The Pegasus Workflow Management System: Translational Computer Science in Practice

Ewa Deelman[a,*], Rafael Ferreira da Silva[a,*], Karan Vahi[a], Mats Rynge[a], Rajiv Mayani[a], Ryan Tanaka[a], Wendy Whitcup[a], Miron Livny[b]

[a] *University of Southern California, Information Sciences Institute, Marina del Rey, CA, USA*
[b] *University of Wisconsin at Madison, Madison, WI, USA*

Abstract

Translational research (TR) has been extensively used in the health science domain, where results from laboratory research are translated to human studies and where evidence-based practices are adopted in real-world settings to reach broad communities. In computer science, much research stops at the result publication and dissemination stage without moving to the evaluation in real settings at scale and feeding the gained knowledge back to research. Additionally, there is a lack of steady funding and incentives to broadly promote translational computer science (TCS) in practice. In this paper, we present how, throughout its lifespan, the Pegasus workflow management system project has incorporated the principles of translational computer science. We report on our experience on building a strong, long-term engagement with a broad range of science communities to establish mutually beneficial relationships between the core R&D team and these communities.

Keywords: Translational Research, Translational Computer Science, Scientific Workflows, Workflow Management Systems, Large-scale Distributed Computing

1. Introduction

The success of modern biomedicine can often be attributed to the workflow adopted by researchers to develop new pharmaceutical compounds via computations and in a laboratory environment, test them within clinical settings, and develop protocols and guidelines for broader community usage of best therapies. The success of this translational approach has brought focus to the need of progressive testing of solutions in real settings, and the need to feed the insights gained (both negative and positive) back to the basic research effort in order to improve research outcomes [1, 2]. Over the past decade, several institutions and funding bodies have supported translational research (TR) as the foundation for transferring research outcomes into practice. For instance, the National Institute of Health (NIH) has made translational research a priority, forming centers of translational research at its institutes and launching specific, TR-focused programs [3]. Institutions of higher-education have also conducted internal surveys to understand the impact of TR practices adopted by their faculty and research staff [4].

Although the benefits of TR have been well-acknowledged in the health sciences domain (both by scientists and funding bodies), the traditional research workflow in computer science (CS) still does not have well-defined best practices. Additionally, there is a lack of support in CS for embracing TR as part of the critical path of the research development process [5]. On the other hand, as CS permeates not only traditional physical sciences, but also social sciences, humanities, and in fact every aspect of our lives, some CS researchers see the value of examining the entire innovation cycle from conceptualization and initial research to testing at scale in real environments to broad community adoption. In this context, this paper examines translational computer science (TCS) as defined by Abramson and Parashar in [5], and describes how this process of innovation has driven the development and adoption of the Pegasus workflow management system [6, 7], and what were the challenges that the project faced over the years. Pegasus came out of an NSF-funded project that aimed to explore the concept of virtual data in science [8]. That grant specifically funded interdisciplinary research in CS and astronomy/physics. As a result, Pegasus was born out of our belief that the value of our work lay not only in the novel computer science algorithms but also in the software that leveraged these algorithms to advance domain science. Thus, the translational aspects of CS grew organically to some degree.

In this paper, we examine the various facets of TCS with the lens of the Pegasus experience, and discuss the challenges we faced for over twenty years of the project's lifespan. Specifically, we delineate how TCS principles have been incorporated into the research and development process of the Pegasus software, and how we have leveraged community engagement and feedback for driving novel software requirements and future research directions. We also discuss the challenges we have faced along these years regarding translational research, and we rec-

[*] Corresponding address: USC Information Sciences Institute, 4676 Admiralty Way Suite 1001, Marina del Rey, CA, USA, 90292

Email addresses: deelman@isi.edu (Ewa Deelman), rafsilva@isi.edu (Rafael Ferreira da Silva), vahi@isi.edu (Karan Vahi), rynge@isi.edu (Mats Rynge), mayani@isi.edu (Rajiv Mayani), tanaka@isi.edu (Ryan Tanaka), wwhitcup@isi.edu (Wendy Whitcup), miron@cs.wisc.edu (Miron Livny)

ommend potential solutions for incorporating TCS into the critical path of development and research processes.

This paper is organized as follows. Section 2 presents an overview of Pegasus and its user communities. In Section 3, we briefly describe the concepts of TCS as used in this paper. Section 4 presents our experience in designing Pegasus, and how TCS has been applied during the lifespan of the project. Section 5 discusses the challenges in performing TCS and how we have overcome them. Section 6 concludes the paper with a summary of discussions and perspectives on future directions.

2. Pegasus Workflow Management System

Scientific workflows have been almost universally used across scientific domains and have underpinned some of the most significant discoveries of the past several decades [6, 9]. Workflow management systems support abstractions and provide automation, which enable a broad range of researchers to easily define sophisticated computational processes and to then execute them efficiently on parallel and distributed computing systems. Pegasus [6] is an open source framework that is being used in production to execute scientific workflows for dozens of high-profile applications in a number of different disciplines including astronomy, gravitational-wave physics, bioinformatics, earthquake engineering, helioseismology, and limnology.

Since 2001, Pegasus has evolved into a robust and scalable system in response to the needs of scientists to conduct complex computations on heterogeneous and distributed cyberinfrastructures (CI) [7]. From the beginning, Pegasus' philosophy was to rely on established research in graph theory, databases, and compilers, and to augment and adapt that knowledge to the concept of workflows and the target CI. Today, Pegasus is built on the foundation of abstractions of directed acyclic graphs, fundamental programming constructs, and scalable algorithms. Workflow execution with Pegasus includes data management, monitoring, and failure handling, and is managed by HTCondor DAGMan [10]. Individual workflow tasks are managed by a workload management framework, HTCondor [11], which supervises task executions on local and remote resources. During execution, Pegasus translates an abstract resource-independent workflow into an executable workflow, determining the executables, data, and computational resources required for the execution.

Over the past two decades, an interdisciplinary community of users in a broad spectrum of the sciences has grown around Pegasus. Notably, Pegasus was used by the Laser Interferometer Gravitational-Wave Observatory (LIGO) collaboration for the analysis that confirmed the existence of gravitational waves [12]; and by the Southern California Earthquake Center (SCEC) to generate the first ever physics-based probabilistic seismic hazard map of Southern California [13], which estimates the probability that earthquake ground motions at a geographic location of interest will exceed some intensity measure, such as peak ground velocity or ground acceleration, over a given time period. Such estimates are useful for civic planners, building engineers, and insurance agencies, and forms the basis for building codes that can influence billions of dollars of construction each year. The engagement with (and the continuous feedback from) these communities has been fundamental for the development of a robust and efficient software framework tailored to specific community needs, all without diminishing its general applicability. For instance, with the introduction of peta-scale high-performance computing (HPC) systems, SCEC and other Pegasus users requested a more native way of executing workflows on such specialized HPC resources. The result of the feedback was the introduction of a specialized workflow execution engine, which is dynamically deployed within an HPC system and enables efficient execution of high-throughput workflows on HPC architectures via a master-worker paradigm [14]. Another example of the community driven development model is the dynamic data cleanup feature, which is a data management algorithm that schedules the removal of intermediate data from the execution site as soon as it is no longer needed by the workflow. This feature was initially requested by LIGO, but it is now a standard capability. It is enabled by default and used by the majority of Pegasus users [15].

3. Translational Computer Science

In [5], translational research in computer science is defined in an analogous way to translational research (TR) in the field of biomedical research. The authors argue that there is a parallel between the translational cycle in biomedical research that feeds back real world findings to research and that of translational computer science, where at-scale evaluation provides feedback to research. TR fosters the multidirectional integration of basic research, patient-oriented research, and population-based research [16]. TR is generally composed of the following components: (i) *bench*, which is defined as basic research, often in a laboratory setting; (ii) *bedside*, which tests the results of the research in clinical settings, often via trials with groups of patients; and (iii) *community*, where the knowledge gained can be implemented as part of protocols for patient care. The authors in [5] further argue that the findings from the trials and community practice are able to influence the basic research that is being conducted. This feedback loop distinguishes the translational process from basic research that stops at the publication of research outcomes and from commercialization efforts, which may result from successful clinical trials.

The authors of [5] consider the concept of translational research and define translational computer science (TCS) to have analogous components, namely: **laboratory**, **locale**, and **community**. The *laboratory*, which encompasses both hardware and software, is where CS research and development takes place. The *locale* is where the results of the research are evaluated and may result in changes to the research being conducted. The *community* is thought to be the users and early adopters who can help more broadly with the evaluation process.

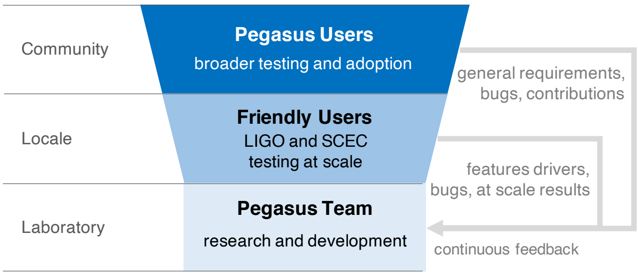

Figure 1: Mapping TCS into Pegasus R&D team and user communities.

4. Pegasus as Translational CS

Applying the principles of TCS has been key for the development of technologies that lead to important scientific discoveries. A natural, and yet often undervalued decision process for designing and realizing such technologies entails incorporating TCS throughout the projects' lifespan. This is the case for most of the long-lasting research projects that supported innovative research that has had demonstrated impact in practice.

In this section, we report on our experience gained over two decades of designing and implementing Pegasus, and how the software has supported an interdisciplinary community from a multitude of fields of science to effectively achieve their computational needs by managing the execution of computational tasks and data processing utilizing distributed, heterogeneous resources. We leverage the definition of TCS presented in [5] to describe how our processes and decisions map to the three components of the TCS model: *laboratory*, *locale*, and *community* (Figure 1).

4.1. Laboratory

The *laboratory* is the most common component among applied and experimental computer science projects [5]. It includes technical aspects of CS such as software and hardware, as well as broader infrastructure (e.g., distributed computing and storage services). In the context of Pegasus, we model the laboratory component as the Pegasus software framework and its core research and development conducted by the project team. Here, software requirements are driven by pressing community needs, while exploratory and innovative concepts are first assessed and verified within a research context. Promising solutions are then translated into software requirement specifications and eventually software capabilities are released as part of the Pegasus software.

In order to develop a robust and sustainable software product that meets the needs of the scientific community, and enables external contribution, we have adhered to proven software engineering processes. Specifically, we have developed Pegasus using Agile software development methods, where software is delivered in short, iterative, and incremental releases [7]. The Pegasus software development cycle follows a rigorous software testing process via unit tests and continuous integration. We use the latter for automated building and testing suites of workflow benchmarks when new capabilities or bug fixes are added to Pegasus. This process ensures that our code can be easily adopted and extended by the community.

Elementary software requirements can be directly implemented and incorporated into Pegasus releases by the professional software development staff (although significant development effort may be required). However, novel architectures and emerging applications may raise challenging research questions that require in-depth analysis and the development and evaluation of new techniques – e.g., optimization algorithms for managing large volumes of data, resource provisioning algorithms for dynamic scaling computing resources, among others [17, 18, 19]. Research within the Pegasus project is typically conducted alongside software development, and may involve: (i) collecting performance data from experimental (or new) computing platforms for developing performance profiles of applications and systems; (ii) development of prototypes or simulators for evaluation and verification of novel approaches; and (iii) development of a plan for incorporating research outcomes into software products. The research is often conducted by graduate students under the guidance of research staff. By fostering interactions between graduate students and the community, students gain firsthand experience in conducting research that could subsequently result in transformative software products for the community, and therefore TCS. Such interactions may require additional efforts (e.g., communication, requirements gathering, and research question refinement) than typical intra-lab research. However, we can point to a number of cases in which graduate students have benefited from external collaborations and from receiving "real-world" feedback [20, 21, 22].

Figure 2 illustrates how research and software development for Pegasus have been interleaved throughout the project's lifespan. We note that translating research outcomes into software products is a long and laborious process and typically requires several iterations between researchers and software developers (a discussion regarding this aspect is presented in Section 5). In the next sub-sections, we discuss how different types of engagement with the community have informed the design and development of the Pegasus software framework.

4.2. Locale

TCS defines *locale* as the local or virtual environment where research outcomes are evaluated in practice. In contrast to unit and integration tests commonly used as best practices in software development (e.g., in Pegasus we perform end-to-end workflow tests [7]), the goal here is to assess the behavior of the software, or its specific capabilities, under non-deterministic scenarios – e.g., at scale, subject to external load, failures, unforeseen configurations, etc. – but still obtain trustworthy responses that may lead to improvements of software components, or even the discovery of new requirements. Here, the term "at scale" targets the evaluation of the Pegasus system with large-scale applications (workflows that have hundreds of thousands jobs, processing terabytes of data) running on leadership-class HPC systems or distributed resources that have on the order of two hundred thousands cores available at any one time,

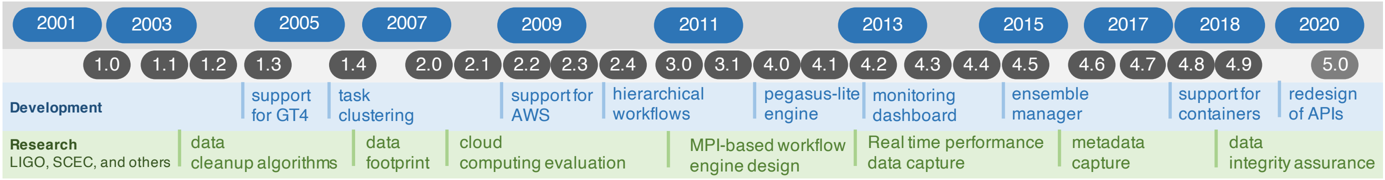

Figure 2: Pegasus development and releases over time. The bottom rows indicate interleaving research and software development. (The second row indicates software release versions.)

such as the Open Science Grid (OSG) [23]. In these evaluations, we actively monitor the performance of Pegasus to identify anomalous or unexpected patterns of operation.

In the context of Pegasus, we map the locale component to a subset of users, named "friendly users" (Figure 1), which are willing to, and are able to, conduct tests at scale in their own environments and have sufficient familiarity with the software to explore a range of functionalities under different constraints. Building trust within such relationships requires long-standing partnerships and commitment from both sides of the collaboration – incentives are key for establishing a mutually beneficial collaboration as discussed in Section 5. In the context of the Pegasus project, such relationships were initially established via collaborative funded projects in which Pegasus was the centerpiece to enable efficient distributed execution of the partners' applications. As Pegasus evolved into a mature and reliable software framework, novel partnerships were formed via engagements with the community (as described in Section 4.3).

During the Pegasus project lifespan, we have developed close collaborations with several classes of users, from individual researchers to large collaborations and centers. Specifically, astrophysicists (LIGO) [12] and earthquake scientists (SCEC) [13, 19], and more recently bioinformaticians, have been the main drivers of new features and research challenges in data management, resource management, data integrity, and fault tolerance, among others (Figure 2). As these projects have large-scale computing and storage demands and utilize national cyberinfrastructure resources, they are ideal use cases for testing Pegasus at scale. For instance, the development of solutions for handling large data footprint and an online monitoring dashboard were mainly steered by LIGO; approaches for managing large numbers of short running tasks and co-locating data-intensive single core workflow tasks with large-scale parallel computations were driven by SCEC (Figure 2) – additional details on how these collaborators have steered the evolution of the Pegasus software can be found in [7]. Although "friendly users" may guide the specification of new requirements, we ensure that our solutions are general as possible so that they can benefit the community at large.

Our involvement in large scale distributed computing infrastructure projects such as OSG [23] and XSEDE [24], has also contributed to Pegasus evolution. Some of our research staff are part of the operations and science support teams in these projects. As a result, these engagements have given us insights into the upcoming infrastructure developments, and allowed us to leverage community-developed technologies for better data management. Involvement with the Extended Collaborative Support Services team at XSEDE, gave us insight on the challenges of setting up workflow systems on large supercomputers and pushed us to explore third party provisioning tools and alternative job submission mechanisms.

4.3. Community

The *community* constitutes the third pillar of TR, which in applied or experimental computer science projects is very limited [5]. Most projects limit their research outcomes to peer-reviewed publications, to the release of software via an open-source license, or as a conduit to creating a startup. In the context of the Pegasus project, the community represents the various users and early adopters of the software framework (Figure 1).

Pegasus' approach for translating research to the community is manifold. In addition to providing an open-source code base for Pegasus, which is accompanied by comprehensive documentation of features, APIs, tutorials, and example workflows; we have continually engaged with the community on a number of fronts. Research outcomes are primarily documented and made available via publication in peer-reviewed conferences and journals. We conduct in-person and online training in conferences/meetings as well as dedicated sessions targeting particular user communities (e.g., climate modelers, bioinformatitians), and bi-monthly virtual "office hours". Mailing lists and interactions with individual users are also key for achieving successful TCS.

In terms of the software development aspect, the community has a key role in testing and identifying issues and limitations of functionalities the software provides. In order to obtain such feedback from the community, it is crucial that the development and research team provides timely responses to community queries with objective expectations of when and how issues (or new requirements) will be fixed (or made available), or clear explanations of the reasons why the request would not be fulfilled. We note that community feedback may take a number of different formats, including comments/assessments via traditional communication channels (mailing lists, emails, chats, in-person conversations), or more recently by leveraging the open source model in which issues tracking and direct contributions (e.g., pull requests) are attributed to community members. In summary, it is important to acknowledge that the quality of code and dissemination is not sufficient for achieving TCS, but also a strong engagement with the community and an understanding of their needs are vital to make the software relevant.

5. Challenges in Translational Computer Science

In the above section, we have described how the Pegasus project has attained TCS along these past two decades. Although we have successfully implemented a model in which TCS is in the critical path of our research and development processes, we have encountered several challenges that have been addressed over these years – yet several of them remain. In this section, we present these challenges, how we have approached them, and discuss potential solutions for improving the support for TCS in applied or experimental computer science projects (Figure 3).

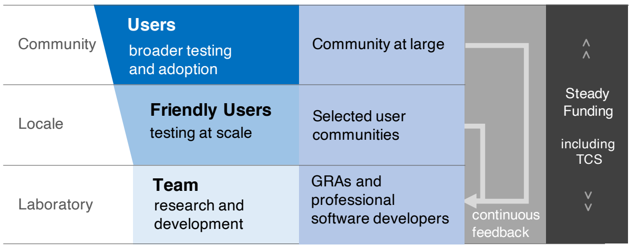

Figure 3: Envisaged TCS with continuous feedback and steady funding support. Activities related to the three pillars of TCS are performed by: the research team in the laboratory, by the friendly users in the locale, and by a broader set of users in the community. Continuous feedback from friendly users and the community inform the activities of the project team. Sustained funding is key for accomplishing TCS in the long term.

Personnel. In TR, there are often different teams conducting the research and the translation processes; in TCS, it is often the same team that performs these operations. Although there are benefits to that approach, in the sense that the insights gained from trials or dissemination can be quickly integrated into the research, the drawback is that the personnel have to fulfill both roles. In Pegasus, research is mostly performed by graduate research assistants (GRAs), the research staff, and faculty, while the development and translation is performed primarily by professional software developers – they all work closely together and translation is often done by both. Our strategy of having professional developers focusing on development and translation ensures continuity over time, while at the same time it allows the GRAs to focus on their research and progressing towards their degrees. Although we strive to also engage students in TCS, such participation is carefully scoped and managed so that the progress towards their degrees is not hindered. For instance, research publications in collaboration with the community typically involve elements of the TCS process. Therefore, we argue that there is a prevailing need for a balance in the team. However, building such a team within an academic environment is challenging. Unlike GRAs, who are relatively inexpensive and require short term (semester long) funding commitments, permanent staff is expensive and requires long term financial commitments on the part of the principal investigator. Additionally, the costs of maintaining research staff and professional developers needed to perform TCS increase over time, unlike the GRA costs, which stay relatively constant. When funding gaps within a group occur, the GRAs can be moved to teaching assistant positions, but maintaining staff during these periods can be challenging and sometimes impossible.

Funding. In CS, research projects are traditionally funded by governmental agencies that typically sponsor projects within an average time frame of 2-3 years. Although this allows research teams to produce reasonable outcomes that may enrich the state-of-the-art in the researched field or produce software elements to fulfill community pressing needs, there is no room for translating research outcomes to practice (if it is even considered at all). To overcome this challenge, the Pegasus project has been supported by a mixture of awards that target specific software development (NSF CI-focused programs), and fundamental and applied research (including NSF, DOE, NIH, and DARPA). While fundamental CS research is mostly focused on elemental aspects of novel algorithms and emerging technologies, applied research targets the distributed and heterogeneous aspect of Pegasus' applications, which drives new requirements – note that in these awards Pegasus is partly supported as a tool for enabling efficient distributed processing. Despite our ability to secure funding from different sources for Pegasus R&D along these nearly two decades, we argue that continued, steady funding (preferably from fewer sources, as expectations may significantly shift from different agencies) is vital to have the time to bring back the real world experiences to the research and the software. For instance, often times, meaningful ”field results” and scientific breakthroughs take time to come about (e.g., confirmation of the existence of gravitational waves [12] occurred 15 years after the initial collaboration between Pegasus and LIGO was established), and as a result the value of TCS is not evident in the near term or within typical funding cycles.

Evaluation. In order to support TCS, the CS community also need to value and advocate for TCS by fostering discussions around the topic, organizing workshops, and recognizing the value of TCS when participating in the proposal review and project evaluation processes. Additionally, the community needs to develop methods for evaluating the success or failure of TCS. In biomedical TR, evaluation of the translation can be measured by the ability of a drug to treat a particular condition and be compared to a control scenario. In CS the evaluation metrics are not clear and have been debated over the last few years. Are we evaluating software based on downloads, the number of users, the number of publications that used the software, the quality of the science that the software enabled? The metrics are many and not all metrics fit all TCS efforts.

Incentives. In spite of our successful model for performing TCS, we underline that early project funding awards lacked incentive for pursuing translation. In biomedicine trials, for example, subjects are sometimes incentivized with monetary rewards or the promise of a cure; in CS there is a promise of a better solution but there is also a cost of evaluating new solutions that is not necessarily compensated. Although early system (hardware) users often get the benefit of additional computing resources while providing feedback to the system providers, early software adopters often have to do significant work to utilize the software, deal with bugs, and conduct tests. They do get better user support and attention to their specific requirements, but they have to have trust in the software team and be assured that their input will be heard and that the software they are investing their time in will be "around" and supported. We argue then that there is a need for more attention to incentives for the early adopters, which could be defined as part of the research plan, which is typically developed when seeking funding. There also needs to be mechanisms to support that software that is needed by the science community, independent of its commercial viability.

Sustainability. Traditionally, sustainability plans for research outcomes often target commercialization or open-source communities, however not all teams want to commercialize their software, or there may not be a clear path to commercialization. For example, although there are examples of commercial workflow systems, the Pegasus team, which is primarily focused on research and its translation to other scientific domains, has not been interested in pursuing commercialization. We believe that this would be a distraction to the main RD effort. In practice, most of the projects sunset at the end of the grant's period of performance when no subsequent funding is found. In [5], the authors argue that commercialization and open-source models to distributed research artifacts are not sufficient to characterize translation, and that typically funding bodies do not intrinsically support translation. Therefore, we argue that there is a need for an effort to balance sustainability and translation, and to produce a plan for supporting TCS and sustainability efforts as distinct, yet necessary, campaigns.

6. Conclusion

In this paper, we presented how the Pegasus workflow management system has incorporated translational computer science in practice along the project's lifespan. We reported on our experience on building a strong, long-term engagement with the community to establish mutually beneficial relationships between the core R&D team and the user community, so that the latter can provide valuable feedback and testing capabilities at scale for the Pegasus software framework, while benefiting from better support and attention to their specific needs. Along this process, we recognized that understanding the community needs is key for driving features development, and therefore achieving TCS. We have also discussed our approach over the past two decades to address numerous challenges in performing TCS in practice. These include lack of targeted, steady funding; lack of incentives for early adopters; balancing TCS and sustainability; and maintaining a heterogeneous research and development team, e.g., GRAs and research staff and faculty focused on research and professional software developers supporting translation. Last, we have provided recommendations on how TCS could be better incorporated by funding bodies and become part of the research project's critical path. Future directions for Pegasus include continuous engagement with user community and funding bodies, and broadening the scope of our friendly users to experiment at scale with emerging applications such as artificial intelligence and particularly machine learning.

Acknowledgments. This work is funded by NSF contract #1664162, "SI2-SSI: Pegasus: Automating Compute and Data Intensive Science".

References

[1] E. Zerhouni, Translational research: moving discovery to practice, Clinical Pharmacology & Therapeutics 81 (1) (2007) 126–128. doi:10.1038/sj.clpt.6100029.

[2] J. Zoellner, L. Van Horn, P. M. Gleason, C. J. Boushey, What is translational research? concepts and applications in nutrition and dietetics, Journal of the Academy of Nutrition and Dietetics 115 (7) (2015) 1057–1071. doi:10.1016/j.jand.2015.03.010.

[3] S. H. Woolf, The meaning of translational research and why it matters, Jama 299 (2) (2008) 211–213. doi:10.1001/jama.2007.26.

[4] Umichigan translational research survey, https://research.umich.edu/sites/default/files/resource-download/translational_research_survey_2017.pdf (2017).

[5] D. Abramson, M. Parashar, Translational research in computer science, Computer 52 (9) (2019) 16–23. doi:10.1109/MC.2019.2925650.

[6] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, K. Wenger, Pegasus, a workflow management system for science automation, Future Generation Computer Systems 46 (0) (2015) 17–35. doi:10.1016/j.future.2014.10.008.

[7] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. Ferreira da Silva, G. Papadimitriou, M. Livny, The evolution of the pegasus workflow management software, Computing in Science Engineering 21 (4) (2019) 22–36. doi:10.1109/MCSE.2019.2919690.

[8] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, P. Ehrens, A. Lazzarini, R. Williams, S. Koranda, Griphyn and ligo, building a virtual data grid for gravitational wave scientists, in: 11th IEEE International Symposium on High Performance Distributed Computing, 2002, pp. 225–234. doi:10.1109/HPDC.2002.1029922.

[9] X. Collaboration, Observation of two-neutrino double electron capture in 124 xe with xenon1t, Nature 568 (7753) (2019) 532–535. doi:10.1038/s41586-019-1124-4.

[10] P. Couvares, T. Kosar, A. Roy, J. Weber, K. Wenger, Workflow management in condor, in: Workflows for e-Science, Springer, 2007, pp. 357–375. doi:10.1007/978-1-84628-757-2_22.

[11] D. Thain, T. Tannenbaum, M. Livny, Distributed computing in practice: the condor experience, Concurrency and computation: practice and experience 17 (2-4) (2005) 323–356. doi:10.1002/cpe.938.

[12] B. P. Abbott, R. Abbott, T. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. Adhikari, V. Adya, et al., Search for gravitational waves from scorpius x-1 in the first advanced ligo observing run with a hidden markov model, Physical Review D 95 (12) (2017) 122003. doi:10.1103/PhysRevD.95.122003.

[13] T. H. Jordan, S. Callaghan, R. W. Graves, F. Wang, K. R. Milner, C. A. Goulet, P. J. Maechling, et al., Cybershake models of seismic hazards in southern and central california, Seismological Research Letters 89 (2B) (2018) 875–876.

[14] M. Rynge, S. Callaghan, E. Deelman, G. Juve, G. Mehta, K. Vahi, P. J. Maechling, Enabling large-scale scientific workflows on petascale resources using mpi master/worker, in: Proceedings of the 1st Conference of the Extreme Science and Engineering Discovery Environment: Bridging from the eXtreme to the campus and beyond, 2012, pp. 1–8. doi:10.1145/2335755.2335846.

[15] G. Singh, K. Vahi, A. Ramakrishnan, G. Mehta, E. Deelman, H. Zhao, R. Sakellariou, K. Blackburn, D. Brown, S. Fairhurst, D. Meyers, G. B. Berriman, Optimizing workflow data footprint, Special issue of the Scientific Programming Journal dedicated to Dynamic Computational Workflows: Discovery, Optimisation and Scheduling (2007). doi:10.1155/2007/701609.

[16] D. M. Rubio, E. E. Schoenbaum, L. S. Lee, D. E. Schteingart, P. R. Marantz, K. E. Anderson, L. D. Platt, A. Baez, K. Esposito, Defining translational research: implications for training, Academic medicine: journal of the Association of American Medical Colleges 85 (3) (2010) 470. doi:10.1097/ACM.0b013e3181ccd618.

[17] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in

iaas clouds, Future Generation Computer Systems 48 (2015) 1–18. doi:10.1016/j.future.2015.01.004.

[18] M. Rynge, K. Vahi, E. Deelman, A. Mandal, I. Baldin, O. Bhide, R. Heiland, V. Welch, R. Hill, W. L. Poehlman, et al., Integrity protection for scientific workflow data: Motivation and initial experiences, in: Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), 2019, pp. 1–8. doi:10.1145/3332186.3332222.

[19] R. Ferreira da Silva, S. Callaghan, T. M. A. Do, G. Papadimitriou, E. Deelman, Measuring the impact of burst buffers on data-intensive scientific workflows, Future Generation Computer Systems 101 (2019) 208–220. doi:10.1016/j.future.2019.06.016.

[20] W. Chen, R. Ferreira da Silva, E. Deelman, T. Fahringer, Dynamic and fault-tolerant clustering for scientific workflows, IEEE Transactions on Cloud Computing 4 (1) (2016) 49–62. doi:10.1109/TCC.2015.2427200.

[21] G. Papadimitriou, M. Kiran, C. Wang, A. Mandal, E. Deelman, Training classifiers to identify tcp signatures inscientific workflows, in: 2019 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS), 2019, pp. 61–68. doi:10.1109/INDIS49552.2019.00012.

[22] T. M. A. Do, L. Pottier, S. Thomas, R. Ferreira da Silva, M. A. Cuendet, H. Weinstein, T. Estrada, M. Taufer, E. Deelman, A novel metric to evaluate in situ workflows, in: International Conference on Computational Science (ICCS), 2020, pp. 538–553. doi:10.1007/978-3-030-50371-0_40.

[23] R. Pordes, D. Petravick, B. Kramer, D. Olson, M. Livny, A. Roy, P. Avery, K. Blackburn, T. Wenaus, F. Würthwein, I. Foster, R. Gardner, M. Wilde, A. Blatecky, J. McGee, R. Quick, The Open Science Grid, Journal of Physics: Conference Series 78 (1) (2007) 012057. doi:10.1088/1742-6596/78/1/012057.

[24] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, et al., Xsede: accelerating scientific discovery, Computing in science & engineering 16 (5) (2014) 62–74. doi:10.1109/MCSE.2014.80.