container's access is limited to data of a specific virtual patient, we exploit *bind paths*—a feature present in both `Docker` [11] and `Singularity` [9]—where we map a constant path on the container to a specific path on the host system, e.g. `/patient/` ⇒ `/host/trial/patient_i/`, simplifying the API. To control the working directories of the containers, `des-ist` only has to update these *bind paths* accordingly, leaving the containers unchanged.

[4] Non-Python code is included using Python's `subprocess` library https://docs.python.org/3.10/library/subprocess.html.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53

When creating a trial using `des-ist`'s command-line interface, the directory structures are automatically generated and provided with the corresponding trial and patient configuration files. For these files, we adopt the YAML format (https://yaml.org/), a format easy to modify by hand and automatically.

2.1 Containerised Environments

Adopting containerised environments provides numerous advantages in context of ISCTs. First of all, the separated containers allow for easy collaboration in large research consortia where various researchers—typically spread over many countries—can use their preferred environments without worrying about linking or interfacing of the multitude of simulation components, as used in INSIST [8]. As long as they adhere to the defined data layout and API, they can adopt any computational framework that is suited best for their specific *in silico* model.

Secondly, the strict container definitions imposed by `Docker` and `Singularity` ensure clear documentation on the container's package dependencies and requirements. These definitions directly enable reproducible research, as the containers can be shared and evaluated by others, thereby overcoming portability problems [9, 11]. Concurrently, the containerised environments enable scaling towards cloud and HPC environments using `Docker` and `Singularity` technologies [9, 11].

Another advantage by adhering to a unified API in event-based simulation, is the plug-and-play nature of the containers. This provides easy exchange of containers in a pipeline. For instance, a high fidelity model can be exchanged for a reduced order surrogate, simply by changing the trial's configuration file, and thereby the graph of events (Fig. 1). Alternatively, one can study the effect of treatment procedures by providing alternative containers for each variant of the treatment. Again, only the trial's configuration needs to be updated.

2.2 Parallelism

To decrease the wall time of *in silico* experiments, we can resort towards parallel evaluation of the *in silico* experiments per patient. This can be achieved in a variety of ways, considering small to large-scale cloud computing or HPC environments. In `des-ist` only the most basic parallelism has been considered so far, where each patient is evaluated on a single Central Processing Unit (CPU), where the distribution and balancing of tasks is assigned to `GNU Parallel` [17] to enable parallelism on single and multi-machine (HPC) architectures (Fig. 2).

3 Example

To illustrate `des-ist`'s use case, we consider the procedure of evaluating an event-based *in silico* trial. For this, we borrow a pipeline from the INSIST project [8], regarding the study of acute ischemic stroke, where the details of the *in silico* trial are omitted for brevity. A schematic illustration is shown in Fig. 3. These *in silico* trials are then generated automatically by `des-ist`, using a simple configuration

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53

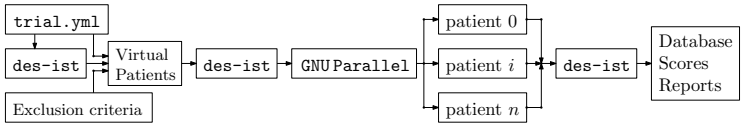

Fig. 2. Conceptual parallel execution approach in `des-ist`. The trial is initialised by `des-ist` considering a trial configuration, i.e. the `trial.yml`, to create a cohort of virtual patients using a specific statistical model. Then, `des-ist` traverses the event graph for each patient, where the simulations are executed (in parallel) by `GNU Parallel` [17]. Ultimately, the output of each simulation is collected and trial output is generated.

file containing, for example, the cohort size, in/exclusion criteria, and random seeds. From there, `des-ist` automates all repetitive, and often error-prone, user interactions to generate the required data layout (Section 2). Furthermore, `des-ist` orchestrates simulation runs, automatically scheduled on the available resources through `GNU Parallel`. For each patient the sequence of discrete events (Fig. 3(e)) is evaluated, sequentially stepping through each simulation. Ultimately, all results are collected and statistical outputs are aggregated over the considered cohort, providing insight into the trial's outcome, such as infarct volume, treatment outcome, and NIHSS distributions across the cohort of virtual patients.

4 Conclusion

To transition towards ISCTs for development of new drugs, medicine, and treatment procedures, we propose a numerical framework: `des-ist` to support users in running large numbers of *in silico* trials. The framework automates generating cohorts of virtual patients, ensures all *in silico* simulations are evaluated in the desired order by traversing an internal acyclic directed graph representing the *in silico* pipeline, enables parallel scheduling by `GNU Parallel`, and provides VVUQ analysis through `EasyVVUQ`. These features greatly reduce the repeated efforts of users in setting up, evaluating, and validating large *in silico* trials.

Additionally, we propose the adoption of containerised environments, using `Docker` or `Singularity`, to capture the individual, discrete simulation events. These containers are combined with a simple API and data layout to enable uniform data communication between the various simulation events. This has not only increased the reproduciblity of the *in silico* trials' simulations, it also reduces barriers for collaboration by reducing compatibility issues and enables scaling towards (large-scale) cloud or HPC compute environments.

Although only an illustrative *in silico* pipeline from the INSIST project was presented, we hope to convey the potential of orchestrating *in silico* trials using `des-ist`. In the near future, we aim to generalise the current implementation to support a variety of *in silico* pipelines and pursue validation of ISCTs. The `des-ist` application is to be released mid 2021 as open source software.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53

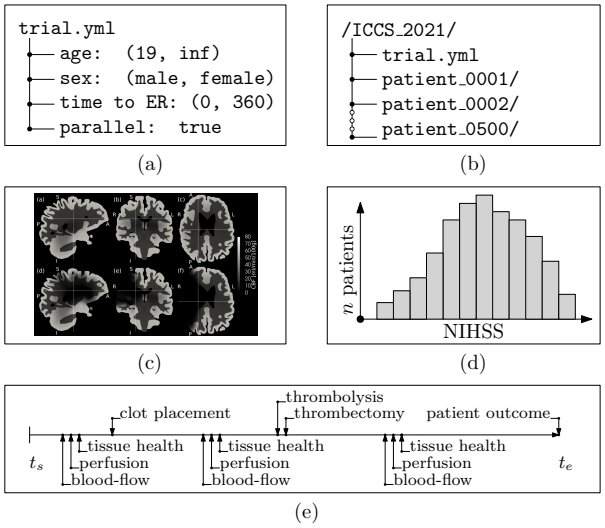

Fig. 3. Illustration of the steps in an *in silico* trial using `des-ist`. First, a trial configuration file is provided (a), containing properties of the trial. Then, `des-ist` creates a directory structure and data layout (b): a trial directory with a subdirectory per patient, after which each patient's simulation pipeline (e) is evaluated (Fig. 2). Finally, the results are aggregated providing per patient results (c) of brain perfusion and trial reports with statistical output measurements of the *in silico* trial (d).

INSIST [8] has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 777072.

References

1. Brott, T., et al.: Measurements of acute cerebral infarction: a clinical examination scale. Stroke **20**(7), 864–870 (jul 1989). https://doi.org/10.1161/01.str.20.7.864
2. Dickson, M., Gagnon, J.P.: Key factors in the rising cost of new drug discovery and development. Nature Reviews Drug Discovery **3**(5), 417–429 (may 2004). https://doi.org/10.1038/nrd1382
3. Fogel, D.B.: Factors associated with clinical trials that fail and opportunities for improving the likelihood of success: A review. Contemporary Clinical Trials Communications **11**, 156–164 (sep 2018). https://doi.org/10.1016/j.conctc.2018.08.001
4. Groen, D., Zasada, S.J., Coveney, P.V.: Survey of multiscale and multiphysics applications and communities. Computing in Science & Engineering **16**(2), 34–43 (mar 2014). https://doi.org/10.1109/mcse.2013.47
5. Gupta, A., Milojicic, D.: Evaluation of HPC applications on cloud. In: 2011 Sixth Open Cirrus Summit. IEEE (oct 2011). https://doi.org/10.1109/ocs.2011.10
6. Hoekstra, A., Chopard, B., Coveney, P.: Multiscale modelling and simulation: a position paper. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences **372**(2021), 20130377 (aug 2014). https://doi.org/10.1098/rsta.2013.0377

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53

7. Józsa, T.I., Padmos, R.M., Samuels, N., El-Bouri, W.K., Hoekstra, A.G., Payne, S.J.: A porous circulation model of the human brain for in silico clinical trials in ischaemic stroke. Interface Focus **11**(1), 20190127 (dec 2020). https://doi.org/10.1098/rsfs.2019.0127
8. Konduri, P.R., Marquering, H.A., van Bavel, E.E., Hoekstra, A., Majoie, C.B.L.M.: In-silico trials for treatment of acute ischemic stroke. Frontiers in Neurology **11**, 1062 (2020). https://doi.org/10.3389/fneur.2020.558125
9. Kurtzer, G.M., Sochat, V., Bauer, M.W.: Singularity: Scientific containers for mobility of compute. PLOS ONE **12**(5), 1–20 (05 2017). https://doi.org/10.1371/journal.pone.0177459
10. Luraghi, G., et al.: Applicability assessment of a stent-retriever thrombectomy finite-element model. Interface Focus **11**(1), 20190123 (2021). https://doi.org/10.1098/rsfs.2019.0123
11. Merkel, D.: Docker: Lightweight linux containers for consistent development and deployment. Linux J. **2014**(239) (Mar 2014)
12. Padmos, R.M., Józsa, T.I., El-Bouri, W.K., Konduri, P.R., Payne, S.J., Hoekstra, A.G.: Coupling one-dimensional arterial blood flow to three-dimensional tissue perfusion models for in silico trials of acute ischaemic stroke. Interface Focus **11**(1), 20190125 (dec 2020). https://doi.org/10.1098/rsfs.2019.0125
13. Prašnikar, J., Škerlj, T.: New product development process and time-to-market in the generic pharmaceutical industry. Industrial Marketing Management **35**(6), 690–702 (aug 2006). https://doi.org/10.1016/j.indmarman.2005.06.001
14. van Rossum, G.: Glue it all together with python (1998), https://www.python.org/doc/essays/omg-darpa-mcc-position/
15. Shibeko, A.M., Chopard, B., Hoekstra, A.G., Panteleev, M.A.: Redistribution of tpa fluxes in the presence of pai-1 regulates spatial thrombolysis. Biophysical Journal **119**(3), 638–651 (2020). https://doi.org/https://doi.org/10.1016/j.bpj.2020.06.020
16. Sloot, P.M., Hoekstra, A.G.: Multi-scale modelling in computational biomedicine. Briefings in Bioinformatics **11**(1), 142–152 (dec 2009). https://doi.org/10.1093/bib/bbp038
17. Tange, O.: Gnu parallel - the command-line power tool. ;login: The USENIX Magazine **36**(1), 42–47 (Feb 2011). https://doi.org/10.5281/zenodo.16303
18. Viceconti, M., Henney, A., Morley-Fletcher, E.: In silico clinical trials: how computer simulation will transform the biomedical industry. International Journal of Clinical Trials **3**(2), 37–46 (2016), https://www.ijclinicaltrials.com/index.php/ijct/article/view/105
19. Viceconti, M., Hunter, P.: The virtual physiological human: Ten years after. Annual Review of Biomedical Engineering **18**(1), 103–123 (2016). https://doi.org/10.1146/annurev-bioeng-110915-114742
20. Viceconti, M., Juarez, M.A., Curreli, C., Pennisi, M., Russo, G., Pappalardo, F.: Credibility of in silico trial technologies—a theoretical framing. IEEE Journal of Biomedical and Health Informatics **24**(1), 4–13 (jan 2020). https://doi.org/10.1109/jbhi.2019.2949888
21. Wouters, O.J., McKee, M., Luyten, J.: Estimated research and development investment needed to bring a new medicine to market, 2009-2018. JAMA **323**(9), 844 (mar 2020). https://doi.org/10.1001/jama.2020.1166
22. Wright, D.W., et al.: Building confidence in simulation: Applications of easyvvuq. Advanced Theory and Simulations **3**(8), 1900246 (2020). https://doi.org/https://doi.org/10.1002/adts.201900246

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77967-2_53