COMPUTING edge

- Scientific Computing
- Data Visualization
- Information Technology
- Quantum Computing

JULY 2020

www.computer.org

IEEE COMPUTER SOCIETY

IEEE

COMPUTING edge

IEEE COMPUTER SOCIETY computer.org

IEEE COMPUTER SOCIETY · IEEE · Printed with inks containing soy and/or vegetable oils

Circulation: *ComputingEdge* (ISSN 2469-7087) is published monthly by the IEEE Computer Society. IEEE Headquarters, Three Park Avenue, 17th Floor, New York, NY 10016-5997; IEEE Computer Society Publications Office, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036.

Postmaster: Send address changes to *ComputingEdge*-IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Printed in USA.

JULY 2020 · VOLUME 6 · NUMBER 7

COMPUTING edge

Scientific Computing

Data Visualization

Information Technology

Quantum Computing

Departments

Magazine Roundup

The IEEE Computer Society's lineup of 12 peer-reviewed technical magazines covers cutting-edge topics ranging from software design and computer graphics to Internet computing and security, from scientific applications and machine intelligence to visualization and microchip design. Here are highlights from recent issues.

Computer

Challenges and Opportunities in the Detection of Safety-Critical Cyberphysical Attacks

Cyberphysical systems (CPSs) are increasingly used in various application domains and face the threat of cyberphysical attacks. In this article from the March 2020 issue of *Computer*, the authors discuss challenges in detecting these attacks. They use power grids and surgical robots to clarify their analysis, and they use this analysis to identify ongoing challenges and future research directions.

Computing in Science & Engineering

Exploratory Metamorphic Testing for Scientific Software

Scientific model developers are able to verify and validate their software via metamorphic testing (MT), even when the expected output of a given test case is not readily available. The tenet is to check whether certain relations hold among the expected outputs of multiple related inputs. Contemporary approaches require that the relations be defined before tests. In this article from the March/April 2020 issue of *Computing in Science & Engineering*, the authors' experience shows that it is often straightforward to first define the multiple iterations of tests for performing continuous simulations, and then keep multiple and even competing metamorphic relations open for investigating the testing-result patterns. The authors call this new approach exploratory MT, and they report their experience of applying it to detect bugs, mismatches, and constraints in automatically calibrating parameters for the United States Environmental Protection Agency's Storm Water Management Model.

IEEE Annals of the History of Computing

The Font Wars, Part 1

The Font Wars were a decades-long competition in the computer industry for dominance in font technology, viewed as a key success factor for personal computing platforms. At the heart of the Font Wars was a fundamental question: What is the best way to turn traditional printed letter forms into digital fonts for computer screens and printers? Answers to this question were researched, implemented, and launched into the marketplace, where their intense competition transformed the 500-year tradition of printing and publishing—placing the electronic literacy on the screens of billions of digital displays, computers, tablets, and smart phones around the world. Read more in this article from the January–March 2020 issue of *IEEE Annals of the History of Computing*.

IEEE Computer Graphics and Applications

Illustrating Changes in Time-Series Data with Data Video

Understanding the changes of time series is a common task in many application domains. Converting time-series data into videos helps an audience with little or no background knowledge gain insights and deep impressions. It essentially integrates data visualizations and animations to present the evolution of data expressively. However, it remains challenging to create this kind of data video. First, it is difficult to efficiently detect important changes and include them in the video sequence. Existing methods require much manual effort to explore the data and

find changes. Second, how these changes are emphasized in the videos is also worth studying. A video without emphasis will hinder an audience from noticing those important changes. This article from the March/April 2020 issue of *IEEE Computer Graphics and Applications* presents an approach that extracts and visualizes important changes of a time series.

IEEE Intelligent Systems

Research on Road Traffic Situation Awareness System Based on Image Big Data

Road traffic is an important component of the national economy and social life. Promoting intelligent and Informa ionization construction in the field of road traffic is conducive to the construction of smart cities and the formulation of macro-strategies and construction plans for urban traffic development. Aiming at the shortcomings of the current road traffic system, this article from the January/February 2020 issue of *IEEE Intelligent Systems*—on the basis of combining convolution neural networks (CNNs), situational awareness, databases, and other technologies—takes the road traffic situational awareness system as its research object and analyzes the information collection, processing, and analysis process.

IEEE Internet Computing

Container NATs and Session-Oriented Standards: Friends or Foe?

This article from the November/December 2019 issue of *IEEE Internet Computing* highlights issues that arise when deploying network address translation middle-boxes through containers. The authors focus on Docker as the container technology of choice and present a thorough analysis of its networking model, with special attention to the default bridge network driver that is used to implement network address translation functionality. They discuss some unexpected shortcomings and elaborate on the suitability of containers for deploying services based on the Interactive Connectivity Establishment standard protocol. To support their findings, they present experiments that they conducted in a real-world operational environment, namely a WebRTC service based on the Janus media server.

IEEE micro

Compute Solution for Tesla's Full Self-Driving Computer

Tesla's full self-driving (FSD) computer is the world's first purpose-built computer for the highly demanding workloads of autonomous driving. It is based on a new system-on-a-chip (SoC) that integrates industry-standard components such as CPUs, ISP, and GPU, with custom neural network accelerators. The FSD computer is capable of processing up to 2,300 frames per second, which is a 21× improvement over Tesla's previous hardware and at a lower cost. When fully utilized, it enables a new level of safety and autonomy on the road. Read more in this article from the March/April 2020 issue of *IEEE Micro*.

IEEE MultiMedia

Metric Learning-Based Multimodal Audio-Visual Emotion Recognition

People express their emotions through multiple channels, such as visual and audio ones. Consequently, automatic emotion recognition can be significantly benefited by multimodal learning. Even though each modality exhibits its unique characteristics, multimodal learning takes advantage of the complementary information of diverse modalities when measuring the same instance, resulting in enhanced understanding of emotions. Yet, their dependencies and relations are not fully exploited in audio–video emotion recognition. Furthermore, learning an effective metric through multimodality is a

crucial goal for many applications in machine learning. Therefore, in this article from the January–March 2020 issue of *IEEE MultiMedia*, the authors propose multimodal emotion recognition metric learning (MERML), learned jointly to obtain a discriminative score and a robust representation in a latent-space for both modalities. The learned metric is efficiently used through the radial basis function (RBF)-based support vector machine (SVM) kernel. The evaluation of the framework shows a significant performance, improving the state-of-the-art results on the eNTERFACE and CREMA-D datasets.

IEEE pervasive computing

CLIMB: A Pervasive Gameful Platform Promoting Child Independent Mobility

Child independent mobility (CIM) refers to the freedom and capability of children to move about their local neighborhoods without constant direct adult supervision. Our CLIMB project combats an observed decline in CIM, offering a pervasive gameful platform for home–school mobility composed of three primary components: the first two using technology to support different levels of child independence and the third providing an element of continuous motivation for positive behavior change. This article from the January–March 2020 issue of *IEEE Pervasive Computing* describes these three novel technologies: PedibusSmart, SafePath, and KidsGoGreen. It reports on four years of success with more than 1,800 elementary-age children, their teachers, and their families. The authors further show how (i) disappearing, pervasive technology contributes to successful adoption; (ii) properly balancing trust and tracking leads to useful, noninvasive technological support; and (iii) in-classroom, gameful technology engages and motivates participation, with behavior changes persisting over time.

IEEE Security & Privacy

The Need for New Antiphishing Measures against Spear-Phishing Attacks

In this article from the March/April 2020 issue of *IEEE Security & Privacy*, the authors provide extensive analysis of the unique characteristics of phishing and spear-phishing attacks, argue that spear-phishing attacks cannot be well captured by current countermeasures, identify ways forward, and analyze an advanced spear-phishing campaign targeting white-collar workers in 32 countries.

IEEE Software

Three Phases of Transforming a Project-Based IT Company into a Lean and Design-Led Digital Service Provider

Digital transformation requires a continuous review of value creation, value capture, and resourcing. In this article from the March/April 2020 issue of *IEEE Software*, the authors define a systematical service design concept to enable all stakeholders to achieve better outcomes in co-creation activities.

IT Professional

Detecting Online Content Deception

The surge of deceptive content (such as fake news) in the past few years has made content deception an important area of research. The authors of this article from the March/April 2020 issue of *IT Professional* identify two main types of content deception based on either fake content or misleading content. They present a classification of deception attacks along with delivery methods. They also discuss defense measures that can detect deception attacks. Finally, they highlight some outstanding challenges in the area of content deception.

Advancing Science with Software

Scientists increasingly use software to facilitate discoveries through simulation and analysis. Scientific software can enable research that might be impractical or impossible using experimentation, observation, or theory. Despite its importance, challenges remain in terms of testing, reliability, and reusability. This *ComputingEdge* issue presents tools for helping scientists effectively utilize software in their research.

“Metamorphic Testing: A Simple Yet Effective Approach for Testing Scientific Software,” from *Computing in Science & Engineering*, aims to help scientists who have limited training in software development employ scientific software successfully. The authors explain that metamorphic testing is a good technique for testing exploratory software with inherent uncertainties. Also from *Computing in Science & Engineering*, “SciPipe—Turning Scientific Workflows into Computer Programs” discusses a tool for automating complex scientific computations involving multiple programs.

Scientific data can be challenging to not only analyze but also convey to the public. This *ComputingEdge* issue includes two examples of creative scientific data visualizations from *IEEE Computer Graphics and Applications*. In “Weather Report: A Site-Specific Artwork Interweaving Human Experiences and Scientific Data Physicalization,” the authors describe a public art installation that educates on climate change. In “OpenSpace: Bringing NASA Missions to the Public,” the authors present a visualization program for use in interactive museum displays that helps people better understand NASA’s missions and discoveries.

Whether scientific or commercial, software should be as energy-efficient as possible. The authors of *IEEE Software*’s “A Manifesto for Energy-Aware Software” argue that software developers must consider energy consumption when designing programs. *IEEE Pervasive Computing*’s “Next Generation IoT: Toward Ubiquitous Autonomous Cost-Efficient IoT Devices” also emphasizes the need for energy efficiency, specifically in IoT devices.

This issue of *ComputingEdge* concludes with two *Computer* articles on quantum supremacy. “Powerball and Quantum Supremacy” teaches quantum concepts using a lottery metaphor. “Beyond Quantum Supremacy” identifies advances that are needed to move quantum computing forward.

This article originally appeared in Computing in Science & Engineering vol. 21, no. 1, 2019

Metamorphic Testing: A Simple Yet Effective Approach for Testing Scientific Software

Upulee Kanewala, *Montana State University*
Tsong Yueh Chen, *Swinburne University of Technology*

Testing scientific software is a difficult task due to their inherent complexity and the lack of test oracles. In addition, these software systems are usually developed by end-user developers who are not normally trained as professional software developers nor testers. These factors often lead to inadequate testing. Metamorphic testing (MT) is a simple yet effective testing technique for testing such applications. Even though MT is a wellknown technique in the software testing community, it is not very well utilized by the scientific software developers. The objective of this paper is to present MT as an effective technique for testing scientific software. To this end, we discuss why MT is an appropriate testing technique for scientists and engineers who are not primarily trained as software developers. Specifically, how it can be used to conduct systematic and effective testing on programs that do not have test oracles without requiring additional testing tools.

WHAT MAKES TESTING SCIENTIFIC SOFTWARE DIFFICULT?

Scientific software is widely used for making critical decisions in various scientific and engineering domains. For example, simulations are often used in place of physical experiments due to the time and cost constraints associated with conducting physical experiments. Furthermore, decisions made by these software systems can affect day-to-day human life, such as predictions made by climate models. Thus, it is important to make sure that these software systems are producing the correct results. Previous studies have reported many instances where scientific software systems were affected by faults such as seismic programs losing precision due to one-off errors,[1] compromised performance in coordinate measuring machines (CMMs) due to software faults,[2] and geoscience software systems producing seemingly correct yet different results that are hard to categorize as incorrect.[3] Previous works also report situations where software faults cause retractions of published work.[4] Testing is the most widely used approach for quality assurance of software. But some inherent characteristics in scientific software make it difficult to conduct systematic testing in these programs. [5]

- *Correct answers are often unknown.* Typically, scientific software is exploratory in nature and due to this, the correct results are often unknown. If the result is known there would be no need to develop the software. In such situations, only bounds or ranges of solutions might be available. Typically, in testing, an expected output is used to decide test case passing

DOI No. 10.1109/MCSE.2018.2875368
Date of publication 18 October 2018; date of current version 6 March 2019.

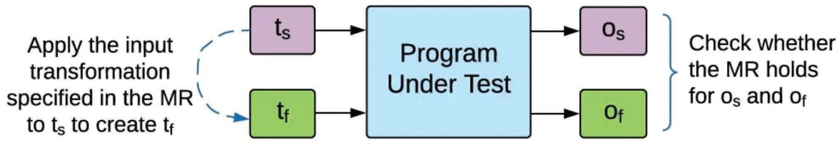

FIGURE 1. MT process. t_s: source test case, t_f: follow-up test case, o_s: output of the source test case and of: o_f: output of the follow-up test case.

or failing. This would make it challenging to conduct systematic testing in these programs.

- *Practically difficult to validate the computed output.* Scientific software often implements mathematical models that involve complex calculations. Furthermore, they tend to produce complex outputs. Both these characteristics make it hard to determine the correctness of the produced output of the software. This makes it challenging to use automated test case generation approaches such as random test generation since the output of such test cases are difficult to validate.
- *Inherent uncertainties.* Often scientific software is written to simulate models with inherent uncertainties. For some of these scientific programs, there may be more than one possible output. This makes it challenging to conduct testing on these programs.
- *Choosing suitable tolerances.* Scientific software systems often involve complex floating-point computations. Thus, specifying the acceptable tolerance for the expected output in test cases is difficult.
- *Incompatible testing tools.* Programming languages such as FORTRAN are widely used in the scientific community for developing scientific software. However, testing tools are usually developed for languages such as JAVA and C++ that are commonly used by the software engineering community. Thus, these testing tools are not effective for testing scientific programs.

WHAT IS METAMORPHIC TESTING (MT)?

Consider a program that will accept a list of real numbers and compute their average. Suppose that the input list has two million real numbers. How can we know the returned average is correct or not? Though we are not able to validate the computed average in this case, we do know some relationships between the outputs of some related inputs. For example, consider a new list of real numbers, which is a permuted list of the original list of real numbers, or which consists of four million real numbers by duplicating the original list of real numbers. For either of the new lists of real numbers, the new average is expected to be the same as the old average (subject to some round-off error tolerance). If the new and old averages are not the same, then we know that the program of computing average has bugs. This is the intuition of MT.

In software testing, passing, or failing of a test case is decided using a test oracle and it is an essential component for conducting systematic testing. MT uses metamorphic relations (MRs) to determine whether a test case has passed or failed. An MR specifies how the output of the program is expected to change when a specified change is made to the input. The following is the typical process for applying MT to a given program and Figure 1 depicts this process:[6]

1. MR identification: identifying MRs for the program under test can be done based on the specification.
2. Source test case creation and execution: commonly used test generation techniques such as random, structural coverage, or fault-based test input generation can be used. Then the generated source test cases are executed on the program under test.
3. Follow-up test case creation: use the MRs identified in Step 1 to transform the source test case to obtain the follow-up test case.
4. Follow-up test case execution: Execute the follow-up test case and compare the outputs

```
@Test
public void testMatrixMultiply() {

        //inputs for the source test case.
        //A and B can be provided by the user or can be randomly generated
        Float64Matrix A = Float64Matrix.valueOf(...);
        Float64Matrix B = Float64Matrix.valueOf(...);

        //Executing the source test case on the matrix multiplication function
        Float64Matrix Os=A.times(B);

        //creating inputs for the follow-up test case
        Random r=new Random();
        Float64Matrix B1 = Float64Matrix.valueOf(new double[][]{
       {r.nextDouble(), r.nextDouble(), r.nextDouble()},
       {r.nextDouble(), r.nextDouble(), r.nextDouble()},
       {r.nextDouble(), r.nextDouble(), r.nextDouble()}});
        Float64Matrix B2=B.minus(B1);

        //Executing the follow-up test cases
        Float64Matrix Of1=A.times(B1)
        Float64Matrix Of2=A.times(B2)

        //Checking whether the metamorphic relation holds
        //between the source and follow-up outputs.
        assertTrue(Os.equals(Of1.plus(Of2)));
}
```

FIGURE 2. A JUnit test script that uses MT approach to test a matrix multiplication function in the JScience library.

> of the source and follow-up test cases to verify whether the corresponding MRs are satisfied. Violation of an MR indicates that the program under test is faulty.

Thus, MT checks whether relationships between inputs and outputs of multiple executions were preserved during the program execution and can be used without knowing the correctness of the output for individual executions.

Consider a program P that multiplies two matrices A and B. Assume that the result of multiplying A with B is C. Matrix multiplication has the following property: $A \times B = A \times B_1 + A \times B_2$ where $B = B_1 + B_2$. We can use this property as an MR to conduct MT on P. For example, Figure 2 shows a test script written using JUnit to conduct automated testing on the matrix multiplication function in the JScience Matrix class (http://jscience.org/api/org/jscience/mathematics/vector/Matrix.html).

The source test case (which consists of A and B) can be provided by the user or can be generated randomly. For example, assume that the user provided the following simple two matrices:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \text{ and } B = \begin{bmatrix} 2 & 1 \\ 3 & 4 \end{bmatrix}.$$

This source test case is executed on the matrix multiplication function under test first. Next, based on the input relationship specified by the above MR, two follow-up test cases are created, namely, the test case consisting of A and B_1 and the test case consisting of A and B_2. Here, B_1 is randomly generated and B_2 is defined as $B - B_1$. Suppose that

$$B_1 = \begin{bmatrix} 1 & 6 \\ 3 & 5 \end{bmatrix}.$$

Then,

$$B_2 = B - B_1 = \begin{bmatrix} 1 & -5 \\ 0 & -1 \end{bmatrix}.$$

As shown in Figure 3, these two follow-up test cases are also executed on the matrix multiplication function under test. Finally, the outputs of the source test and the follow-up test cases are validated against the above MR. As shown with this test script, this MR based testing approach allows to generate follow-up test cases automatically and verify relationships

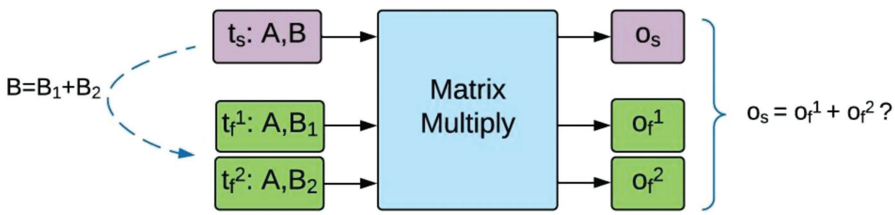

FIGURE 3. MT of a matrix multiplication program.

between multiple outputs without any manual intervention. Readers who are interested to know more about MT may consult the article by Chen *et al.*[7]

WHY USE MT FOR TESTING SCIENTIFIC SOFTWARE?

- Scientific software is often written by scientists who have the domain knowledge required to develop them. However, scientists might lack the knowledge to apply different forms of conventional testing methods. But, as evident from the example in Figure 2, MT is simple in concept, and hence could be easily learned and applied without any prior knowledge of software testing or without any software testing experience.[8]
- As shown in the example given in Figure 2, MT is easy to implement: test scripts could be easily prepared by the scientific software developers to automate the testing process or to incorporate MT into existing testing infrastructures such as JUnit. Thus, MT does not require the developers to buy or maintain additional expensive testing tools.
- As we discussed in Section "WHAT IS METAMORPHIC TESTING (MT)?", many scientific and engineering applications face the test oracle problem. This makes it challenging to conduct automated systematic testing on these programs. MT supports automated systematic testing on such programs.
- MT uses MRs to determine whether test cases pass or fail. Often scientific software is developed by domain experts who know the properties of these algorithms the best. Thus, it would be easy for them to derive MRs for testing these programs.
- Scientific software developers will be able to identify the most effective MRs and prioritize them to test their programs using their domain knowledge. For example, consider a program that computes the sine value of a given angle x. We can derive the following two MRs for testing the sine function based on its properties: MR1: sin(x′) = sin(x) where x′ = x + 360°. MR2: sin(x′) = −sin(x) where x′ = −x. Due to the constraints on the testing budget, suppose that we can conduct testing with only one MR. In such a situation, an electrical engineer will most likely choose MR1 to test her program due to the periodicity of current. But, on the other hand, a land surveyor may choose MR2 since she usually works with positive and negative angles to represent clockwise and anticlockwise measurements of angles.
- MT provides an effective way to conduct unit testing for scientific software. One of the reasons for the lack of unit testing in scientific software is the difficulty in validating the expected output of the unit for a randomly generated test input. In such situations, MT can be used to conduct automated unit testing by means of MRs.
- Scientists often conduct testing using a limited number of test cases with known outputs that they obtain from experiments or analytical solutions. MT can be used to effectively extend these limited number of test cases by deriving MRs and creating follow-up test cases according to the MRs. These follow-up test cases are most likely to execute parts of the program that might not have been executed with the original set of test cases. Thus, MT provides a way to extend existing test cases.
- Many scientific software involves elements of randomness which makes testing difficult. MT can still be applicable in such situations.

SOME EXAMPLES

Testing Epidemiological Model Implementations Using MT

Pullum *et al*. used MT to verify and validate an epidemiological model implementation.[9] Such implementations are used to model how diseases are spread in populations. Thus, it is important to verify and validate these models since they will be used to make critical decisions during a disease spread. Epidemiological model implementations face the oracle problem because these programs are written to find the answer in the first place. Therefore, developing an oracle for testing these programs is practically difficult. One of the approaches used to test these models is to compare the output of the model with data obtained from real phenomena. Obviously, such data is limited. Other approaches used to test this type of programs include comparing the output with results obtained from mathematical models and comparing the results with other simulation models. These techniques are not sufficient for conducting systematic and comprehensive testing on these programs.

The authors tested an ordinary differential equation based epidemiological model and an agent-based epidemiological model using MT. They used the data from the 1918 Influenza outbreak to calibrate the models. The authors defined 11 MRs based on making changes to various model parameters and the expected effects that those changes would have on the model output. These MRs were defined using the authors' domain knowledge about these models. Through MT, authors identified an error in the output method of the agent-based epidemiological model.

Using MT to Conduct Automated Unit Testing on a Small Angle X-Ray Scattering (SAXS) Program

We used MT to conduct automated unit testing on an open source program written to analyze small angle x-ray scattering data called SAXS.[10, 11] This program reconstructs macromolecular structures using scattering patterns obtained from experiments. This program was initially tested by running the program on a selected set of inputs where the correctness of the produced outputs was determined by domain experts. However, when we showed the domain experts the outputs generated by versions of the program injected with faults, they were unable to identify that the outputs were produced by a faulty version of the program.

Here we report the results of conducting automated unit testing on the following functions that perform several main calculations in the SAXS program:[10]

- *calculateDistance (f1)*: computes the distance between atoms;
- *findGyrationRadius (f2)*: computes the gyration radius of groups of atoms;
- *scatterSample (f3)*: main function responsible for scattering.

We used the machine learning based MR prediction approach proposed by Kanewala *et al*.[12] to predict the likely MRs for these functions. The test inputs were generated randomly. There were no violations of these predicted MRs when applied to these three functions.

To evaluate the effectiveness of MT for conducting unit testing, we created faulty versions, known as *mutants*, of these functions using the µJava (https://cs.gmu.edu/_offutt/mujava/) mutation engine. This mutation engine creates mutants of the program by making a syntactic change in the source code. With MT, we say that a mutant is killed if an MR is violated when the corresponding source and follow-up test cases are executed on that mutant. Therefore, the fault detection effectiveness of MT can be measured by the number of mutants killed during the MT process. Obviously, the higher the percentage of mutants killed, the more effective MT is in revealing bugs of a program. We use this process to evaluate the fault detection effectiveness of MT in the functions mentioned above.

Table 1 shows the percentage of mutants killed through MT for individual functions. Overall, 90% of the mutants could be killed using MT. The important thing to note here is that the entire unit testing process was fully automated starting with MR identification, source test case generation, test execution and further, did not require the domain experts to evaluate the correctness of the test outputs. Though no violations of MRs were detected for SAXS, MT helps to establish our confidence on the quality of the SAXS program.

Testing a Monte Carlo Simulation Program With MT

Ding and Hu[13] used MT for testing a Monte Carlo modeling program that simulates photon propagations in biological tissues for the purpose of accurate generation of reflectance images. The biggest challenge for testing this program is the lack of test oracles. One solution is to compare the results of the Monte Carlo simulation program to experimental results. But, as with many scientific software, building the necessary infrastructure to conduct the relevant physical experiments is time consuming and expensive. For example, in this specific case, conducting a physical experiment would require a laser beam that would produce a specific number of photons, an environment without interruptions from other light sources and good reactive imaging cameras. Thus, the authors used MT to conduct testing on this program.

The authors identified five MRs for the program based on domain knowledge and experimental results. They generated tests that cover all the branches and functions in the program. Through the violation of one of the MRs used for testing, the authors discovered faults in the program and corrected it.

They further evaluated the effectiveness of MT using mutants. The authors created 150 mutants for the Monte Carlo simulation program and they were able to detect 90% (135) of these mutants using MT.

TABLE 1. Mutants detected by predicted MRs. f_1: calculateDistance, f_2: findGyrationRadius, and f_3: scatterSample.

	f_1	f_2	f_3	Total
No. of faulty versions used	19	54	139	212
No. of faulty versions detected by MT	19	45	127	191
% of detected faulty versions	100	83	91	90

SUMMARY

Some characteristics in scientific software, such as not knowing the correct answers and inherent uncertainties in calculations, make testing them difficult. MT can be an effective testing technique to test these programs. Instead of checking the correctness of individual test outputs, MT checks whether the changes in the test outputs are according to what is expected by the program with respect to the changes in the inputs. These relationships between inputs and the expected changes in the outputs are referred to as MRs. Scientists, who typically develop this scientific software, would be in a great position to identify effective MRs because of their domain knowledge and, thus would be able to effectively test their software using MT. MT has been successfully applied for testing various scientific software including epidemiological model implementations, SAXS programs, and Monte Carlo simulations. We are strongly confident that MT is one of the most appropriate and cost-effective testing techniques for scientists and engineers.

REFERENCES

1. L. Hatton, "The T experiments: Errors in scientific software," *IEEE Comput. Sci. Eng.*, vol. 4, no. 2, pp. 27–38, Apr.–Jun. 1997.
2. A. J. Abackerli, P. H. Pereira, and N. Calônego, Jr., "A case study on testing CMM uncertainty simulation software (VCMM)," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 32, pp. 8–14, Mar. 2010.
3. L. Hatton and A. Roberts, "How accurate is scientific software?," *IEEE Trans. Softw. Eng.*, vol. 20, no. 10, pp. 785–797, Oct. 1994.
4. G. Miller, "A scientist's nightmare: Software problem leads to five retractions," *Science*, vol. 314, no. 5807, pp. 1856–1857, 2006. [Online]. Available: http://www.sciencemag.org/content/314/5807/1856.short
5. U. Kanewala and J. M. Bieman, "Testing scientific software: A systematic literature review," *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1219–1232, 2014.
6. S. Segura, G. Fraser, A. B. Sánchez, and A. R. Cortés, "A survey on metamorphic testing," *IEEE Trans. Softw. Eng.*, vol. 42, no. 9, pp. 805–824, 2016. [Online]. Available: https://doi.org/10.1109/TSE.2016.2532875
7. T. Y. Chen et al., "Metamorphic testing: A review of challenges and opportunities," *ACM Comput. Surveys*, 2017, to be published.
8. T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "An effective testing method for end-user programmers," in *Proc. First Workshop End-User Softw. Eng. (WEUSE 2005)*, 2005, pp. 21–25. [Online]. Available: http://doi.acm.org/10.1145/1082983.1083236

9. L. L. Pullum and O. Ozmen, "Early results from metamorphic testing of epidemiological models," in *Proc. ASE/IEEE Int. Conf. BioMed. Comput.*, Dec. 2012, pp. 62–67.
10. U. Kanewala, A. Lundgren, and J. M. Bieman, "Automated metamorphic testing of scientific software," J. C. Carver, N. P. Chue Hong, and G. K. Thiruvathukal, Eds. *Soft. Eng. Sci.*, Taylor & Francis, 2016, doi: https://doi.org/10.1201/9781315368924.
11. [Online]. Available: http://cgi.cs.arizona.edu/~mstrout/Projects/SAXS/software.php, 2011. Accessed on: Sep. 11, 2017.
12. U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: A machine learning approach using graph kernels," *Softw. Testing, Verification Rel.*, vol. 26, no. 3, pp. 245–269, 2016, stvr.1594. [Online]. Available at: http://dx.doi.org/10.1002/stvr.1594
13. J. Ding and X. Hu, "Application of metamorphic testing monitored by test adequacy in a Monte Carlo simulation program," *Softw. Qual. J.*, vol. 25, no. 3, pp. 841–869, 2017. [Online]. Available at: https://doi.org/10.1007/s11219-016-9337-3

UPULEE KANEWALA is an Assistant Professor with Montana State University, Bozeman, MT, USA. Her research interests include software testing, metamorphic testing, and quality assurance of scientific software. She received the Ph.D. degree in computer science from Colorado State University, Fort Collins, CO, USA, in 2015, the Master of Science degree in computer engineering from Purdue University, West Lafayette, IN, USA, in 2010, and the Bachelor of Science degree in computer engineering from University of Peradeniya, Peradeniya, Sri Lanka, in 2007. She has authored or coauthored multiple peer-reviewed articles and book chapters on metamorphic testing including the first paper on automatic detection of Metamorphic Relations. Her address is Gianforte School of Computing, 357 Barnard Hall, Montana State University, Bozeman, MT 59717. Contact her at upulee.kanewala@montana.edu.

TSONG YUEH CHEN is a Professor with Swinburne University of Technology, Melbourne, VIC, Australia. His main research interest focuses on software testing. He received the B.Sc. and M.Phil. degrees from The University of Hong Kong, Hong Kong, the M.Sc. degree from University of London, London, U.K., the DIC degree from the Imperial College, London, U.K., and the Ph.D. degree from The University of Melbourne, Melbourne, VIC, Australia. Prior to joining Swinburne, he taught at The University of Hong Kong and The University of Melbourne. He is the Inventor of metamorphic testing and adaptive random testing. His contact address is: Department of Computer Science and Software Engineering, Swinburne University of Technology, VIC 3122, Australia. Contact him at tychen@swin.edu.au.

This article originally appeared in
computing in SCIENCE & ENGINEERING
vol. 21, no. 3, 2019

SciPipe—Turning Scientific Workflows into Computer Programs

Samuel Lampa, Martin Dahlö, Jonathan Alvarsson, and Ola Spjuth, *Uppsala University*

INTRODUCTION

Scientific Workflows are becoming increasingly popular as a way to automate complex scientific computations consisting of multiple programs.

One of the main motivations behind this development is increased robustness and reproducibility of computational analyses. Chaining together multiple programs using plain scripts, as is often the first step in automating a pipeline, can easily become fragile and error prone due to the manual management of file paths and program invocations. Also, plain scripts are not optimal if for some reason you have to cancel a run and try to restart it from any partially finished steps. It can be hard to know which output files are properly finished and which are truncated from the cancelled run. Last but not least, plain scripts do not by default save an execution trace of what was run, such that the full procedure used to create a specific output file can be clearly presented. These are all aspects that scientific workflows are designed to help with.

Despite many hundreds of scientific workflow tools published over the years, there can still be significant challenges when trying to use many of them.

One reason for this is that many workflow tools have been designed with a very narrow use case in mind, often building in assumptions unique to the specific problem domain aimed at, which might make them less applicable for scientific pipeline needs in general.

Even among the large numbers of general workflow tools, surprisingly many contain various constraints and assumptions limiting their generality. Often, this might not be obvious before trying to apply them to complex tasks.

At the Department of Pharmaceutical Biosciences, we have spent the last few years using workflow tools to automate machine learning pipelines for predictive toxicology among other things. We have reviewed the top dozen workflow tools popular in our field of bioinformatics. We even tried out Luigi, created by music company Spotify, which is popular in industry and far from a bioinformatics-aimed tool.

A recurring theme has been how often tools contain various limits that make them hard to use for complex use cases. Machine learning workflows in particular often lead to highly complex workflows because of their common inclusion of cross validation and parameter sweeps from hyperparameter optimization. Not only are these workflows complex, but they also show some characteristics not always common in other domains: The need for dynamic scheduling. That is, they need to be able to parametrize and start tasks based on information obtained during the workflow run. Somewhat surprisingly, this is a problem in a majority of workflow tools because of how common it is that they have a strict separation between the scheduling and execution phases of the workflow run. That is, after a workflow has progressed into its execution phase, it is commonly not possible to schedule and start new tasks with parameter values obtained in the current run. At least not without initiating completely new workflow runs.

Anyway, after a lot of evaluation, it seemed at the time that Luigi[1] was the most promising way forward for us. Later, we learned that Luigi's functional programming inspired API design was not quite fit for our needs of dynamic workflow rewiring, which is one of the reasons why we later chose to develop the SciPipe

DOI No. 10.1109/MCSE.2019.2907814
Date of current version 26 April 2019.

library (more on that later), but there is an interesting story to tell about our Luigi phase too.

Workflows as Computer Programs

It turned out that because Luigi was implemented as a programming library, it was flexible enough that we could build an alternative API on top of it, which resulted in the SciLuigi helper library. [2] Specifically, SciLuigi enabled us to keep the dependency network definition separate from task definitions—a core principle in flow-based programming, as we will explain shortly—which makes it much easier to reconnect workflows without changing internals of workflow components in complex ways.

This positive experience from a programming API-based workflow tool helped push a realization that has grown over a number of years of discussions and experimenting: Workflows are in the end just a glorified version of computer programs.

It turns out that although many use cases can be modeled as simple linear sequences of program invocations depending on each other, not all cases are that simple. There are many examples where the need for logic to control the workflow structure is so complex that any attempt at modeling it with a declarative workflow language ends up implementing what is already available in existing programming languages.

This tendency can also be seen in popular workflow engines building on domain specific languages (DSLs). Tools that become popular often either have a really flexible DSL from the start, e.g., because the DSL is implemented in an existing scripting language (e.g., Nextflow, [3] building on the Groovy language), or their DSLs are step by step becoming increasingly complex, and in the end approaching computer programming languages in their capability (e.g., Cuneiform,[4] implementing a powerful functional language [5]).

SciPipe

The above lesson is something we were taking into account when we set out to design the SciPipe workflow tool from scratch after finding out about several limitations with the Luigi/SciLuigi setup we were already using (primarily the need for dynamic scheduling and lack of compile time warnings about errors in workflow connectivity).

SciPipe[7] is designed from the start as a programming library embedded in the implementation language (Google's Go, or "Golang"), rather than inventing new textual syntax or graphical tools. It thus leverages the full power and flexibility of the Go programming language for implementing workflow logic. So far, we have not encountered a workflow use case that we have not been able to model with this approach. Even complex machine learning workflows with nested branching has been solvable, as exemplified in a recent paper by the authors.[8] Another nice side effect of the Go language in particular is that it compiles to self-contained executable files, which makes deployment of most Go programs very straightforward. SciPipe is open source software (MIT licensed). A simple "Hello World" style workflow example is shown in Figure 2. For more information, source code and documentation, see the work of Lampa *et al.* [6], [9]

Flow-Based Programming Focuses on Data Flow

Now, there *are* some differences between most workflow programs and most normal programs. The main difference can be seen in the strong focus in workflow programs on *data flow*. When defining how multiple programs depend on each other we are in effect defining how the data will flow through these programs. There is actually a looming risk for workflow tool developers to miss this detail and model the workflow dependency graph as just dependencies between programs and not their inputs and outputs. This can quickly lead to problems because one program typically do not depend on just one other program but rather specific outputs of possibly multiple upstream programs. That is, data needs to be a first class citizen when defining workflows, or we risk missing important details that will otherwise be buried in less thought-out *ad hoc* code.

One paradigm that takes note of this fact is flow-based programming (FBP).[10] Invented at IBM in the late 1960s and used on large mainframe computers at banks and other large institutions, the flow-based programming paradigm has seen some resurge in popularity in recent years, possibly driven by the recent trends toward multicore CPUs, distributed computing, and message-oriented architectures.

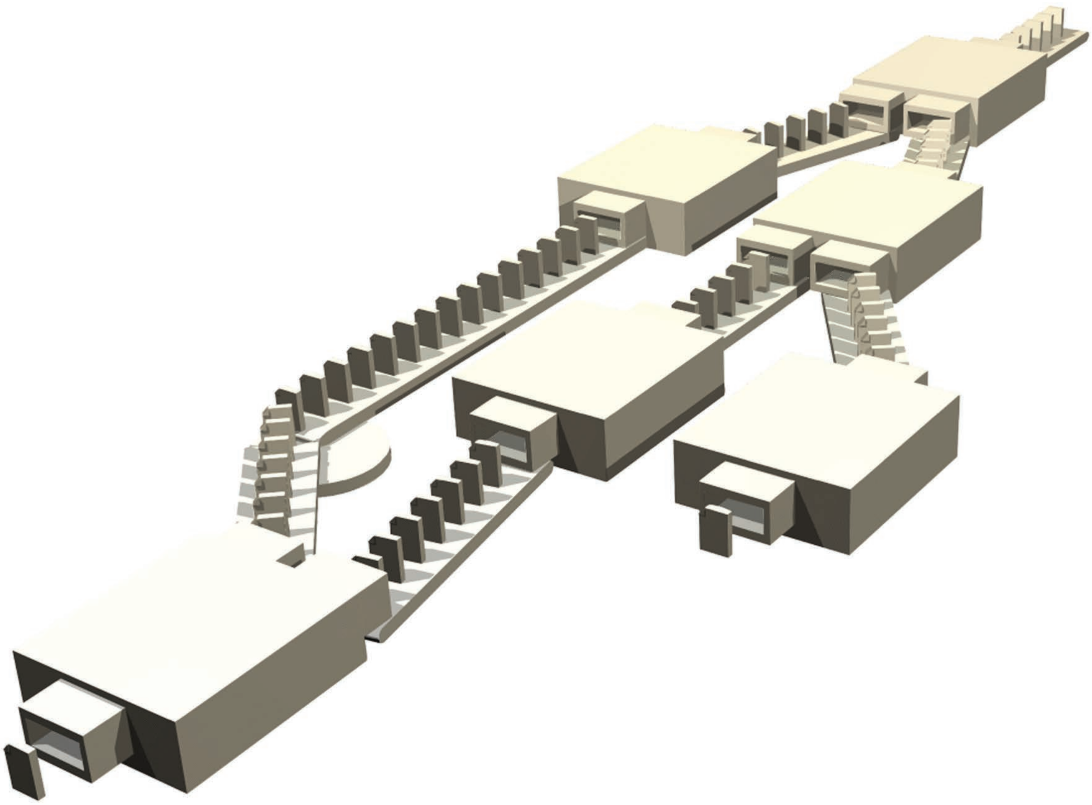

FIGURE 1. Flow-based programs can be likened to a factory with processing stations connected with conveyor belts, upon which data items "flow" through the network of stations and conveyor belts.

Flow-based programming ordains a number of design principles. The most important one in the context of dependency definition though is that it models dependencies between processes in an appropriate level of detail; in terms of data inputs and outputs. The data itself are modeled through so-called "information packets" and inputs and outputs as "ports"—a kind of pluggable component between which yet another concept can be connected; "channels." Channels have bounded buffers and act as a kind of conveyor belt between processes, letting processes work on information packets from its in-ports asynchronously and sending them on their out-ports (mostly) independently from the processing rate of other processes. Figure 1 tries to depict this in an artistic way.

The core idea of flow-based programming is how it draws all of these things together into a declarative data flow definition *separate* from the process implementations. This allows easy rewiring of the data flow without changing a single line of process implementations. It thus allows us to create libraries of reusable components which can be plugged in at any place in the program network, as long as its in- and out-ports are compatible with the in- and out-ports they connect to.

Note that while FBP is often associated with visual programming, that is far from a requirement. In our experience, skipping the visual part and focusing on a simple programming API has more than fulfilled our needs, while letting us avoid depending on the complexity of a visual programming framework. The fact that the Go language provides the most important pieces for this to work (independently running go-routines and channels with bounded buffers) certainly helps.

Reproducibility in Workflow Programs

We have presented some rationale for writing workflows as computer programs, but what about the other aspects important for workflows, such as

```go
package main

import (
    // Import the SciPipe library
    "github.com/scipipe/scipipe"
)

func main() {
    // Initialize a workflow with max 4 concurrent tasks
    wf := scipipe.NewWorkflow("hello_world", 4)

    // Initialize processes, and file extensions
    hello := wf.NewProc("hello", "echo 'Hello ' > {o:out|.txt}")
    world := wf.NewProc("world", "echo $(cat {i:in}) World > {o:out|.txt}")

    // Define the data flow from out-port to in-port
    // on the two processes in the workflow
    world.In("in").From(hello.Out("out"))

    // Run the workflow
    wf.Run()
}
```

FIGURE 2. Simple example workflow implemented with SciPipe. The workflow consists of two processes: "hello" and "world," where "hello" writes the word "Hello" to a file, which the "world" process uses as input and appends the word "World," before writing the combined output to a new file. Processes are defined from shell commands, where placeholders on the form of {i:inport|file-extension} and {o:outport|file-extension} are used to define in-ports and out-ports. These placeholders act as templates that will be replaced with specific file names during the workflow execution, based on the files that are sent on the processes' in-ports. Although this is a very simple example, Go programmers will recognize that this workflow definition is also a simple Go program. More detailed examples are provided on the main SciPipe website.[6]

reproducibility? Are not there areas where the traditional ways of designing workflows still have its merits?

In terms of reproducibility, we argue that maximizing the declarative nature of the workflow definition is not necessarily the key requirement, although that has been a prioritized focus area in many workflow frameworks. Rather, we think what is most important is to be able to produce an unambiguous record of how each data output from a workflow was produced.

In SciPipe we think we have found a simple yet powerful way to capture provenance, leveraging its low complexity and flow-based programming-based design.

The fact that programs are defined, even programmatically, as a graph of processes with dataflow connections on which data items "flow," made it straightforward to implement a provenance mechanism around the information packets themselves, letting them aggregate information about the processing nodes in the graph as they pass through them. Since one data item might be the merger of two initially separate dataflow paths, this is done in a hierarchical way. This way of tracing the execution also means that the provenance will be captured at the granularity that matters; at the data level. In concrete terms, it means SciPipe can—and does—save a provenance record for every output file of the workflow.

Saving a provenance record only for the full workflow run risks introducing a distance between the data and the workflow run that over time will only grow bigger. By instead saving the provenance information in an accompanying file associated with every output file of the workflow, as is done in SciPipe, means it is far easier to ensure this information stays associated.

CONCLUSION

By providing increased robustness to failure and reproducibility of results, workflows are an established workhorse of computer-aided science as well

as an increasingly important part also in data pipelines in industry.

Dominant approaches to workflow authoring, such as specialized domain specific languages or graphical workflow authoring tools, do not always cater to the full spectrum of workflow needs, such as machine learning workflows consisting of parameter sweeps nested with cross validation and dynamic scheduling.

For such complex and dynamic needs, we argue for the benefits of workflows implemented as computer programs, using workflow tools implemented as programming libraries. Furthermore, by using flow-based programming principles, a declarative while programmatic API can be presented, while still hiding a lot of low-level details under the hood.

Being able to provide workflows in the form of compiled executable files has benefits both for the ease of deployment and future reproducibility. Workflow tool implementations are at risk for bit rot or incompatible changes well before the mainstream programming languages—or processor architectures—are.

For more information on SciPipe, check out the main website at scipipe.org.

REFERENCES

1. E. Bernhardsson, E. Freider, and A. Rouhani. spotify/luigi - GitHub. Accessed: March 23, 2019. [Online]. Available: https://github.com/spotify/luigi
2. S. Lampa, J. Alvarsson, and O. Spjuth, "Towards agile large-scale predictive modelling in drug discovery with flow-based programming design principles," *J. Cheminform.*, vol. 8, no. 1, p. 67, 2016.
3. P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotech*, vol. 35, no. 4, pp. 316–319, 2017.
4. J. Brandt, M. Bux, and U. Leser, "Cuneiform: A functional language for large scale scientific data analysis," in *Proc. EDBT/ICDT Workshops*, 2015, pp. 7–16.
5. J. Brandt, W. Reisig, and U. Leser, "Computation semantics of the functional scientific workflow language cuneiform," *J. Functional Program.*, vol. 27, p. e22, 2017.
6. S. Lampa, SciPipe website. Accessed: Mar. 23, 2019. [Online]. Available: http://scipipe.org
7. S. Lampa, M. Dahlö, J. Alvarsson, and O. Spjuth, "Scipipe—A workflow library for agile development of complex and dynamic bioinformatics pipelines," Forthcoming, GigaScience, 2019.
8. S. Lampa, J. Alvarsson, S. Arvidsson Mc Shane, A. Berg, E. Ahlberg, and O. Spjuth, "Predicting off-target binding profiles with confidence using conformal prediction," *Front. Pharmacol.*, vol. 9, 2018, Art. no. 1256.
9. S. Lampa, M. Czygan, and J. Alvarsson, SciPipe source code repository at GitHub. Accessed: Mar. 23, 2019. [Online]. Available: https://github.com/scipipe/scipipe
10. J. P. Morrison, *Flow-Based Programming: A New Approach to Application Development*, 2nd ed. Charleston: Self-published via CreateSpace, May 2010.

SAMUEL LAMPA received the M.Sc. degree in molecular biotechnology engineering and the Ph.D. degree in pharmaceutical bioinformatics from the Department of Pharmaceutical Biosciences, Uppsala University, Uppsala, Sweden, in September 2018. His research interests span intelligent data processing systems, reproducibility, artificial intelligence, and biomimetics in information technology, among other things. He blogs at bionics.it. Contact him at samuel.lampa@farmbio.uu.se.

MARTIN DAHLÖ is currently working toward the Ph.D. degree in bioinformatics at the Department of Pharmaceutical Biosciences. He received the Master of Science degree in bioinformatics engineering from Uppsala University, Uppsala, Sweden, in 2010, and has worked at Uppsala University since then. More and more of the work has involved working closely with the UPPMAX HPC Center with a focus on user efficiency. Contact him at martin.dahlo@scilifelab.uu.se.

JONATHAN ALVARSSON is a researcher in the field of Pharmaceutical Bioinformatics, Department of Pharmaceutical Biosciences, Uppsala University, where he also received the PhD degree in 2015. His main research interest has been in machine learning and molecular fingerprints as well as the development of scientific software.

OLA SPJUTH is an associate professor at the Department of Pharmaceutical Biosciences, Uppsala University. His main research interests include intelligent systems for drug discovery and chemical safety, including how automated high-throughput and high-content molecular and cell profiling technologies coupled with AI and predictive modeling can enable study of complex phenomena in pharmacology, toxicology, and metabolism.

EDITORS: Bruce Campbell, bcampbel01@risd.edu
Francesca Samsel, figs@cat.utexas.edu

DEPARTMENT: ART ON GRAPHICS

This article originally appeared in IEEE Computer Graphics and Applications vol. 38, no. 4, 2018

Weather Report

A Site-Specific Artwork Interweaving Human Experiences and Scientific Data Physicalization

Daniel F. Keefe and Seth Johnson, *University of Minnesota*
Ross Altheimer, *TENxTEN Studio*
Deuk-Geun Hong, *University of Minnesota*
Robert Hunter, *James Dayton Design, Ltd.*
Andrea J. Johnson, *University of Minnesota*
Maura Rockcastle, *TENxTEN Studio*
Marc Swackhamer, *University of Minnesota*
Aaron Wittkamper, *Wittkamper / Reiff*

Weather Report is a site-specific art installation that entices visitors to examine climate change at a human scale, both physically and metaphorically. Weather data are displayed using the balloons as physical pixels that can be touched, part of an effort to make objective, scientific data graspable by non-scientists. Visitors contrast these objective weather data with their own data – weather-related memories they enter at a kiosk to create a subjective weather record from the Twin Cities community.

The historic Stone Arch Bridge, once the railroad gateway to the city of Minneapolis, crosses the Mississippi River at St. Anthony Falls. Walking across this gently curving path on a warm June night, we hear the sounds of the Northern Spark art festival. Held each year in the Twin Cities, Northern Spark has grown to attract tens of thousands of people who view performances, explore temporary installations in the streets and along the riverfront, and gather for one night to experience art as a community until the sun rises.

Looking down from the bridge on this night in 2016, colored lights flicker and reflect off the water, as a stream of local residents and visitors wind down from the bridge, through Mill Ruins Park, and along the walking path to what people are describing as the "balloon tunnel." More than 800 miniature weather balloons are suspended to form two walls that undulate, like air-filled sheets in the night wind. Both the balloons and the steady stream of visitors perform an animated dance as the Mississippi River flows by on a parallel course (see Figure 1).

Our design collective, MINN_LAB, composed of architects, landscape architects, and computer scientists, created *Weather Report* in response to theme of "Climate Chaos | Climate Rising."[1] We identified common threads around the experience we wished to create: *making climate personal* and *connecting objective scientific data to subjective human experiences*. Then, over months of interdisciplinary design and discussion, we developed, interwove, and revised these threads drawing upon our plurality of design voices and different technical research interests.

CONCEPTUAL FRAMEWORK AND EARLY DESIGN

Weather Report uses local, human experiences with weather as an entry point for discussing the difference between objective weather data and subjective interpretations and memories. Although the data are on the human timescale (45 years of objective data and memories), the objective-subjective comparisons the piece asks visitors to make speak to the broader question of how the earth's climate has shifted over a much larger period of time, how this is measured and interpreted today, and how rigorous scientific processes differ from everyday discussions of weather.

FIGURE 1. "Balloon pixels" create an illuminated physical manifestation of 4.5 decades of local weather data, installed along the Mississippi River during the Northern Spark arts festival. Photo credit: Authors, 2016 (Used with permission.)

Scientists often struggle to explain the objective basis for climate change, what it means to the average citizen, and the grand timescale on which it operates. In contrast, our friends and family have no trouble at all explaining everyday human experiences with weather and extrapolating from these: *On the day you were born, there was an amazing blizzard, it took me four hours to shovel the car out of the driveway; that used to happen all the time, but we don't have storms like that anymore.* Or, just as common, *we have never had a winter with so many violent storms in a row; grandpa had to buy a generator because the power kept going off; this year is the worst ever. Weather Report* asks visitors, which of these weather memories is true? How do human experiences and memories compare with objective data? How do human timescale data points relate to the much larger climate timescale, and how do scientists objectively measure those data?

Getting Physical with Balloon Pixels

Inspired by the role weather balloons play in data collection, we came to view the balloon as a simple, physical, relatable manifestation of the scientific process. Each weather balloon provides a small data-driven contribution to the larger picture of the science. Reinterpreting this in the context of an experiential display of data, we reasoned that balloons could function as physical "pixels,", changing appearance in response to individual data readings and collectively presenting a broad picture of the scientific data.

The balloon-as-pixel concept could have many interpretations, and this fueled a rich, several month-long period of sketching and ideation within the team. Drawing upon the architectural tradition, the site for the installation was critical in the design. One of the prominent features of Mill Ruins Park is a walking path that follows the river. Thus, we were

FIGURE 2. A tunnel of balloon pixels along the walking path in Mill Ruins Park, Minneapolis, MN. Photo credit: Authors, 2016 (Used with permission.)

inspired to create a walk-though experience—something viewers could experience and touch as they traversed the path.

The method for illuminating or otherwise adjusting the balloon pixels in response to data also required design. We experimented with balloons on strings with motors and internal LEDs but found that the best visual results could be achieved by projecting colored light onto the balloons. Light is transmitted through the balloon, creating the effect of a glowing orb.

Two Walls for Contrasting Views of Weather

The final design arranges the balloons into two walls, each constructed from a 12x36 grid of balloon pixels, forming a tunnel around the walking path (See Figure 2). Within the park, the tunnel is positioned so as to be visible at a distance from the Stone Arch Bridge or the hillside looking down to the river and so that light from the illuminated balloons reflects off the water at night.

Conceptually, the mirrored walls create a strong physical basis for visual and body-centric comparisons of the data. Viewers can stand inside the tunnel and point with one hand to a balloon that depicts objective temperature data for a particular hour, day, month, year, or decade and then point directly across the tunnel with the other hand to find the corresponding data point in the subjective record.

ILLUMINATED VISUALS AND USER INTERFACE

To stay true to the science, we established a goal to make sure that the data-to-visual mapping could be considered accurate. However, we also recognized that, unlike many data visualizations, the primary goal of this site-specific artwork is not facilitating analysis using specific data points but rather creating an experience—a feeling of being aware of and immersed within the data. This creates the catalyst for discussion that is the real goal of the piece.

Mapping Temperature Data to Balloon Walls

The objective local weather data were recorded at the US Weather Station KMSP located at the Minneapolis–St. Paul International Airport (retrieved via the Wolfram Alpha computational engine). The dataset contains nearly 4.5 decades of hourly readings for temperature, wind speed, and precipitation (rain and snow) from 12 a.m. on 1 January 1960 to 7 a.m. on 11 June 2016.

All of the multi-decade, multi-variable data are displayed (over time) using only the 12x36 pixels available for each wall. To accomplish this, the balloon pixels are organized into a hierarchical time grid, where each balloon represents a time window, as illustrated in Figure 3. The base color for each balloon is set by applying a warm-to-cool color map to the average temperature within the balloon's time window.

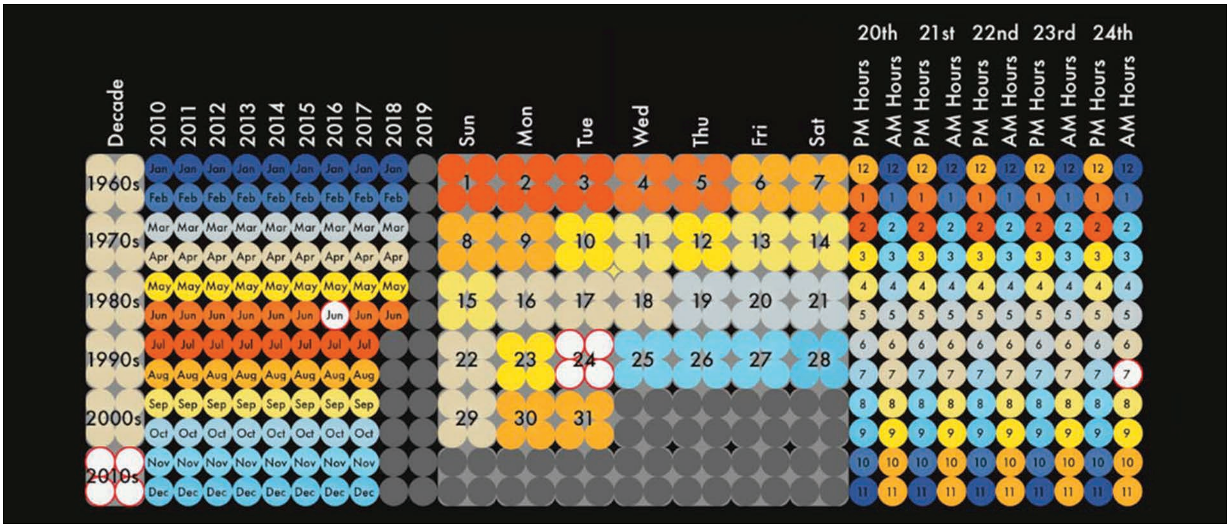

FIGURE 3. The data-to-visual mapping for each wall uses a hierarchical arrangement for time, decades are displayed in the two leftmost columns, with the current decade highlighted, followed by months of the year with the current month highlighted, followed by days of the month, and so on. 7 a.m. on Tuesday, 24 June 2016 is highlighted in this figure.

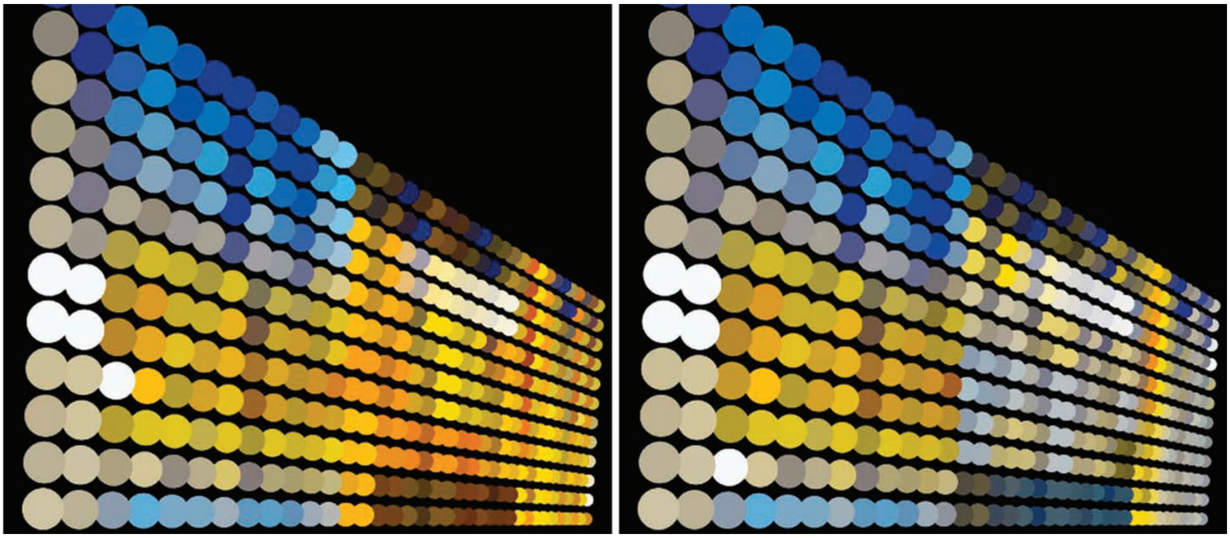

FIGURE 4. Frames from an animated blue rain effect superimposed over the objective weather wall.

The animated visualization updates at a rate of one historical hour each half-second of real-world clock time with the "current time" signified by a white highlight. This means that the white highlight for the current hour in the far right column moves quickly from the top to the bottom of the column every six seconds. When it reaches the end, the hourly-data columns each shift to the right by one, paging a new column of twelve hours into the display. Likewise, when the current time reaches the end of the last day in a month, a new month of data is paged into the display.

Animated Effects for Secondary Data Variables

Secondary data variables (rain, snow, wind speed, and cloud cover) are also included, but these are treated as discrete weather "events". When the current time reaches an hour that includes one or more weather events, a three-second animated weather effect is added to a queue. These effects are applied to the entire display as a semi-transparent overlay, using all of the pixels as illustrated in Figure 4. For a rain event, blue pixels stream down the wall. Snow events create

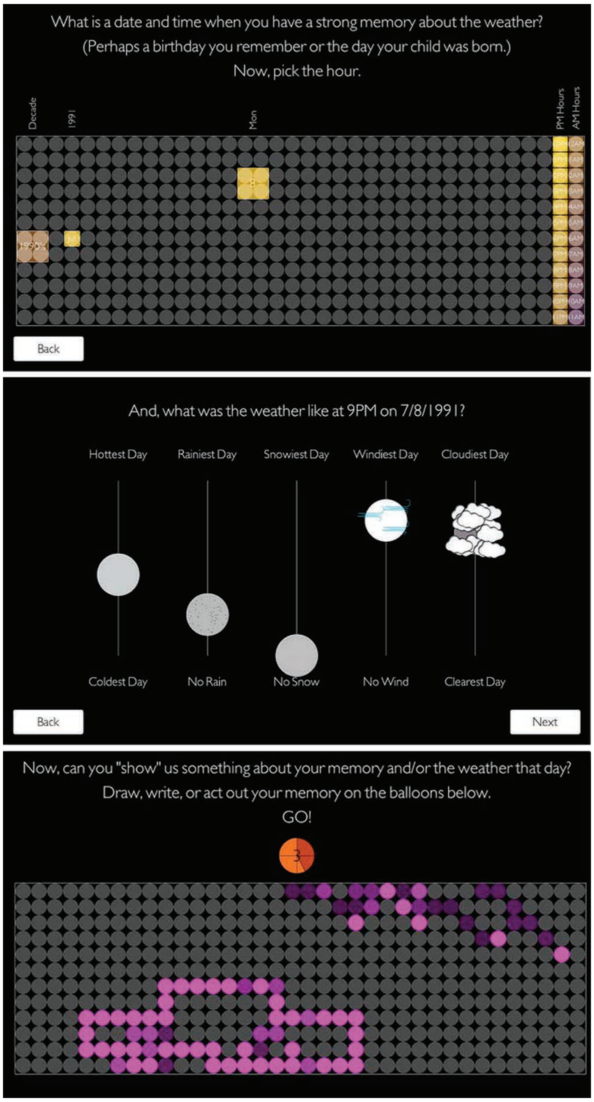

FIGURE 5. Visitors enter weather memories using a three-stage, multi-touch interface.

slower, gentler white pixels wafting down the wall. Cloud-cover events tint the top of the wall light gray, and wind events cause gray particles to fly across the wall from one side to the other.

The Subjective Weather Record

The subjective dataset contains the same variables but is much sparser. Over the course of the night, visitors fill in this subjective weather record, and the visualization interpolates between each data point to fill in the gaps.

The kiosk for entering weather-related memories uses a multi-touch interface. The specific hour for the memory is entered by selecting the decade, then year, then month, then day, and finally hour by touching the corresponding balloons on a diagram (Figure 5, top)—in this way, the data entry process teaches visitors about the data mapping used on the walls. Then, the temperature, rain, snow, wind speed, and cloud cover are entered using sliders with extremes labeled in relative, not numeric terms (e.g., the hottest day vs. the coldest day) (Figure 5, middle). Finally, visitors act out their weather memory, using the multi-touch screen to create an animation (Figure 5, bottom).

Similar to the animated weather effects, these weather memory animations are replayed as semi-transparent overlays immediately after entering the memory and again whenever the animation reaches the time associated with the memory.

Projecting on Balloons: Some Technical Notes

The balloons are lit from outside the tunnel by four short-throw projectors and a custom graphics program with five coordinated output windows. Calibration is critical because the site, next to the river and a hill, requires projecting from extreme angles. The projected graphics are created by first rendering a regular grid of 2D colored circles to an off-screen buffer. Then, this temporary image is texture mapped onto a deformable four-sided polygon. To calibrate each projector, a keyboard user interface is used to interactively adjust the coordinates of the corners of the polygon, skewing it until the colored circles project onto the correct balloons.

THE EXPERIENCE

Thousands of visitors experienced Weather Report at Northern Spark (see Figure 6). People walked through the tunnel and around the sides. They touched the balloons as they walked. More than 200 people contributed memories to the subjective weather record. The earliest entry was for 13 Dec. 1961, and there were 22 entries for the Halloween Blizzard of 1991. It would have been ideal to include additional stations or mobile devices for entering subjective data, as there were often long lines at the kiosk. Visitors who had a chance to act out their weather memories using their hands and then see the result displayed on the balloons said they felt as if they were playing a 36-foot-long instrument.

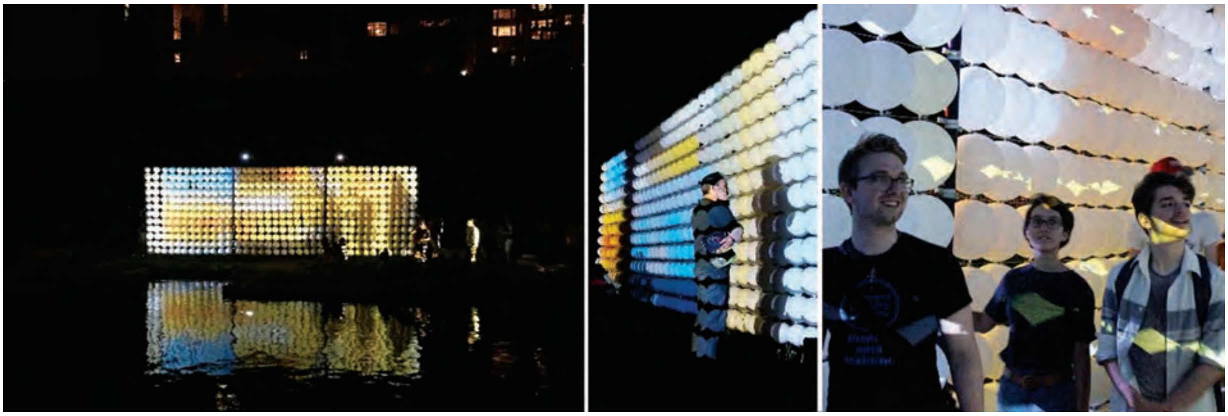

FIGURE 6. *Weather Report* at Northern Spark 2016. Photo credits: Authors (left, middle), Krista McCullough (right), 2016. (Used with permission.)

The biggest surprise was in how viewers reacted to the projection. As designers, we treated the projection as simply a behind-the-curtain technology – the technology that just happened to provide the best method of illuminating the balloons. To our surprise, the projection beams became interactive play areas, places for visitors to dance, pose for pictures with data covering their bodies, and cast shadows on the data that could be seen from across the river and around the festival. Visually, this added another layer of human connection and embodied movement to the data-driven visuals.

CONCLUSIONS

Interdisciplinary collaborative design processes are both rewarding and challenging. Our experience and results reflect months of discussion, much of it devoted to learning to speak the language of each other's disciplines. The whole team had to work within the constraints of an outdoor, dusk-to-dawn festival and a specific site. Each discipline had to stretch to accommodate the interests and expertise of the other. This process led to a unique result, one that no individual on the team would have created. It also led to new thinking that broadens and refuels our primary disciplines.

This is one of the powerful recurrent themes in the work highlighted in this *Art on Graphics* column. In our case, we will surely continue to collaborate as an interdisciplinary collective, but even as we return to our primary disciplines we bring new knowledge with us that will impact our future work. Architects have learned to become a little bit more like computer scientists and now have a foundation for further explorations of data and science in the built environment, described in more detail in a companion paper within the architecture community.[2] Computer scientists have learned to become a little bit more like architects and now have a foundation for further explorations of embodied experiences and physicality in data visualization.

Data visualization researchers will connect this work to the emerging theme of "data physicalization."[3] Although *Weather Report* includes digital components, the resulting piece is a decidedly physical, real-world, data-driven experience. This is a topic where the computer science community has much to learn from architects, designers, and artists. Collaborations like this one make it possible to explore concepts of data in physical space at a scale and connecting with the local community in ways that are simply impossible in a traditional computer science context.

ACKNOWLEDGMENTS

This work was supported in part by a grant from the University of Minnesota Digital Design Center.

REFERENCES

1. "Northern Spark Call For Proposals: Climate Chaos | Climate Rising," *Northern Lights.mn*, 2016; http://www.mnartists.org/content/northern-spark-call-proposals-climate-chaos-climate-rising.
2. M. Swackhamer et al., " Weather Report: Structuring Data Experience In The Built Environment," *Proceedings Of Architectural Research Centers Consortium*, 2017, pp. 102–111.
3. Y. Jansen et al., " Opportunities And Challenges For

Data Physicalization," *Proceedings Of The 33rd Annual ACM Conference On Human Factors In Computing Systems* (CHI 15), 2015, pp. 3227–3236.

DANIEL F. KEEFE is an associate professor in the Department of Computer Science & Engineering at the University of Minnesota. Contact him at dfk@umn.edu.

SETH JOHNSON is a PhD student in the Department of Computer Science & Engineering at the University of Minnesota. Contact him at joh08230@umn.edu.

ROSS ALTHEIMER is a principal and cofounder at TENxTEN Studio. Contact him at ross@tenxtenstudio.com.

DEUKGEUN HONG is an architectural designer at MSR (Meyer, Scherer & Rockcastle, Ltd.) Design. Contact him at deukgeun@msrdesign.com.

ROBERT HUNTER is a senior associate at James Dayton Design, Ltd. Contact him at rhunter@jddltd.com.

ANDREA J. JOHNSON is associate director of MS Research Practices in the School of Architecture at the University of Minnesota and principal of Andrea J. Johnson Architecture. Contact her at andreajj@umn.edu.

MAURA ROCKCASTLE is a principal and co-founder at TENxTEN Studio. Contact her at maura@tenxtenstudio.com.

MARC SWACKHAMER is a professor and head of school at the School of Architecture at the University of Minnesota. Contact him at swack@umn.edu.

AARON WITTKAMPER is founder and design lead at Wittkamper Studio and cofounder at Wittkamper / Reiff. Contact him at aaron@wittkamper.studio.

Contact department editor Bruce Campbell at bcampbel01@risd.edu or department editor Francesca Samsel at figs@cat.utexas.edu.

EDITOR: Theresa-Marie Rhyne, theresamarierhyne@gmail.com

DEPARTMENT: VISUALIZATION VIEWPOINTS

This article originally appeared in
IEEE Computer Graphics and Applications
vol. 38, no. 5, 2018

OpenSpace: Bringing NASA Missions to the Public

Alexander Bock, *New York University*
Charles Hansen, *University of Utah*
Anders Ynnerman, *Linköping University*

This viewpoint presents OpenSpace, an open-source astrovisualization software project designed to bridge the gap between scientific discoveries and their public dissemination. A wealth of data exists for space missions from NASA and other sources. OpenSpace brings together this data and combines it in a range of immersive settings. Through non-linear storytelling and guided exploration, interactive immersive experiences help the public to engage with advanced space mission data and models, and thus be better informed and educated about NASA missions, the solar system and outer space. We demonstrate this capability by exploring the OSIRIS-Rex mission.

The endeavors of space missions have typically been communicated and demonstrated through produced movies or animations. NASA missions, such as OSIRIS-REx, are the products of years of planning and scientific research. Explaining missions to the public in an immersive setting is challenging and has only previously been possible with produced shows for planetariums or interactively with limited mission details and external data.[1,2] Our goal is to provide an interactive experience in which the public can see and experience space missions to better understand the science, the benefit to mankind, and the challenges of deep-space missions.

To achieve this, we developed the open source interactive data visualization software OpenSpace[3] to visualize the entire known universe and portray our ongoing efforts to investigate the cosmos through large-scale, contextualized, multimodal astrovisualization. OpenSpace incorporates the latest techniques from visualization research and supports interactive presentation of dynamic data from observations, simulations, and space mission planning and operations.[4–6] OpenSpace is capable of providing immersive experiences to multiple observers by leveraging the projection capabilities of modern planetariums.[7] These immersive experiences will help better inform and educate the general public by allowing interactive exploration of NASA missions. OpenSpace has been developed with partners from the American Museum of Natural History, Linköping University, the Scientific Computing and Imaging Institute at the University of Utah, and New York University's Tandon School of Engineering.

SPICE FILES AND NASA MISSIONS

OpenSpace relies on a continuous coordinate system that enables developers to position and orient data with extreme accuracy. NASA's Spacecraft Planet Instrument C-matrix Events (SPICE) observation geometry system for planetary science missions, for example, provides the ability to display mission planning, data acquisition, and post-mission data analysis using high-resolution data files called *kernels*. Each mission has a different set of SPICE kernels that

describe the spatial and temporal aspects of the mission and its components. For example, in collaboration with the Applied Physics Laboratory, the OpenSpace team worked to visualize the Navigation and Ancillary Information Facility's SPICE-specified instrument targeting during New Horizons' fly-by of Pluto.[4] SPICE's guided visualization of planet and spacecraft orbits, instrument view frustums, and sequenced imagery over time allow for comparison of observed data with model output. While New Horizons was a specific application, the techniques apply generically across all space missions with available SPICE data. SPICE compliance accurately defines positions, orbits, trajectories, orientations, and instrument views used in data collection for Earth and planetary science, heliophysics, and astrophysics.

The ability to read SPICE kernels during mission planning and flight, including actual configurations returned by telemetry during flight, will allow for both historical reconstruction and conceptual visualization of missions within OpenSpace. To date, such full SPICE visualization has only been available in specialized mission planning software, in produced videos such as those created by NASA's Scientific Visualization Studio, or in a limited capability with commercial software.[2] Never before has there been an interactive, deep level of SPICE capability in planetariums or digital theaters using freely available open source software. Visualizing spacecraft attitude with respect to an observation schedule is an important link between mission operations and the science it is designed to acquire. Visualizing the engineering of how space mission science is conducted is especially valuable with respect to public education.

DATA SOURCES

A major product of many space missions is comprehensive planetary maps of the bodies in our solar system. Examples of this include the MESSENGER mission that mapped Mercury, the Lunar Reconnaissance Orbiter that mapped the Moon, the Mars Reconnaissance Orbiter retrieving unprecedented details of the surface of Mars, and the numerous Earth-orbiting spacecraft providing time-resolved imagery of our own planet. Many of these maps are provided as images online through the Web Map Service standard. Through this standard, images are stored in a tree with varying resolution and only images of appropriate resolution are streamed from an online server to the client application. Streaming data from a variety of online sources enables the rapid exploration of datasets from different spacecraft and also provides the ability to access near-real time imagery without the need to download entire image catalogs, thus reducing the latency between data acquisition and visualization. NASA's Global Imagery Browse Services (GIBS) provides daily updates for many of their image sources, such as Suomi-NPP's Visible Infrared Imaging Radiometer Suite, which provides a full global view of Earth every day depicting weather and cloud coverage. Figure 1 shows an image of the three major hurricanes that impacted the United States in 2017 and that was generated from the GIBS dataset and is being inspected by a user in real time. In addition to visible light measurements, many other mission instruments are available through this standard, for example cloud layers, ozone concentration, and temperature measurements, thus providing an immense wealth of information that can be dissected in the context of the observing spacecraft.

MULTIUSER IMMERSIVE ENVIRONMENTS

The benefits of multiuser immersive environments are deeply rooted in the human desire to share experiences and the need for a social context in which knowledge can be understood and assessed. Immersive multiuser environments come in many different flavors, with different advantages and disadvantages. The available systems range from shared VR environments

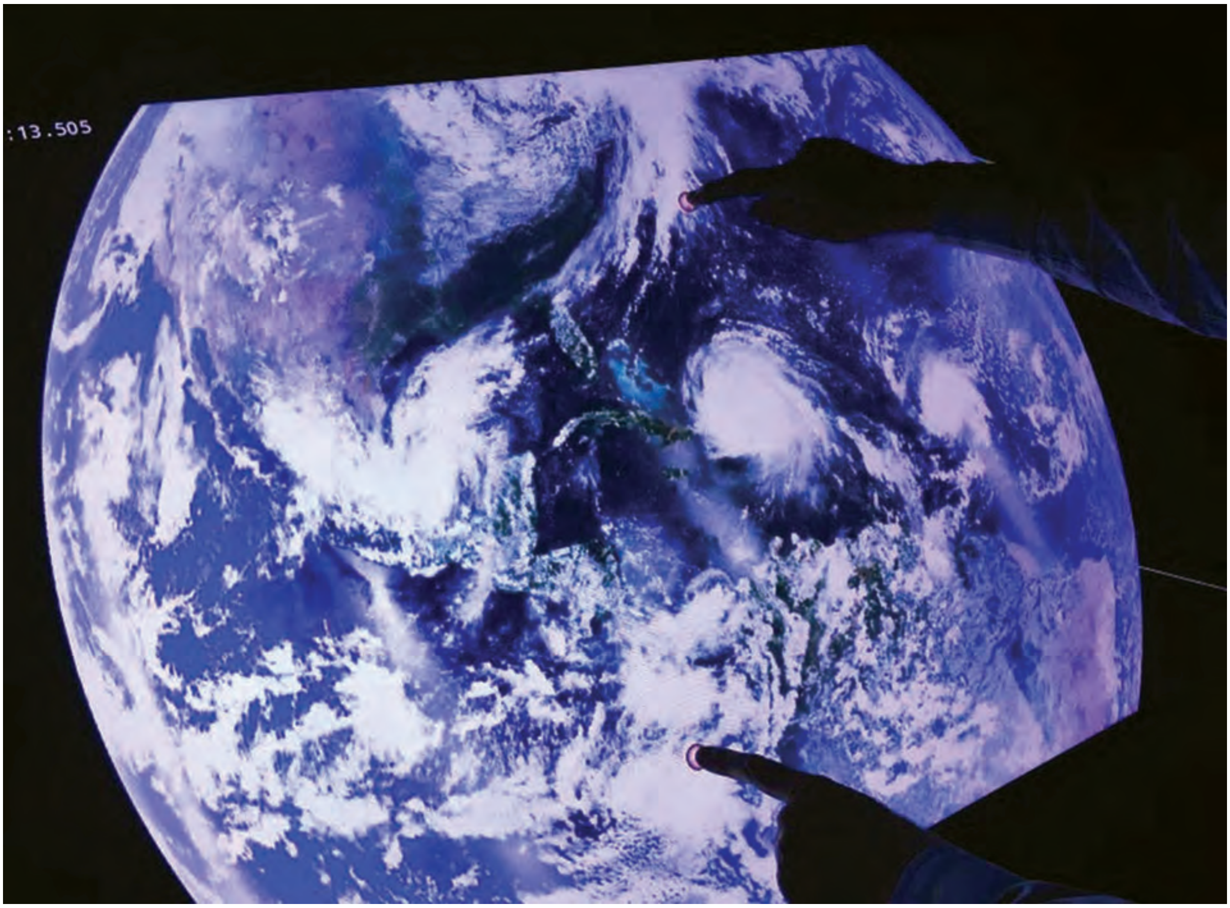

FIGURE 1. A view of hurricanes impacting the United States on 9 September 2017 based on a NASA's Global Imagery Browse Services dataset. From West to East (left to right): hurricanes Katia, Irma, and Jose.

using multiple head mounted displays to immersive environments such as CAVE(s) and large-scale display systems such as dome theaters. One key difference between these environments is the number of users who are tracked to provide correct first-person view of the virtual world being displayed. In multiuser head mounted systems, all users are tracked. In CAVE(s), only one person has the "correct view". In domes on the other hand, usually no user is tracked.

Our experience shows that multiuser immersion can be facilitated in several ways without the need of tracking users. In dome theaters the size of the display and the distance to the users make it possible to generate reasonably correct views for most of the users, and the size of the environments adds the advantage of scalability in number of simultaneous users. The same argument also holds for large flat displays, such as projection walls, that support high-resolution immersive experiences such as Figure 2 shows. In these environments, user interaction is, however, often limited to one user presenting and interacting. In dome theaters, this leads to the notion of a mediated immersive experience in which a presenter, often together with a pilot, guides the audience through a demonstration. As such, this becomes a powerful tool for immersion because the story and the story teller becomes integrated components of the immersion. OpenSpace was developed primarily for dome theater immersion. It benefits greatly from the interactive capabilities of the system whereby the multiusers perceive the 3D effect of the cosmos as long as the projected image reflects camera motion. If it stops, the users suddenly experience a 2D image projected onto the dome. The 3D illusion makes this environment extremely compelling for contextualized astrovisualization.

FIGURE 2. Multitouch table driving a powerwall using a linked OpenSpace session such that the viewpoint on the power wall is controlled by the touch table. The imagery is the Valles Marineris of Mars obtained from orbiting satellites and streamed to OpenSpace through the Web Map Service.

Interaction can also provide a path to immersion. Multiuser interaction on touch tables is common these days and enables immersion via shared input, along with content involvement among users sharing interaction, which can sometimes result in conflict but can also lead to discussion and user engagement.

TOUCH TABLE AND PROJECTION WALL

Using large tangible touch surfaces with a multitouch navigation interface is more engaging to users than a mouse and keyboard, while also enhancing understanding of the navigation control and thus, decreasing the learning curve of the system's user interface.[8] Additionally, combining a multitouch interaction model together with a screen-space direct-manipulation formulation produces a user-friendly interface. We have integrated the OpenSpace user interface with a commercial multitouch display table provided by the company Interspectral AB[9].

Astronomical visualizations have long been an interesting application for touch-based interaction. However, due to the scale of the solar system compared to its celestial bodies, any existing object rapidly becomes exceedingly small. A direct-manipulation solution alone becomes non-trivial as it requires 3D points in the scene to constrain the manipulation and such points cannot be well tracked in empty space.

The multitouch interface is implemented in OpenSpace using the TUIO library to support a variety of multitouch devices, although we primarily targeted the multitouch table. Using this interface, the user can interact with any celestial body in the scene and traverse it through multitouch gestures. For example, Figure 1 shows interaction with satellite imagery weather data. This allows the user to intuitively zoom into the geographical area of interest.

Additionally, OpenSpace provides the capability for coupling OpenSpace sessions across devices. These could be multiple planetariums, as was done for the New Horizons encounter coupling, among others, ANMH/Hayden Planetarium with the Norrköping Visualiseringscenter C Planetarium for a cross-continent, interactive visualization and story-telling of the mission while the encounter was taking place. With the multitouch interface, we found it advantageous to

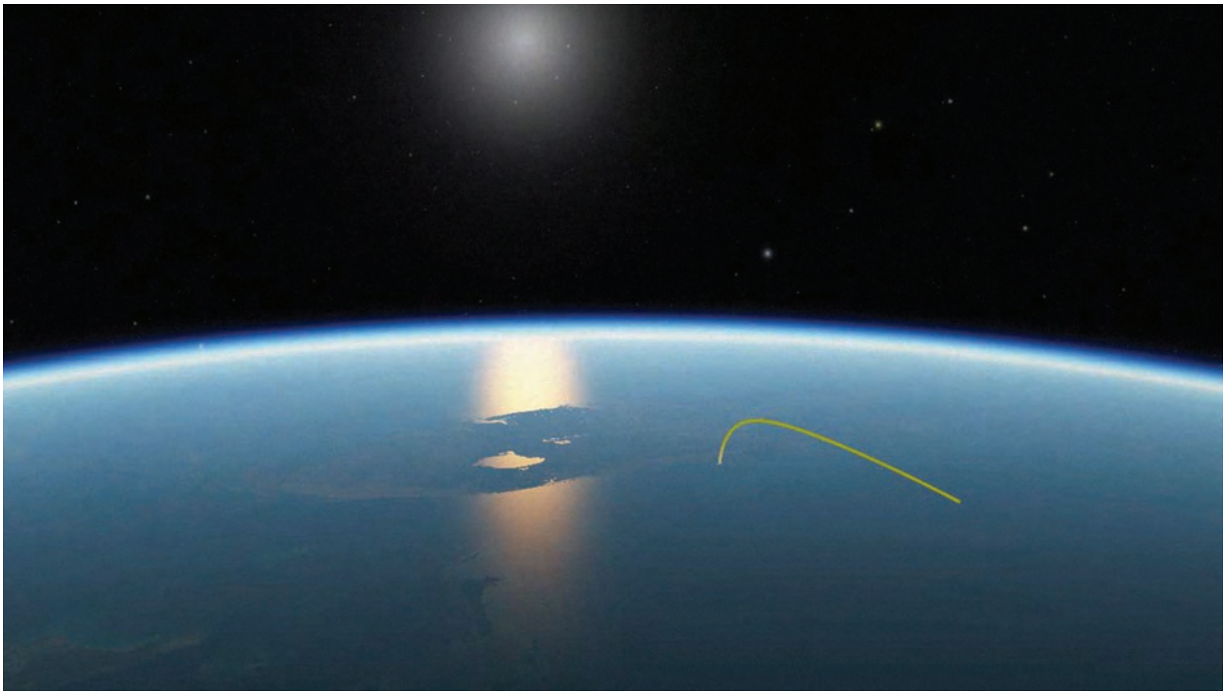

FIGURE 3. OSIRIS-REx lift-off from Cape Canaveral on 8 September 2016. The location of the spacecraft at a particular time-point is determined from NASA's Spacecraft Planet Instrument C-matrix Events data for the mission.

couple the touch-table interface with a large tiled display to effectively contextualize NASA missions to groups of K-12 students and the general public. Figure 2 shows the table driving the tiled display wall to seamlessly control the OpenSpace session for multiple participants. A YouTube video of the interaction is available.[10]

OSIRIS-REx Mission

OSIRIS-REx launched on 8 September 2016 at 7:05 PM EDT from Cape Canaveral. Figure 3 shows the trail of the spacecraft a few minutes after lift-off. OSIRIS-REx orbited the Sun and in September 2017, it used a gravity assist in an Earth fly-by to increase the orbital inclination to deviate from an orbit in the solar plane to an orbit matching the orbital plane of asteroid Bennu. Figure 4 shows the spacecraft's "slingshot" to the orbital plane of Bennu.

The OSISRIS-REx mission seeks to gather a sample from the surface of Bennu, a carbonaceous asteroid and return the sample to Earth. The sample may provide information on the formation of life on Earth and the Earth's oceans, such as whether Bennu contains organics, precious metals, or water. In addition to the sample, the mission will map the asteroid and use its advanced instruments to measure the Yarkovsky effect (non-gravitational forces which cause orbital deviation) as well as compare close-range observations with Earth-based observations.

By using NASA's SPICE kernels for this mission, the reconnaissance campaign mapping the asteroid can be visualized and the instrument activation highlighted as shown in Figure 5. By using the actual mission planning data contained in the NASA SPICE event file, which contains the mission timeline, a visualization and contextualization of the mission is accomplished. This can be shown in an immersive environment to describe the mission and its scientific actions to better inform and educate the public.

CONCLUSION

We have described OpenSpace, a software for the visualization and demonstration of NASA missions. The ability to interactively display and communicate missions during mission planning and/or mission activity and/or post-mission data analysis is a powerful method for public education. One key component is the ability to visualize the data acquisition operations

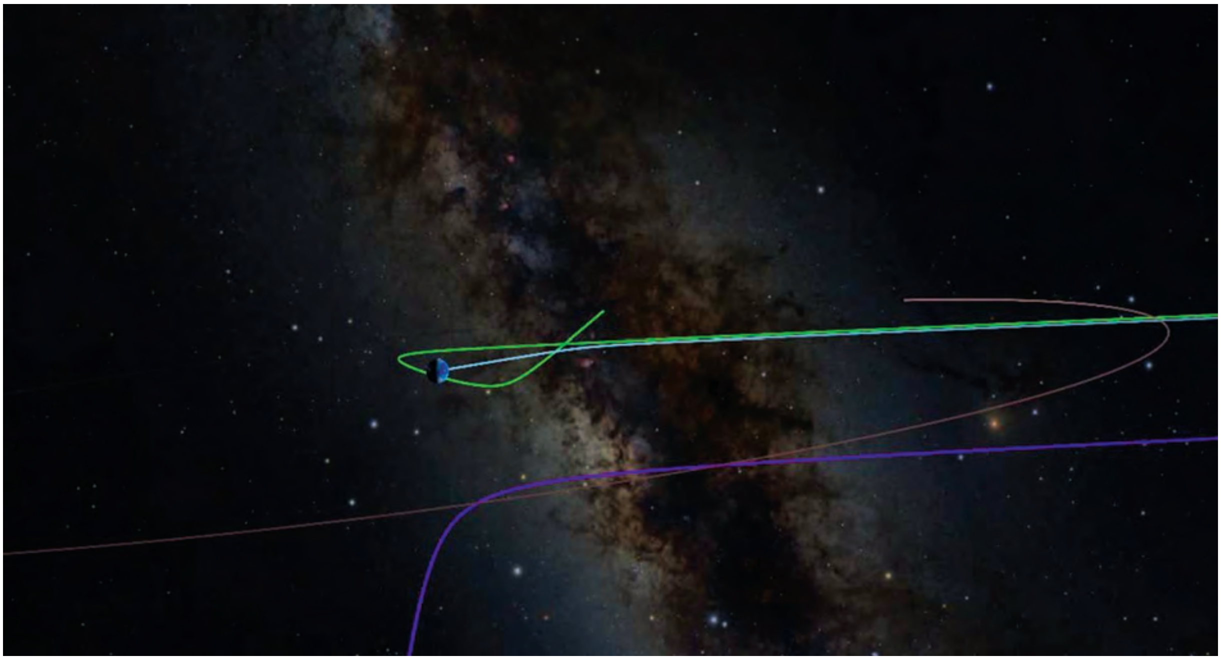

FIGURE 4. On 22 September 2017, OSIRIS-REx used Earth's gravitational field to slingshot from the solar orbit to the orbital plane of the asteroid Bennu. Earth is the cyan trail while OSIRIS-REx is the green trail. The celestial bodies and spacecraft locations are read from NASA's Spacecraft Planet Instrument C-matrix Events data for the OSIRIS-REx mission.

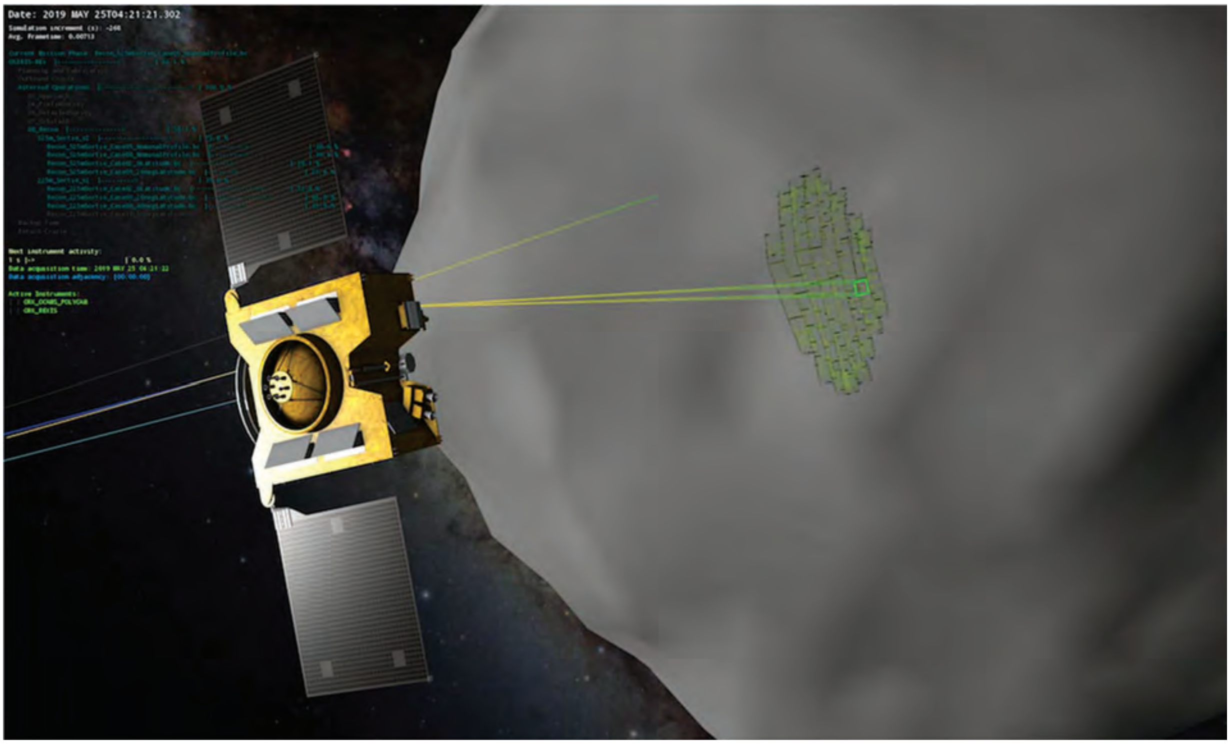

FIGURE 5. OSIRIS-REx maps Bennu on 25 May 2019 before obtaining the samples. The spacecraft and Bennu are modeled as geometry. Once the actual images are sent back to Earth, the mapping would appear as textures on the asteroid surface. The spatial/temporal locations of the asteroid, spacecraft, and its instruments are read from NASA's Spacecraft Planet Instrument C-matrix Events data for the mission.

from the viewpoint of the spacecraft using SPICE data. Immersive environments add to the realism of presenting NASA missions and a variety of immersive devices are supported; from display walls to touch tables to multiuser immersive theaters such as planetariums. We believe extensions to educate the public on space science, such as explaining the importance of space weather, will result in better public awareness and support of NASA missions. Our viewpoint is that diverse information needs to be aggregated and made available to visualization software that can integrate multiple sources, allowing a coherent visual communication to a wide variety of audiences and space science consumers. This requires the flexibility of a number of delivery and interaction mechanisms from the software.

REFERENCES

1. "The Systems Toolkit," Analytical Graphics, Inc., other; www.agi.com/products/stk.
2. S. Klashed et al., "Uniview -- Visualizing The Universe," *Proceedings Of Eurographics 2010 -- Areas Papers*, 2010, pp. 112–118.
3. A. Bock et al., "OpenSpace -- Changing The Narrative Of Public Dissemination In Astronomical Visualization From What To How," *IEEE Computer Graphics And Applications*, vol. 38, no. 3, 2018, pp. 112–118.
4. A. Bock et al., "OpenSpace: Public Dissemination Of Space Mission Profiles," *2015 IEEE Scientific Visualization Conference* (SciVis 15), 2015, pp. 112–118.
5. E. Axelsson et al., "Dynamic Scene Graph: Enabling Scaling, Positioning, And Navigation In The Universe," *Computer Graphics Forum*, vol. 36, no. 3, 2017, pp. 112–118.
6. K. Bladin et al., "Globe Browsing: Contextualized Spatio-Temporal Planetary Surface Visualization," *IEEE Transactions On Visualization And Computer Graphics*, vol. 24, no. 1, 2018, pp. 112–118.
7. C. Emmart et al., "OpenSpace: From Data Visualization Research To Planetariums And Classrooms Worldwide," *American Geophysical Union, Fall General Assembly*, 2016.
8. T. Isenberg, "Interactive Exploration Of Three-Dimensional Scientific Visualizations On Large Display Surfaces," *Collaboration Meets Interactive Spaces*, C. Anslow, P. Campos, J. Jorge, Springer, 2016.
9. "Interspectral AB: Commercial Provider Of Touch Table Technology," other; www.interspectral.com.
10. J. Bosson, "Touchtable Direct-manipulation With OpenSpace," other https://youtu.be/uL-2Rpdl-68, 13 July 2017; https://youtu.be/uL-2Rpdl-68.

ALEXANDER BOCK is a Moore-Sloan Postdoctoral Research Fellow with New York University's Center for Data Science and the Lead Developer of OpenSpace. Contact him at alexander.bock@nyu.edu.

CHARLES HANSEN is a professor of computer science in the School of Computing and an Associate Director of the Scientific Computing and Imaging Institute at the University of Utah. Contact him at hansen@sci.utah.edu.

ANDERS YNNERMAN is a professor of scientific visualization at Linköping University and an adjunct professor at the University of Utah. He is also the director of the Norrköping Visualization center and one of the co-founders of the OpenSpace project. Contact him at anders.ynnerman@liu.se.

Contact department editor Theresa-Marie Rhyne at theresamarierhyne@gmail.com.

EDITOR: Philippe Kruchten, University of British Columbia, pbk@ece.ubc.ca

This article originally appeared in Software vol. 36, no. 6, 2019

DEPARTMENT: SOUNDING BOARD

A Manifesto for Energy-Aware Software

Alcides Fonseca, Rick Kazman, and Patricia Lago

According to recent estimates, computing and communications could account for 20% of energy usage globally by 2025.[1] This trend shows no sign of slowing. The annual growth in power consumption of Internet-connected devices is 20%. Data centers alone are now accounting for more than 3% of global emissions. Even if you are not worried about this trend on the mega scale, you are likely concerned with the power consumption of the devices in your pocket, on your wrist, and in your ears.

Software, hardware, and network attributes all contribute to power usage, but little attention has been given to this topic by the information and communications technology (ICT) community. For example, as software engineers, we were never taught to consider, much less manage, the energy consumption of the software systems we created. Despite our lack of awareness and preparation, we are now facing an undeniable reality: the software community must learn to design for, monitor, and manage the energy usage of software. For this reason, we argue the need for energy-aware software and present a manifesto describing nine guiding principles. By *energy-aware software*, we mean software that is consciously designed and developed to monitor and react to energy preferences and usage. Energy efficiency is, therefore, one possible (but not the only) response to being energy aware.

This manifesto and the principles it proposes have arisen from our experience and from the experience of more than 100 researchers and practitioners who have participated in six international workshops on the engineering of green and sustainable software.[2] Why do we need a manifesto? Why now? Although there has been some attention to this area,[3] we believe it has been grossly insufficient given the high stakes involved. The vast majority of practitioners (and researchers) are completely ignorant of energy concerns; they, and the programs they create, are anything but energy aware.[4]

DOI No. 10.1109/MS.2019.2924498
Date of current version: 22 October 2019

THE NINE PRINCIPLES OF ENERGY AWARENESS

Energy awareness is a necessary but not sufficient precondition for energy efficiency. Energy awareness is required from all stakeholders, such as end users who may choose product A versus product B based on energy characteristics. Our goal in this manifesto is to call for changes in how we think and what we do. This will not come for free, but we believe that the cost of inaction is far greater.

Public Awareness Is Key for Widespread Adoption

We believe that the key to widespread adoption of energy-aware software is to sensitize and empower end users. The scary statistics regularly published have proven ineffective so far (the amount of energy being consumed by ICT, the increasing amount of energy consumed by cloud providers, the massive amounts of data being stored in data centers as opposed to the negligible percentages of data being actually used, and so forth). Neither do the worrisome energy-consumption predictions seem to spur us to action (such as the increasing number of things being connected to the Internet or the booming growth in mobile devices and their increasingly sophisticated applications).

We need to turn these alarming trends into an opportunity: 1) to sensitize end users to the amount of energy consumed by the software they use and 2) to create awareness of the fact that software solutions

with similar features may yield very different energy profiles. Imagine that we were able to attach "green" labels (like Energy Star ratings) to the apps available in Google Play or the Apple Store. End users of mobile devices could then compare the apps they are seeking (such as apps for "meditation" or for "scanning documents") not only in terms of features, rating, and price; if green labels were added, end users would be empowered to make better-informed decisions based on the labeled level of energy use. Such labels would force software companies to invest in optimizing the energy impact of their applications (if they wanted to improve their market position), with a resulting positive effect on the resources (for example, cloud services and networks) that such applications use.

Incentives for Software Stakeholders Should Be Provided

We believe that, although some people are altruistic, most people respond to incentives. Therefore, such incentives should be put in place to encourage the creation of energy-aware systems, which will, in turn, lead to energy efficiency. Such incentives would also help raise the consciousness of engineers and end users. Most stakeholders need to be incentivized to actively pursue software and systems that are energy aware. This is a key step to raising the priority of energy-aware software in our companies and in our society. Such incentives, both positive and negative, have been used successfully for decades in other parts of our economy: there are so-called sin taxes to discourage drinking and smoking, and there are tax rebates to encourage people to make their homes more energy efficient.

Energy-Aware Software Engineering Should Be a Priority for Every Stakeholder

The efficiency of system energy consumption is relevant for everyone, regardless of their roles in the development process—from end users, concerned with their battery life, to business owners, concerned with reducing their electricity costs and from the developers, who should understand the energy impact of their contributions, to the product owners, who have to decide in what way energy efficiency is a requirement.

Energy-related information should flow to all stakeholders. Early in the development process, clients should be asked about their energy requirements. These should then be propagated to developers, testers, and operations personnel so that they can be taken into account, tested for, and monitored. Results from these phases should be reported back to the stakeholders, just like other important metrics.

Education and Professional Training Should Cover Energy-Aware Software by Default

To create energy-aware software, we must educate the next generation of engineers who need to acquire the competencies (and provide training for the current-generation workforce, too). Depending on the target audience and the learning objectives, educational programs may adopt a centralized approach (concentrating the competencies crucial for energy-aware systems in one or two courses), distributed (revising traditional courses to include competencies for energy awareness), or blended (including both types of courses across the curriculum). Just as every programmer should understand algorithmic complexity, energy awareness must become a standard competency of every ICT practitioner and a standard consideration for every decision maker and end user.

Broad Adoption Requires Attention to Usability

To encourage broad adoption of energy-aware systems, we must pay attention to the usability, from a developer perspective, of the tools that we create. Best practices for energy-aware software engineering should end up being embedded in the tools, packages, and frameworks we create so that software engineers do not need to reinvent the wheel. You should not need a soldering iron or a circuit diagram—or a degree in electrical engineering—to manage the energy consumption of an ICT system. Furthermore, it should be easy to reuse experience and best practices for engineering energy-aware software.

Energy Awareness Should Be Engineered Throughout the Lifecycle

Energy awareness can and should be treated like an architectural quality attribute,[5] no different from how we design for, analyze, prototype, and manage other

qualities in an architecture such as modifiability, performance, availability, or security. This means that architectures are design blueprints with system-wide resource-management strategies. For example, power usage requirements should be explicitly collected during requirements gathering, designed for, and tested for.

In particular, energy awareness and energy efficiency must be designed into a system early in its lifecycle and considered when making major changes to the system. Leaving this consideration until the system is already built is a recipe for disaster. Experience tells that any quality that you address late in development tends to be treated superficially (or at great cost). If you don't measure it, you cannot manage it.[6] This energy awareness will have an associated cost, in terms of system complexity, and this cost must be acknowledged and accepted by the ICT community.

Software Quality Should Not Come at the Expense of Energy Awareness

Energy-aware software development does not imply that energy efficiency should be prioritized over other quality attributes. Being energy aware involves taking into account the energy consumption of software across the software development lifecycle. Energy awareness cannot be ignored and should be explicitly considered in tradeoff decisions, even if the final decision is to prioritize some other quality attribute over it.

Making software development processes explicitly energy aware allows for stakeholders to be informed of the options chosen regarding the energy consumption of software. Data about the energy impact of design choices can be used to inform future decisions and improve designs with respect to their energy efficiency.

Energy Awareness Demands Dynamic Adaptability

Energy awareness is heavily influenced by the context in which software is being used, both because resource availability varies over time (for example, the battery load is limited, network connectivity is location dependent, or electricity rates change during the day) and because as end users move, their needs change (for example, driving the most energy-aware route depends on our location, and if we need to charge our car battery, it depends on the availability of charging stations nearby).

We believe that software should perform its core functionality while simultaneously ensuring energy awareness. If this happens, end users can count on software applications to be reliable and to promise the best tradeoff between energy and functionality. To do so, software must be able to detect that its context has changed and that (possibly) some resources have become scarce and flexibly adapt by replacing them with alternatives or downgrading the delivered functionality.

We Value Measures Over Beliefs (and Reliable Trends Over Precision)

Energy awareness can be easily ignored early in the software lifecycle if there are no data to support the fact that energy should be a concern. Another common pitfall is to believe that optimizing for energy efficiency is difficult, if not impossible.[7] Measuring energy consumption is a first step toward energy awareness. At a minimum, it provides a baseline to compare when introducing changes into the system. Costly additions, in terms of energy, may need to be revised or even discarded if low power consumption is a priority.

Different methods for measuring energy consumption exist. Because software is often dynamic and depends on runtime information, such as the size of exchanged data, understanding the trend of energy efficiency is more important than knowing the raw logged values. Just like the Big-O notation in time complexity, energy efficiency can be seen as another complexity metric.

We believe that energy-aware ICT is inevitable for many reasons: economic reasons, sustainability reasons, and because users will increasingly demand it. To hasten its emergence, we have polled the R&D community in a series of workshops and collected these nine principles to help focus the emergence of energy awareness as a true subdiscipline of ICT and software engineering. We can see that the successful emergence of this discipline depends on three foundations: 1) awareness, 2) education and training, and 3) the creation of a body of engineering knowledge. We need all of them to make energy-aware software a reality.

REFERENCES

1. "'Tsunami of data' could consume one fifth of global electricity by 2025," *The Guardian*, Dec. 11, 2017. [Online]. Available: https://www.theguardian.com/environment/2017/dec/11/tsunami-of-data-could-consume-fifth-global-electricity-by-2025
2. "*6th International Workshop on Green and Sustainable Software*." [Online]. Available: http://greens.cs.vu.nl
3. C. Becker et al., "Sustainability design and software: The Karlskrona manifesto," in *Proc. 37th Int. Conf. Software Engineering*, vol. 2, pp. 467–476.
4. G. Pinto and F. Castor, "Energy efficiency: A new concern for application software developers," *Comm. ACM*, vol. 60, no. 12, pp. 68–75, Dec. 2017.
5. N. Condori-Fernandez and P. Lago, "Characterizing the contribution of quality requirements to software sustainability," *J. Syst. Softw.*, vol. 137, pp. 289–305, Mar. 2018. doi: 10.1016/j.jss.2017.12.005.
6. "*Peter Drucker*," Wikipedia. Accessed on: Sept. 16, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Peter_Drucker
7. C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What do programmers know about software energy consumption?" *IEEE Softw.*, vol. 33, no. 3, pp. 83–89, 2016. doi: 10.1109/MS.2015.83.

ALCIDES FONSECA is an assistant professor at LASIGE, Faculdade de Ciências da Universidade de Lisboa, Portugal. Contact him at alcidesfonseca@gmail.com.

RICK KAZMAN is a professor at the University of Hawaii, Mãnoa, Honolulu, and a visiting scientist at the Software Engineering Institute, Pittsburgh, Pennsylvania. Contact him at kazman@hawaii.edu.

PATRICIA LAGO is a professor in software engineering at the Vrije Universiteit Amsterdam, The Netherlands. Contact her at p.lago@vu.nl.

EDITOR: Gerd Kortuem, g.w.kortuem@tudelft.nl

DEPARTMENT: IOT NEWS

This article originally appeared in pervasive COMPUTING vol. 18, no. 4, 2019

Next Generation IoT: Toward Ubiquitous Autonomous Cost-Efficient IoT Devices

Moustafa Youssef, ***Alexandria University***
Mahbub Hassan, ***University of New South Wales***

IoT enables a new set of novel applications that promise to transform the physical world as we know it today. Current IoT applications have demonstrated the feasibility and promise of this technology, but there is still a large space open for innovation. We present our vision for the next generation of IoT devices: IoT devices that are autonomous, in terms of energy and communication, as well as cost-efficient to deploy. We present enabling-technologies to realize our vision and open research challenges.

AUTONOMOUS IOT DEVICES

Iot devices are expected to be deployed everywhere at scale, from well-powered homes to isolated environments and rural areas with limited power supply. For IoT devices to work independently, for an extended time and with cost-efficiency, it is important for them to be self-powered and consume minimal energy for their different functionalities.

Technologies proposed over the years for energy harvesting include RF-, light-, and mechanical/vibration-based.[1,2] Each technology has its own suitable environment for deployment, and they can be combined together to harvest more energy. New generations of solar panels, both opaque and **transparent**, are promising technologies for future IoT devices because they allow for full coverage of the device body by solar panels, increasing the amount of harvested energy.[3] For example, a smartwatch can be powered by opaque solar cells on the belt and transparent cells right on the screen (see Figure 1). This is possible because the new generation of transparent solar cells absorb and harvest energy from infrared and ultraviolet light, but let the visible light pass through so we can see through them like a clear glass.

Back-scatter communication can also be used to further reduce the operational energy of the IoT device, e.g., by leveraging the ambient RF signals to communicate with minimal energy to nearby devices, e.g., a gateway deployed in the field.[4] Such backscattering is proving effective for data communication at kilo bps speed while consuming power only at a few micro Watts.

VALUE-ADDED SERVICES FOR AUTONOMOUS IOT DEVICES

Another interesting direction to save the energy of IoT devices, decrease their deployment cost, and increase their autonomy is to provide value-added services, i.e., new applications that leverage the existing hardware and signals in novel ways. For example, the device-free identification concept,[5] where an object's presence,[6] location,[7,8] and features[9] can be determined based on analyzing the changes in the ambient RF signals can also be used to further sense the environment in

DOI No. 10.1109/MPRV.2019.2947974
Date of current version 21 January 2020.

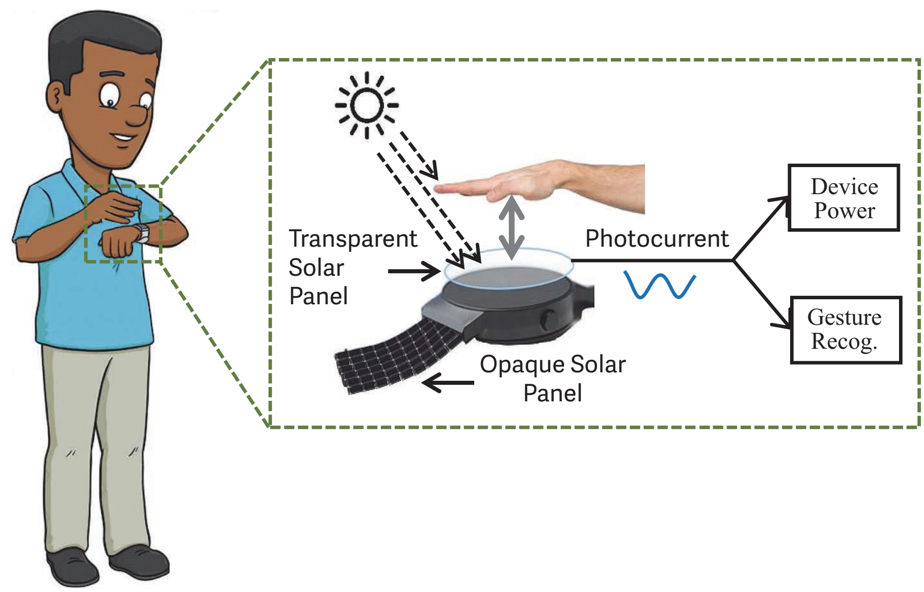

FIGURE 1. Smart-watch surface can be over-fitted with both opaque (on the straps) and transparent (on the surface) solar panels to maximize the harvested energy. The change in the harvested current due to the user's hand blocking the light arriving at the watch can be used to detect different gestures.

a sensor-less fashion, i.e., without deploying special sensors on the nodes. This can be leveraged in applications such as intrusion detection, activity recognition, and gesture-based device control, among many others.[10] Extending this to different energy-harvesting technologies provides both new opportunities and challenges.

For example, the change in the harvested power due to the user interaction with the device can be used to infer different user gestures. For instance, a user's hand blocking a solar panel will reduce or eliminate the energy harvested from light. The analysis of the harvested energy level can then be used to estimate the specific interaction pattern with the device as well as some of its attributes, e.g., how far away the user's hand is from the device, number of repetitions, and speed of motion.[11] Similarly, the human body affects the received ambient RF signal, which can be leveraged to estimate the presence of people in the area of interest based on the changes in the RF-harvested power. Along the same line, a vibration-energy harvester can be used to estimate the spoken words of a person.[5] What is common about all these ideas is that they use the harvested energy, without using any special hardware, to detect how the user interacts with the devices, whether through hand gestures or spoken words. This allows energy-efficient interactions with the future IoT devices in a cost-efficient and user-friendly manner. Moreover, voice and hands-free interaction with the IoT devices will enable persons with vision and hearing impairments to operate future IoT devices everywhere with ease.

OPEN CHALLENGES

Several challenges still need to be addressed to realize our vision for the ubiquitous autonomous cost-efficient IoT devices. One challenge is to optimize energy harvesting under different IoT operational scenarios. For example, if vibrational energy harvesting is used, one must tune the resonance frequency carefully to resonate with the vibration frequency of the device, because that is when the maximum energy is harvested. However, an IoT device may be subject to different levels of vibrations at different times, making optimal energy harvesting difficult. Material and mechanical engineering researchers are working on a new breed of hardware that may solve this problem in the future by dynamically tuning the resonance frequency of the device. Similarly, if transparent solar cells are used on the mobile device screen, the energy output will be low if the transparency is high for a better quality of viewing experience (Table 1). However,

Solar cell type	Material	Current Density	Power Conversion Efficiency
Opaque	Silicon	35 mA/cm^2	15–21%
Transparent	Organic	3.82 mA/cm^2	7–12%

TABLE 1. Example on the different solar cell types parameters.[11]

the user does not always look at the screen. As such, mechanisms are needed that can dynamically adjust the transparency so it can be tuned to the highest transparency when the user is looking at the screen and switch to the lowest transparency for the highest energy harvesting at other times. We still do not have materials that can fulfill these requirements, but perhaps techniques such as dynamic reflection control can be used at a separate layer to address this issue.[12]

To support autonomous communication, we should look for techniques beyond the radio-based backscattering to ensure that IoTs can communicate even when there is no WiFi or other ambient radio signals available. For fully autonomous battery-free IoTs, which are powered only through energy harvesting, we need novel communication paradigms that can effectively communicate in the face of dynamic and unpredictable power supply. There are new developments in radio communications that promise to generate extremely low-energy radio pulses in the terahertz frequency band using graphene.[13] The energy consumption of such graphene pulses can be as low as only a few atto Jules due to the extremely short pulse durations on the order of femtoseconds, which is made possible because of graphene's ability to resonate in an extremely wide frequency band. How such graphene-based radio technology can help achieve communication autonomy for the next generation battery-free IoTs is a new challenge open for research.

Unpredictability in power supply also creates challenges for long computation tasks that may have to be executed by energy-harvesting IoTs. Any disruption in power supply in the middle of a computation task may force the IoT device to start all over again when the power supply is back. How to jointly coordinate computation tasks and energy harvesting within such autonomous IoTs require careful design of novel computational models.

The heterogeneity and scale of IoT devices raise a number of new challenges to supporting sensor-less sensing, including quantifying the quality of the different signals coming from the different heterogeneous IoT devices, performing multi-person identification and separation, increasing the SNR, handling shadowing effect, and reducing the calibration effort (being device and person independent/agnostic).

Different people in the area of interest will create several challenges including interference. Interference can be handled in some application domains by leveraging the redundancy of the IoT devices. For example, in an RF environment, the IoT devices installed in the area may hear different RF transmitters (e.g., WiFi APs) concurrently. The information collected from all devices about the detected APs along with their locations can be fused to separate the different events in the area. The idea is that an interfering human may affect some IoT devices but not all of them. Counting the number of people in the area of interest and separating their interference signatures is also a potential direction for research.

Due to the small amount of harvested energy, the SNR of the harvested signal is usually low, making it harder for a classification algorithm to detect specific interactions with the device. One way to handle this is to leverage deep learning algorithms and attempt automatic robust feature selection. Another idea is to try to increase the SNR, e.g., by using active light sources to interact with the solar panels to do the gesture, instead of relying on the change in the ambient light. Using active light sources has the extra benefit of increasing the harvested energy too.

Another area to explore is multiservice IoT devices. The idea is to allow the same deployment of IoT devices to perform different services concurrently such as intrusion detection and gesture recognition. This increases the value of the deployed IoT nodes and opens the door for novel applications.

CONCLUSION

Future IoT devices will have autonomous operation requiring zero human intervention for maintenance, allowing for ease of operation at scale and

mass adoption. A vision to achieve this goal is through multitechnology energy harvesting and sensor-less sensing. The new technologies for energy harvesting through transparent solar panels will allow energy-harvesting from the entire surface of virtually any device. Leveraging the changes in the energy-harvested signal to sense the environment as well as to detect the user interaction with the IoT device will not only provide a ubiquitous, user-friendly way of interacting with IoT devices but will also indirectly reduce the energy consumption and deployment cost of the IoT echo-system by providing value-added services without any extra hardware.

REFERENCES

1. B. Yang and K.-S. Yun, "Piezoelectric shell structures as wearable energy harvesters for effective power generation at low-frequency movement," *Sensors Actuators A, Phys.*, vol. 188, pp. 427–433, 2012.
2. S. Kim, et al., "Ambient RF energy-harvesting technologies for self-sustainable standalone wireless sensor platforms," *Proc. IEEE*, vol. 102, no. 11, pp. 1649–1666, Nov. 2014.
3. M. B. Upama, et al., "High performance semi-transparent organic solar cells with 5% PCE using non-patterned MoO3/Ag/MoO3 anode," *Current Appl. Phys.*, vol. 17, no. 2, pp. 298–305, 2017.
4. P. Zhang and D. Ganesan, "Enabling bit-by-bit backscatter communication in severe energy harvesting environments," in *Proc. NSDI*, 2014, pp. 345–357.
5. M. Youssef, M. Mah, and A. Agrawala, "Challenges: Device-free passive localization for wireless environments," in *Proc. 13th Annu. Int. Conf. Mobile Comput. Netw.*, Sep. 9–14, 2007.
6. A. E. Kosba, A. M Saeed, and M. Youssef, "Robust WLAN device-free passive motion detection," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2012, pp. 180–189.
7. H. Abdelnaser, R. Muhammad, I. Sabek, and M. Youssef, "MonoPHY: Mono-stream-based device-free WLAN localization via physical layer information," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2013, pp. 4546–4551.
8. A. Eleryan, M. Elsabagh, and M. Youssef, "Synthetic generation of radio maps for device-free passive localization," in *Proc. IEEE GLOBECOM*, 2011, pp. 1–5.
9. H. Abdelnasir, K. Harras, and M. Youssef, "UbiBreathe: A ubiquitous non-invasive WiFi-based breathing estimator," in *Proc. ACM Mobihoc*, 2015, pp. 277–286.
10. M. Youssef, "CoSDEO 2016 Keynote: A decade later — Challenges: Device-free passive localization for wireless environments," in *Proc. 5th IEEE COSDEO Workshop, Conjunction IEEE PerCom*, 2016, pp. 1–2.
11. 11.D. Ma, et al., "SolarGest: Ubiquitous and energy-free gesture recognition using solar cells," in *Proc. ACM Mobicom*, 2019, pp. 1–15.
12. J. Park, J.-H. Kang, S. J. Kim, X. Liu, and M. L. Brongersma, "Dynamic reflection phase and polarization control in metasurfaces," *Nano Lett.*, vol. 17, no. 1, pp. 407–413, 2017.
13. J. M. Jornet and I. F. Akyildiz, "Femtosecond-long pulse-based modulation for terahertz band communication in nanonetworks," *IEEE Trans. Commun.*, vol. 62, no. 5, pp. 1742–1754, May 2014.

MOUSTAFA YOUSSEF is a Professor of computer and systems engineering with Alexandria University, Alexandria, Egypt. Contact him at moustafa@alexu.edu.eg.

MAHBUB HASSAN is a Professor of computer science and engineering with the University of New South Wales, Sydney NSW, Australia. Contact him at mahbub.hassan@unsw.edu.au.

EDITOR: Erik P. DeBenedictis, erikdebenedictis@gmail.com

This article originally appeared in Computer vol. 52, no. 10, 2019

Powerball and Quantum Supremacy

Erik P. DeBenedictis

In an attempt to communicate better with the nonexpert public, this article explains recent developments in quantum supremacy in a form that is closer to home: lotteries.

Society's embrace of positional numbers about 1,500 years ago may be a road map for today's quantum information revolution. Positional numbers provide an exponential advantage over the unary representations found on prehistoric artifacts. The unary representation of values in the millions uses millions of symbols, but only $\log_{10}$(millions) = 7 in our current Hindu–Arabic positional notation.

One current pre-quantum-supremacy demonstration[1] runs essentially the same problem on a one-chip, n-qubit, $n = 72$ quantum computer and, for comparison, the world's largest supercomputer, Oak Ridge's Summit. The most straightforward supercomputer code would require an impossibly large 2^n entry table, but researchers have found clever methods of reducing the memory requirements. The advantage may be exponential or nearly so.

Positional numbers may have been invented by a handful of mathematicians, but they are now taught to children. The ability of positional numbers to represent large amounts of money allowed economies to grow, the ability to represent real numbers through decimal fractions allowed the development of science, and automatic processing of numbers by computers led to civilization-changing applications like the Internet. If quantum information is to have the same downstream effect, we will need to teach children destined to be experts in other fields enough about quantum information to advance their chosen field with quantum information.

A QUANTUM INFORMATION MODEL

Powerball, where there can be multiple winners of a single lottery, is an existence proof that a significant fraction of society can understand statistical trials with correlations. Let's see if this can be a starting point for a user-level quantum information model.

Lottery tickets and qubits both have two phases in their lives. A ticket is characterized by pick numbers during the first phase of its life. The second phase starts with the drawing, where the pick numbers become irrelevant, and all that matters is whether a ticket wins or loses. A ticket viewed in isolation either wins or loses at random, yet tickets with the same pick numbers will win or lose at the same time. Lottery tickets actually have multiple pick numbers, so there can be complex correlations between tickets that win and lose.

Qubits have a similar life cycle. Quantum computers actually create qubits in a standard form, which physicists call |0>. To create a custom ticket, or qubit, single-qubit quantum gates effectively modify the pick numbers. For greater flexibility, two-qubit quantum gates can mix up the pick numbers, although some of these gates increase the size of the pick number space multiplicatively, for example, a ticket with four pick numbers and another with eight yielding two tickets with 32 each. The drawing, which physicists call *qubit measurement*, yields a result that has a lot of

DOI No. 10.1109/MC.2019.2930166
Date of publication: 24 September 2019

randomness but also has correlations due to common pick numbers.

We teach long division in elementary school, but the math required to invent the long division algorithm is taught only in college and only to math majors. Let's see if we can explain quantum supremacy and Shor's algorithm well enough for somebody to appreciate and apply the result but not necessarily well enough to understand how it was invented or to improve it.

QUANTUM SUPREMACY IN DETAIL

Some of the leading pre-quantum-supremacy work[1,2] views a quantum computer as a machine for simulating lotteries, or qubits. The quantum computer is configured with a random qubit gate circuit. Each run starts with *n* lottery tickets, or qubits, initialized in standard yet uncorrelated states. Figure 1 shows this with simple sequential pick numbers. The tickets are passed through the series of quantum gates and then measured. Today's qubit technology is called *Noisy Intermediate-Scale Quantum*, which currently has a limit of $n \leq 72$ and a maximum of around 40 levels of quantum gates before errors dominate and the result is not useful.

If you run the quantum computer repeatedly on the same gate configuration, you will get a sequence of *n*-bit patterns representing the winning tickets across multiple lotteries. To test the quantum computer, you might think of running it repeatedly and computing the probability of each of the 2^n patterns. If you were more ambitious, you could also compute the same probabilities on a classical computer. The quantum and classical probabilities won't match exactly because today's qubits are notoriously noisy, but comparing the probabilities would characterize the quantum computer's noise and could be very useful.

There are several problems. As mentioned previously, the table of probabilities for one of today's state-of-the-art quantum computers would have 2^n

> SOME OF THE LEADING PRE-QUANTUM SUPREMACY WORK VIEWS A QUANTUM COMPUTER AS A MACHINE FOR SIMULATING LOTTERIES, OR QUBITS.

$= 2^{72} = 4.7 \times 10^{21}$ entries, which is too big even for the world's largest supercomputer. Filling the table would be even more challenging. However, some lotteries, or quantum gate circuits, will have a much smaller number of highly probable winning patterns. Say there were, for example, a dozen patterns with 1% probability. If you run a lottery 10,000 times, each of these patterns will occur about 100 times. So, 10,000 runs would identify the highly probable patterns and compute their probability to about one decimal place accuracy.

Let's look at specific numbers for one such demonstration.[1] A 72-qubit quantum computer is run on a random quantum circuit. Decoherence times are under 1 ms, so an actual run should be about that long.

The experimenters then computed the correct probabilities for some of the more probable states on

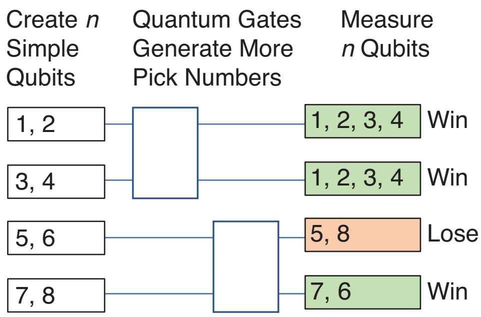

FIGURE 1. A user-level illustration of a quantum computer. Draw: 1, 2, 3, 4, 6, 7. The results (top to bottom) are { Win, Win, Lose, Win } = $1101_2 = 13$.

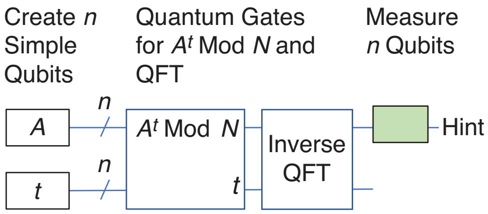

FIGURE 2. An illustration of Shor's algorithm. Additional temporary qubits will be required. QFT: quantum Fourier transform.

Summit. The majority of the research was the development of the (classical) supercomputer code to reproduce the results of the 72-qubit quantum computer. To reduce the memory requirements, the authors outline techniques that can compute an adequate answer based on time–space tradeoffs and exploiting the fact that a short decoherence time limits the depth of the circuit and, hence, its complexity. Yet runtimes on Summit are still in the hours. The problem is highly parallel, so runtime isn't the right measure. But Summit burns about 104 times the energy as the quantum computer doing the same thing.

Did the researchers[1] demonstrate quantum supremacy? It seems that a one-chip, 72-qubit quantum computer can do things that swamp the world's largest supercomputer. But is this a useful application or a contrived benchmark?

SHOR'S ALGORITHM

I suggest that Shor's famous factoring algorithm (Figure 2) can be appreciated as an application of a lottery machine, although the simplified model would not have allowed the invention of the system in the first place nor would it be possible to improve the algorithm. This could be useful for the same reasons we teach children long division. The lottery machine description also shows why Shor's factoring algorithm, a practical problem, is too complex for today's quantum computers.

The quantum hardware for Shor's algorithm does not actually factor numbers but instead gives a hint to a classical computer about where to look for a factor. The classical computer does a few arithmetic operations based on the hint, yielding a candidate factor, which is then checked by division. Some hints do not yield factors, and, of course, the hints won't lead to a valid factor if the quantum computer makes a mistake.

Factoring N requires $n = 2 \times \lceil \log_2 N \rceil$ lottery tickets, or qubits, each representing a bit of two n-bit numbers A and t. Temporary qubits are also required, and there could be quite a few. The qubits pass individually through what physicists call *Hadamard gates*, setting the pick numbers so each has a 50% chance of winning, with all qubits uncorrelated.

The lottery tickets, or qubits, are run through a circuit that has the same structure as a (reversible) Boolean logic circuit that computes A^t mod N. Today's logic designers should be able to create the circuit using $O(n^3)$ gates configured into ripple-carry Boolean adders without having to reference and textbook. Fast multiplier circuits[3] would be more efficient for large n. To provide the status on the state of factoring, the RSA challenge[4] of factoring a 768-bit number was completed in December 2009. This puts the number of gates for the A^t mod N circuit far out of range of today's quantum computer hardware.

I'll summarize the rest of Shor's factoring algorithm just to show the connection to quantum supremacy experiments. If N, $f(t) = A^t$ mod N will be zero for all integral values of t but an essentially random function on the unit circle otherwise. There is a well-known QFT technique in quantum computing, the inverse of which essentially computes $e^{i\pi f(t)}$, with some constants. If A is a factor of N, this will be the fast Fourier transform (FFT) of a constant, which is a delta function at zero frequency. If not, it will be the FFT of white noise, which is a constant function. Since the drawing probability is the amplitude squared, there will be a vastly increased probability of the drawing, or qubit measurement, yielding a win–lose pattern corresponding to a value of A that is a factor of N. This value becomes the hint. However, it's still a lottery, so a run of the quantum computer won't always lead to a factor.

When I got involved in quantum computing about 15 years ago, the people in the field were physicists and algorithms experts. There was no possibility of running an interesting algorithm on the hardware of that era. So, algorithms were developed for applications requiring millions of qubits at the same time that physicists struggled to make one and two qubits and the corresponding gates. Now, we

have improved hardware capable of running downscaled algorithms and yielding a result that, with a bit of faith, looks like it might work.

Progress will continue, provided high public interest keeps governments funding R&D and quantum R&D companies' stock market valuations remain high so they have money. For this to happen, the quantum technical community needs a support base that can appreciate the results and the potential of quantum computers. However, a significant part of the support base will not require the understanding of an engineer or scientist but need to understand quantum concepts just enough to invest and vote. We just need something more approachable than math and physics lingo and jokes like 1/√2 |dead> + 1/√2 |alive>.

If this approach catches on, we'll need to make a quantum-specific lottery and name it something other than *Powerball*. The name *Powerball* implies a specific set of rules and a drawing method that would not create quantum speedup.

REFERENCES

1. B. Villalonga et al., Establishing the quantum supremacy Frontier with a 281 Pflop/s simulation. 2019. [Online]. Available: arXiv:1905.00444
2. A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, Validating quantum computers using randomized model circuits. 2018. [Online]. Available: arXiv:1811.12926
3. R. Rines and I. Chuang, High performance quantum modular multipliers. 2018. [Online]. Available: arXiv:1801.01081
4. Wikipedia, *RSA Factoring Challenge*. Accessed on: Aug. 19, 2019. [Online]. Available: https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

ERIK P. DEBENEDICTIS a volunteer supporting the IEEE Computer Society, IEEE Council on Superconductivity, IEEE Rebooting Computing, and the IEEE Quantum Initiative. Contact him at erikdebenedictis@gmail.com.

EDITOR: Erik P. DeBenedictis, Zettaflops, LLC, erikdebenedictis@gmail.com

This article originally appeared in
Computer
vol. 53, no. 2, 2020

DEPARTMENT: REBOOTING COMPUTING

Beyond Quantum Supremacy

Erik P. DeBenedictis, *Zettaflops, LLC*

Demonstrations of quantum supremacy have become convincing enough that it's time to think seriously about what comes next.

The media tantalizes the public by having single quantum computer chips outperform the world's largest computer,[1] bringing publicity to the participating companies. The demonstrations are driven by real progress on analog, quantum, and scalable logic families and basic automata, that is, state machines.[2] The effort also reveals an emerging quantum computer architecture flexible enough to build systems for targeted applications.

Quantum supremacy is the demonstration of a quantum computer performing a computation that is infeasible or impossible for a regular computer. As this article is being written, Google has a quantum computer chip that indisputably outperforms the world's largest supercomputer, Summit, which is located at the Oak Ridge National Laboratory, Tennessee.[1] There is just one issue, albeit on just one program. The quantum computer chip runs for 200 s on a problem that Google estimates would take Summit 10,000 years—establishing quantum supremacy. However, IBM says it knows how to program Summit to perform the simulation in 2.5 days—refuting the claim of quantum supremacy.[3] In my opinion, both the 10,000-year and 2.5-day figures are good enough.

Quantum supremacy requires advances in several areas at once. Today's transmon qubits lose their state in 10 s of μs, meaning that the quantum program must run to completion in a few dozen gate delays or cycles. Google's current chip has 53 qubits, so the "application program" is essentially a logic diagram of roughly 1,000 quantum gates. Surprisingly, a cogent quantum computer architecture is emerging that looks like a systolic array[4] or a solidly packed array of gates like a digital multiplier array. In this quantum architecture, the quantum gates naturally perform an entangling quantum operation on nearly every cycle, making the architecture hard to simulate on a supercomputer and ideal for demonstrating quantum supremacy.

DOI No. 10.1109/MC.2019.2958446
Date of current version: 12 February 2020

BASICS OF QUBIT SIMULATORS

Let's take a look at simulating a quantum computer on a supercomputer, because this is the central issue in assessing the advantage of the quantum approach. This discussion will focus on the full Schrödinger-state simulation, although there is a continuum of other methods that trade memory for computation. There are also fundamentally faster simulation methods that rely on specific properties of the quantum network, such as all of the gates being classical. The latter category is not helpful here.

Figure 1(a) shows a data structure for a group of $k = 3$ entangled qubits, where entanglement is the quantum property that yields the special computational capabilities. Each qubit will have the value of zero or one when read out. However, entanglement means the entire group will take on each specific pattern of $k = 3$ zeros and ones with a specific probability. Figure 1(a) represents the group by an array, a, of $2^k = 8$ rows, each row representing the pattern of zeros and ones via the binary representation of its index.

The value in each row of the array includes the probability that measuring the group of qubits returns the pattern of zeros and ones in the binary representation of the index. For a quantum simulator to duplicate unique quantum-information effects, it needs a second degree of freedom in each array entry, represented as ϕ. In lieu of a percentage and ϕ, a typical quantum simulator uses complex numbers to represent both degrees of freedom, but we don't need more detail on this second degree of freedom for this article. There is no reason to

store the indices, so the state of k qubits can be represented by an array of 2^k complex numbers.

Let's look at a two-input quantum gate, say between qubits q_m and q_n, $m = 0$, and $n = 2$ in Figure 1(a). Pick a row, perhaps row 2, and then flip the 2^m bit of the index, the 2^n bit, and both bits (highlighted in yellow). This will yield the group of four rows 2, 3, 6, and 7, as shown. Treat the four complex numbers in the array as a vector, multiply the vector by a 4 × 4 complex matrix that defines the gate, and store the result in place. Now repeat until every qubit has been updated once.

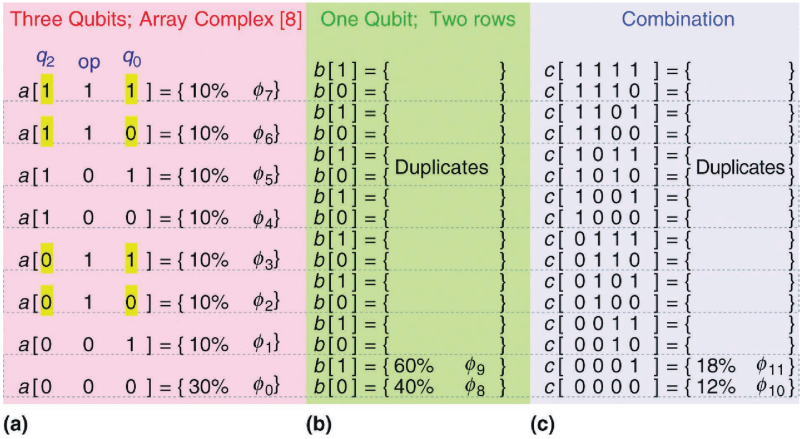

FIGURE 1. (a) A simulation array for three qubits containing 2^3 = 8 rows. Each entry can be considered to have a probability and phase or complex number. (b) A single qubit replicated eight times. (c) A four-qubit group, with c resulting from the combination of a and b.

In Figure 1(a), the two's bit is the only other one in the index, but Google's chip has 53 qubits, so there would be 2^{51} = 2×10^{15} additional updates. Thinking about Figure 1(a) reveals that a two-qubit operation involves a lot of memory and memory bandwidth, with the access pattern ranging from local to global depending on the location of the qubits in the index field.

To simulate a two-qubit gate on qubits in different groups, the groups have to be combined. Figure 2(b) shows a second array, b, with just one qubit, or two rows in an array containing its state. To create the state of all of the qubits combined, each row in array a is replaced with the entire array b, concatenating the bits of the index and multiplying the probabilities. This leads to an array whose size is the product of the sizes of the initial arrays, as illustrated in Figure 1(c). The largest quantum computer chips demonstrated today have 49, 53, 54, and 72 qubits, which would require arrays with 2^{49}, 2^{53}, 2^{54}, and 2^{72} entries, which is bigger than the main memory on today's largest supercomputers.

However, IBM's rebuttal to the quantum supremacy claim[3] analyzed the use of solid-state drive secondary storage to increase Summit's effective memory size. This required some additional gymnastics to circumvent the speed penalty for secondary storage. Essentially, the memory-access patterns in the previous discussions depend on how qubits are mapped to bits in the array indices. Programmers frequently manage the stride of vector access and working-set size to improve efficiency. In this case, memory is essentially a cache for the disk, and careful attention to the qubit-to-index bit mapping can affect the amount of input–output to the secondary storage.

THE GAME OF "SWAMP THE SIMULATOR"

To demonstrate quantum supremacy with the least hardware, the pure-play strategy is to design the quantum circuit so the simulator's data structures grow exponentially on every step, quickly swamping the simulator. Start with every qubit in its own group so that the simulator's arrays would have 2^1 = 2 rows. Apply a two-qubit gate to every even-numbered qubit

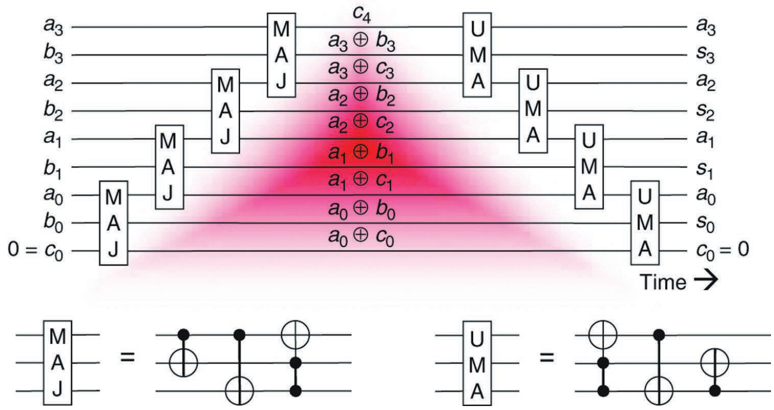

FIGURE 2. The quantum 4-bit ripple-carry adder. The circuit includes a ripple-carry chain on the left, but the circuit must uncompute the carry to maintain reversibility. The inverted "V" structure has a gap in the middle as the carry propagates, hampering the quantum speed-up. MAJ: majority; UMA: unmajority-add. (This image was derived from Cuccaro et al.[5])

and the odd-numbered one that follows it, halving the number of groups but increasing their size to $2^2 = 4$ rows. Then apply two-qubit gates to qubits in different groups, yielding four-qubit groups and squaring the array size to $2^4 = 16$ rows. Continuing this would lead to simulator arrays of 2^8, 2^{16}, and 2^{32} rows. The next combination would require arrays with 2^{64} rows, which is too big to be practical.

Unfortunately, the gate-connection pattern for the pure-play strategy is similar to a fast Fourier transform, requiring long wires and wires that cross over each other. Physical-science issues such as noise, the loss from long wires, and crosstalk have been analyzed theoretically, and the theory has been tested against results from real chips.[2] The architectural remedy is a quantum chip where the qubit interconnection pattern has just enough connectivity to enable complete entanglement before the qubits lose state. The 1D and 2D qubit arrays with nearest-neighbor interconnections would be scalable without long wires and crossovers, with 1D being simpler. However, the graph diameter of a 1D qubit array is too large, and today's qubits would lose state before entanglement could be established between all of them. So, the current designs are 2D.

QUANTUM SUPREMACY CREATES A BARRIER TO DOING USEFUL THINGS

The quantum ripple-carry adder in Figure 2[5] illustrates why a chip designed for a quantum-supremacy demonstration will have a hard time doing something useful. Essentially, computer architecture principles require idle time between gate operations, yet leaving quantum gates idle will hamper the game of swamp the simulator, so the quantum computer chip will have a harder time showing a performance advantage over supercomputers. Many proposed quantum algorithms use quantum numerical data types, such as integers, where the bits have been replaced by qubits. The algorithms make use of variables that are essentially a superposition of numeric values.

Readers familiar with a classical ripple-carry adder know that it consists of a series of stages, each with a carry-in, c_{in}; one bit of the a and b numeric inputs; a carry-out, c_{out}; and an s output with a bit of the sum. The quantum version in Figure 2 has $c_{in} = c_k$ on the lower left of each majority (MAJ) block and $c_{out} = c_{k+1}$ on its upper right. The initial $c_0 = 0$ means no carry, and the vertically propagating carry chain is essentially the same as a classical ripple-carry adder. The additional outputs of MAJ are $a_k \oplus c_k$ and $ak \oplus b_k$.

Quantum circuits must be reversible, so the carry signal, c_4, must be decomputed back to zero, which is accomplished by the top-to-bottom sequence of unmajority-add (UMA) blocks. The lower right of each UMA block outputs the reconstructed $c_{in} = c_k$ signal from $c_{out} = c_{k+1}$, $a_k \oplus c_k$, and $a_k \oplus b_k$. The UMA block also computes s_k, a bit of the sum, and provides a copy of a_k. The copy of a_k is also needed to make the

overall adder reversible. If the circuit is mirrored or run backward, it will decompute quantum integers *s* and *a* back to the original *a* and *b*, effectively implementing subtraction.

The problem is that the inverted "V" structure in Figure 2 has a lot of white space, illustrated in red, between the prongs of the V. This means that qubits (such as $a_0 \oplus b_0$) will be idle for many cycles, waiting for the carry signal to propagate to the most significant bit and back again.

Logic diagrams are normally sheets of paper with boxes on them representing the resources used in a design, but in the current state of quantum computing, white space on the paper counts as a resource as well. This is because white space represents the passage of time where the qubits will decohere but the simulator data structures will not grow, reducing the relative advantage of the quantum computer. Computer architects will recognize that logic diagrams for memories, multiplier arrays, systolic arrays, and so on can be drawn without white space, but many other functions require it, and there's not much that can be done about it.

One possible next step is to consider computational problems that are addressed by something short of a solid pack of quantum gates, which might look like a logic diagram with less than the normal amount of white space. Computer scientists and engineers understand that many important problems grow sparser at scale, but this is not universally true, and quantum computers have not reached the humongous scale of classical computers.

The following may be a bit of an oversimplification, but there is a proposed quantum computer benchmark called *quantum volume*,[6] which essentially quantifies the maximum number of gate operations that a quantum computer will execute. Google's recent work shows that 53 qubits are enough for quantum supremacy—or at least to create a lot of discussion about it. However, it assumes that all 53 gates work all of the time. If there were twice as many qubits, say 10^6, or double the runtime, the system ought to be able to demonstrate quantum supremacy for algorithms that have 50% white space in their schematic diagrams. The conventional wisdom is that chemical simulations will be the applications area that fits this next stage best, but I'm suggesting that the issue is really the architecture of the gates, not chemistry.

As quantum volume increases further, the range of problems that demonstrate quantum supremacy will grow. Eventually, quantum computers will just be computers and face many of the same issues that current computers face, like word and bus width, application programming interfaces, error logging, and so forth. I'm suggesting that the time is right for some computer scientists and engineers to consider these issues for quantum computation.

I'm not suggesting that other paths forward are unimportant. We need longer coherence times for qubits, quantum error correction, the random interconnect, control electronics, and the ability to adapt to problems as needed. Architecture goes hand in hand with these issues to move this area of computation forward.

REFERENCES

1. F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
2. A. Erhard et al., "Characterizing large-scale quantum computers via cycle benchmarking," *Nat Commun.*, vol. 10, p. 5347, Nov. 2019.
3. E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, Leveraging secondary storage to simulate deep 54-qubit sycamore circuits. 2019. [Online]. Available: arXiv:1910.09534
4. S. Y. Kung, *VLSI Array Processors*. Englewood Cliffs, NJ: Prentice Hall, 1988, p. 685.
5. S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. Petrie Moulton, A new quantum ripple-carry addition circuit. 2004. [Online]. Available: arXiv:quant-ph/0410184
6. A. W. Cross, L. Bishop, S. Sheldon, P. Nation, and J. Gambetta, "Validating quantum computers using randomized model circuits," *Phys. Rev. A*, vol. 100, no. 3, p. 032328, 2019.

ERIK P. DEBENEDICTIS is a principal with Zettaflops, LLC. He is the editor in chief of *IEEE Transactions on Quantum Engineering* and a volunteer supporting the IEEE Computer Society, Council on Superconductivity, and Quantum Initiative. Contact him at erikdebenedictis@gmail.com.

Conference Calendar

IEEE Computer Society conferences are valuable forums for learning on broad and dynamically shifting topics from within the computing profession. With over 200 conferences featuring leading experts and thought leaders, we have an event that is right for you. Questions? Contact conferences@computer.org.

AUGUST

1 August

- JCDL (ACM/IEEE Joint Conf. on Digital Libraries), virtual

14 August

- SmartIoT (IEEE Int'l Conf. on Smart Internet of Things), Beijing, China

16 August

- Hot Chips, Palo Alto, USA

17 August

- ACSOS (IEEE Int'l Conf. on Autonomic Computing and Self-Organizing Systems), Washington, DC, USA

19 August

- RTCSA (IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications), virtual

31 August

- RE (IEEE Int'l Requirements Eng. Conf.), Zurich, Switzerland

SEPTEMBER

7 September

- EuroS&P (IEEE European Symposium on Security & Privacy), virtual

14 September

- CLUSTER (IEEE Int'l Conf. on Cluster Computing), Kobe, Japan

21 September

- ASE (IEEE/ACM Int'l Conf. on Automated Software Eng.), Melbourne, Australia

24 September

- BigMM (IEEE Int'l Conf. on Multimedia Big Data), virtual

27 September

- ICSME (IEEE Int'l Conf. on Software Maintenance and Evolution), Adelaide, Australia

28 September

- SecDev (IEEE Secure Development), virtual

OCTOBER

3 October

- PACT (Int'l Conf. on Parallel Architectures and Compilation Techniques), Atlanta, USA

5 October

- EDOC (IEEE Int'l Enterprise Distributed Object Computing Conf.), Eindhoven, the Netherlands

12 October

- ISSRE (IEEE Int'l Symposium on Software Reliability Eng.), virtual

16 October

- ICEBE (IEEE Int'l Conf. on e-Business Eng.), Guangzhou, China

17 October

- MICRO (IEEE/ACM Int'l Symposium on Microarchitecture), Athens, Greece

18 October

- ICCD (IEEE Int'l Conf. on Computer Design), Hartford, USA
- MODELS (ACM/IEEE Int'l Conf. on Model-Driven Eng. Languages and Systems), Montreal, Canada

19 October

- DFT (IEEE Int'l Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems), Frascati, Italy

21 October

- FIE (IEEE Frontiers in Education Conf.), Uppsala, Sweden

25 October

- VIS (IEEE Visualization Conf.), Salt Lake City, USA

NOVEMBER

6 November

- CCEM (IEEE Int'l Conf. on Cloud Computing in Emerging Markets), Bengaluru, India
- SmartCloud (IEEE Int'l Conf. on Smart Cloud), Washington, DC, USA

9 November

- ICTAI (IEEE Int'l Conf. on Tools

with Artificial Intelligence), virtual
- IRC (IEEE Int'l Conf. on Robotic Computing), Taichung, Taiwan
- ISMVL (IEEE Int'l Symposium on Multiple-Valued Logic), Miyazaki, Japan

11 November
- SEC (IEEE/ACM Symposium on Edge Computing), San Jose, USA

15 November
- SC, Atlanta, USA

16 November
- FOCS (IEEE Symposium on Foundations of Computer Science), Durham, USA
- LCN (IEEE Conf. on Local Computer Networks), Sydney, Australia

29 November
- ICDCS (IEEE Int'l Conf. on Distributed Computing Systems), Singapore

30 November
- ICHI (IEEE Int'l Conf. on Healthcare Informatics), Oldenburg, Germany

DECEMBER

2 December
- CSDE (IEEE Asia-Pacific Conf. on Computer Science and Data Eng.), Gold Coast, Australia
- ISM (IEEE Int'l Symposium on Multimedia), Naples, Italy

6 December
- HOST (IEEE Int'l Symposium on Hardware-Oriented Security and Trust), San Jose, USA

7 December
- BDCAT (IEEE/ACM Int'l Conf. on Big Data Computing, Applications and Technologies), Leicester, UK
- UCC (IEEE/ACM Int'l Conf. on Utility and Cloud Computing), Leicester, UK

9 December
- CC (IEEE Int'l Conf. on Conversational Computing), Irvine, USA
- AIKE (IEEE Int'l Conf. on Artificial Intelligence and Knowledge Eng.), Irvine, USA

10 December
- BigData (IEEE Int'l Conf. on Big Data), virtual

14 December
- CloudCom (IEEE Int'l Conf. on Cloud Computing Technology and Science), Bangkok, Thailand
- HPCC (IEEE Int'l Conf. on High Performance Computing and Communications), Cuvu, Fiji

16 December
- BIBM (IEEE Int'l Conf. on Bioinformatics and Biomedicine), virtual
- HiPC (IEEE Int'l Conf. on High-Performance Computing, Data, and Analytics), virtual

29 December
- BigDataSE (IEEE Int'l Conf. on Big Data Science and Eng.), Guangzhou, China
- EUC (IEEE Int'l Conf. on Embedded and Ubiquitous Computing), Guangzhou, China

2021

JANUARY

5 January
- WACV (IEEE Winter Conf. on Applications of Computer Vision), Waikoloa, USA

17 January
- BigComp (IEEE Int'l Conf. on Big Data and Smart Computing), Bangkok, Thailand

27 January
- ICSC (IEEE Int'l Conf. on Semantic Computing), Laguna Hills, USA