VOLUME 10 ISSUE 3

The International Journal of

Design Management and Professional Practice

The Lens of the Lab

Design Challenges in Scientific Software

FRANCISCO QUEIROZ AND REJANE SPITZ

ASSOCIATE EDITORS
Articles published in *The International Journal of Design Management and Professional Practice* are peer reviewed by scholars who are active participants of the Design Principles and Practices Journal Collection or a thematically related Knowledge Community. Reviewers are acknowledged as Associate Editors in the corresponding volume of the journal. For a full list, of past and current Associate Editors please visit www.designprinciplesandpractices.com/journals/editors.

THE INTERNATIONAL JOURNAL OF DESIGN MANAGEMENT AND PROFESSIONAL PRACTICE
www.designprinciplesandpractices.com
ISSN: 2325-162X (Print)
ISSN: 2325-1638 (Online)
doi:10.18848/2325-162X/CGP (Journal)

First published by Common Ground Publishing in 2016
University of Illinois Research Park
2001 South First Street, Suite 202
Champaign, IL 61820 USA
Ph: +1-217-328-0405
www.commongroundpublishing.com

The International Journal of Design Management and Professional Practice is a peer-reviewed, scholarly journal.

ARTICLE SUBMISSION
The International Journal of Design Management and Professional Practice publishes quarterly (March, June, September, December). To find out more about the submission process, please visit http://www.designprinciplesandpractices.com/journals/call-for-papers.

ABSTRACTING AND INDEXING
For a full list of databases in which this journal is indexed, please visit www.designprinciplesandpractices.com/journals/collection.

KNOWLEDGE COMMUNITY MEMBERSHIP
Authors in *The International Journal of Design Management and Professional Practice* are members of the Design Principles and Practices Journal Collection or a thematically related Knowledge Community. Members receive access to journal content.
To find out more, visit
www.designprinciplesandpractices.com/about/become-a-member.

SUBSCRIPTIONS
The International Journal of Design Management and Professional Practice is available in electronic and print formats. Subscribe to gain access to content from the current year and the entire backlist.
Contact us at cg-support@commongroundpublishing.com.

ORDERING
Single articles and issues are available from the journal bookstore at www.ijg.cgpublisher.com.

HYBRID OPEN ACCESS
The International Journal of Design Management and Professional Practice is Hybrid Open Access, meaning authors can choose to make their articles open access. This allows their work to reach an even wider audience, broadening the dissemination of their research. To find out more, please visit
www.designprinciplesandpractices.com/journals/hybrid-open-access.

The Lens of the Lab: Design Challenges in Scientific Software

Francisco Queiroz, PUC-Rio, Brazil
Rejane Spitz, PUC-Rio, Brazil

Abstract: Playful and gameful design could improve the quality of scientific software. However, literature about gamification methods for that particular type of software is presently scarce. As an effort to fill that gap, this paper introduces a set of design challenges and opportunities that should be informative to professionals approaching the area. This research is based on literature review on scientific software development, also contemplating material on the gamification of science, software, and work. From the gathered information, we identify, map, and discuss key aspects of development and use of professional scientific software. Those findings are, then, formatted as a Design Lens—a set of questions designers should ask themselves to gain insight, from a particular perspective, on their work. We propose the Lens of the Lab as a design lens to support designers working in collaboration with scientists and software engineers in professional scientific software initiatives.

Keywords: Specification, Gamification, Scientific Software

Introduction

It has been suggested that gamification should be applied to scientific software as a way to improve user experience (Wolff 2015). This paper investigates scientific software characteristics, focusing on usability, to propose a set of principles organized as a design lens. Originally concerned about videogame-inspired experiences, the lens could be useful to UI/UX designers approaching scientific software.

Definitions

Before proposing a design lens to facilitate gamification and playful design of scientific software, we must introduce those terms. *Scientific software* is defined by three characteristics: (1) it is developed to answer a scientific question; (2) it relies on the close involvement of an expert in its scientific domain; and (3) it provides data to be examined by the person who will answer that question (Kelly 2013). In the light of the recent trend of citizen science, we should clarify that this paper is primarily concerned with software "developed by scientists for scientists" (Sletholt et al. 2012, 24)—even if aware and informed by gamified citizen science.

Design lenses were first elaborated by Jesse Schell (2015) as a way of expressing principles that should inform design decisions. They are usually constituted by a brief explanation of a topic, followed by a set of questions. Originally developed for game design, they have been expanded to user experience (Scott 2010) and gamification practices (Deterding 2013).

Gamification, "the use of game design elements in non-game contexts" (Deterding et al. 2011, 2), is typically associated with structured play (e.g., games) and *gameful* behavior (McGonigal 2011). It is opposite and complimentary to *playful design*, based on unstructured play (e.g., toys) (Deterding et al. 2011). For all practical purposes, this study embraces the whole spectrum from games to toys, *ludus* to *paidia* (Caillois 2001), *toyplay* to *goalplay* (Madsen et al. 2007), focusing on videogames as inspiration for better software.

The International Journal of Design Management and Professional Practice
Volume 10, Issue 3, 2016, www.designprinciplesandpractices.com

ISSN: 2325-162X (Print), ISSN: 2325-1638 (Online)

Methodology

Throughout the next subsections, we describe the semi-systematic literature review (SSLR) on which this study is based.

Issues and Concerns

Through this SSLR, we were interested in investigating:

- How is scientific software developed and used?
- How are user interfaces for scientific software designed?
- Which guidelines and case studies can be used as reference?
- How can scientific software be gamified?

Sources, Queries, and Search Results

We have selected the following databases for a thorough search: ACM Digital Library, IEEE Explore, Scopus, AIS Electronic Library, and Web of Knowledge. Aiming for a comprehensive result, searches for the following terms were performed:

(1) "scientific software" and "gamification"
(2) "citizen science" and "gamification"
(3) "scientific software" and "user experience"
(4) "scientific software" and "user interface"
(5) "scientific software" and "HCI"
(6) "scientific software development"

After excluding duplicates and unrelated material, we have reached the number of 270 references. The number of results for each query by database, followed by the number of selected articles after screening and deduplication, are shown in Table 1.

Table 1: Search Results for Queries

	ACM DL	*IEEE Exp.*	*Scopus*	*AIS EL*	*WoK*	*Total*
"scientific software" and "gamification"	0	1	0	1	0	1
"citizen science" and "gamification"	8	17	11	5	3	21
"scientific software" and "user experience"	0	17	4	9	1	6
"scientific software" and "user interface"	2	201	42	55	10	132
"scientific software" and "HCI"	2	11	0	7	0	5
"scientific software development"	13	140	81	62	54	105
Total	25	387	138	139	68	270

Source: Data updated from selected databases on December 4, 2015.

Further Refinement and Additional Material

We have included articles on related subjects, found through sources such as Science Direct, Google Scholar, and gamification-research.org, as a way to acknowledge significant contributions otherwise eluded by the initial databases. Also, we have refined our selection through further screening, reaching the number of 221 unique references.

Findings

Through the next subsections, we present the most relevant findings collected.

How Is Scientific Software Developed and Used?

Scientific software (SS) has gained importance in the last three decades, moving science "from test tubes into silicon-based simulation" (Woollard et al. 2008, 38). SS is used for "processing, analyzing, visualizing, managing, sharing, experimenting and […] generating new raw data" (Ahmed and Zeeshan 2014, 55), and allows the conduction of research in otherwise impossible conditions (Segal and Morris 2008). A testbed for new technologies (Mills et al. 1995), SS connects "abstract theoretical and real industrial worlds" (Prego and Seisdedos 2011, 1). It can be used to control field equipment (Mielke et al. 2005) or geographically distributed lab facilities (Gertz, Stewart, and Khosla 1994). SS is diverse in scope and size: it could mean a single software library, a plugin, or a fully-fledged software. It could be built from scratch or as a module for third-party solutions (Frank et al. 2007).

A survey from 2009 claims that scientists who use SS spend on average 30 percent of their working time developing and 40 percent using it. Developing their own software is important or very important to 84.3 percent of those who use it. Additionally, SS developed by scientists might be important to other researchers (Hannay et al. 2009). Recently, a poll indicated that seven out of ten UK researchers consider research without software an impossibility (Hettrick 2014). Moreover, although only a minority uses SS, a vast majority benefits from the advancements it brings (Kelly and Skordaki 2015). Very often, SS is built for the developer himself or groups under ten people—although there is a significant number of projects intended for larger groups (over 100 people), and cases of SS reaching over 5,000 users—mostly the case for open-source or commercial packages (Hannay et al. 2009; Nguyen-Hoan, Flint, and Sankaranarayana 2010). Reasons behind development are usually (1) first-hand research; (2) training and education; (3) external decision support (Sanders 2008, 36). A survey from 2010 claims that most projects are developed within universities—followed by industry, research, and government—often by small teams comprising one to six members (Nguyen-Hoan, Flint, and Sankaranarayana 2010).

The scientific and academic nature of SS is responsible for its most pronounced characteristics, needs, and challenges. First, there is *motivation*: unlike in most development environments, the main goal is not making software, but science (Basili et al. 2008). As such, professional reputation comes from publishing papers—and scientists can perceive software exclusively as a means to that. This attitude could hurt software quality (Killcoyne and Boyle 2009), demotivate developers, and harm their collaboration with users/scientists (Howison and Herbsleb 2011), who could fail to engage into the development process (Segal 2009). In fact, *collaboration* is vital, yet potentially problematic, since it can be undermined by a personal sense of authorship (Turk 2014). Team communication is essential (Morris and Segal 2012; Taweel et al. 2009), as members might come from different backgrounds, work across the globe (Marinovici, Kirkham, and Glass 2014) for extended periods of time, and hold particular visions for the project (Spencer 2015). Ideally, the software itself should support collaboration, facilitating data integration, content sharing, documentation, workflow, and knowledge management (Pinto et al. 2002). Additionally, supporting collaborative communities for know-how sharing can stimulate software use and adoption (De Roure and Goble 2009).

Communities for free and open SS resources have been fostering collaboration for a long time, evolving from personal memberships in the 1950s to global communities, through e-mail, Arpanet, and Internet servers, during the 1980s and 90s (Dongarra et al. 2008). In the last ten years, new types of massively collaborative efforts, such as open data, crowd science, and citizen science, have become a new paradigm (Vanschoren et al. 2014). Crowd science disrupts from "traditional" science in two ways: openness to a wide base of contributors (up to hundreds of thousands), and disclosure of sensitive information (e.g.: data, algorithms) normally restricted to scientists undertaking the research. It defies the traditional model of closed research groups competing for publication impact by suggesting new ways of collective authorship. As such, it could influence how science is traditionally made regarding collaboration, motivation, knowledge creation, openness, transparency, organization of teams, the embedding of scientific processes, and rigor in software (Franzoni and Sauermann 2014). In many cases, crowd science is gamified—a relationship we investigate in a further subsection.

SS often depends on the collaboration between scientists and software engineers (Holthouse and Greenberg 1978). Often, scientists who are technically apt might develop software themselves. Segal (2007) refers to them as the "professional end-user developer," for whom software development is a secondary activity, and who develops alone or along other scientists and programmers, for himself or his community of practice. As a result, SS development culture is often described as averse to software engineering best practices, informal, with no clear requirements or design (Ahmed and Zeeshan 2014). This lack of planning could affect development and commercialization (Sanders 2008, 40) and is inadequate to complex systems, although sometimes sufficient for punctual research (Spencer 2015). Kelly (2007) identifies a "chasm" between software engineering (SE) professionals (who lack domain-specific knowledge) and scientific computing (SC) community members (who lack SE knowledge) communities. Developers might downplay initiatives to improve development process, failing to acknowledge room for improvement—in which case advocates for solutions might have to demonstrate potential gain by joining the development process (Brebner 1998). Focus on SE practices might be perceived as bureaucratic and demotivate scientists (Spencer 2015). On the other hand, both SC and SE communities agree that SS quality could be increased by the adoption of SE practices (Mohammad 2010); Heaton and Carver 2015), including well-specified requirements, automated testing, and documentation (Koteska and Mishev 2013). Likewise, it would be advisable for software engineers to gain knowledge on the scientific domain in question (Marinovici, Kirkham and Glass 2014), and on how scientists work and report their progress (Killcoyne and Boyle 2009).

The "professional end-user developer" mindset is not the only obstacle between SS and SE practices: scientific computing offers specific challenges—starting with the difficulty in establishing *requirements*. Brooks stated that "the hardest single part of building software is deciding precisely what to build" (Brooks apud Ovaska, Rossi, and Smolander 2005, 32). That is particularly true to SS, as scientific theories behind the software might change throughout research, forcing requirements to change along development. Requirements are, then, understood along the process (Segal 2007), "as the software and the concomitant understanding of the domain progress" (Segal and Morris 2008, 18). Requirements are often undocumented, although some fields require formal documentation—in some cases using UML (Daniluk 2012), low and high-fidelity prototypes (Aragon et al. 2008), or interactive storyboards (Sanders 2008). Informally asking users for requirements is often unproductive (Morris and Segal 2012) and, ideally, requirements should be consolidated as technical, clear descriptions of specific features (Marinovici, Kirkham, and Glass 2014). In some cases, requirements can be established through the use of personas (Schneidewind et al. 2012).

Another challenge is *longevity*: SS can be developed, used, and maintained for a long time, often decades (Basili et al. 2008; Sanders 2008; Kelly 2009). For that reason, it might require refactoring (Heaton and Carver 2015; Spencer 2015), and should be planned for sustainability (Morris and Segal 2012) and *expandability*, since it could grow beyond initially planned (Basili et al. 2008). Keeping software simple and avoiding unnecessary features is advisable (Gorton 2013). In some cases, new features are better incorporated via "modules" added to the main software (Sanders 2008). On the other hand, there are cases where software life is expected to be short and the software itself, disposable (Segal and Morris 2011).

To face such challenges, SS is usually developed in an *iterative* fashion (Sanders 2008). Performing *incremental* changes is a best practice for scientific computing (Ackroyd et al. 2008; Wilson et al. 2014), since regular iterations and small releases "let users influence development and form requirements" (Ackroyd et al. 2008, 46). From a design perspective, responding to user's needs and having a user-base in mind can be more effective than striving for a complete solution (De Roure and Goble 2009). Adopting development methodologies can be problematic: this iterative nature makes it somewhat incompatible with traditional Waterfall development methodology (Holthouse and Greenberg 1978). Agile seems to be more appropriate (Crabtree et al. 2009; Nantaamornphong et al. 2014; Ahalt et al. 2014). However, SS communities can be selective in the adoption of Agile practices (Sletholt et al. 2012), some of which can be unfeasible to apply to scientists (Kelly and Smith 2010). Instead of embracing an established methodology, developers might adopt a "loose version(s)" (Basili et al. 2008, 35) or a few selected practices (Nanthaamornphong and Carver 2015). Sometimes, developers decide for an "amethodological" approach (Kelly 2015). Methodologies tailored to SS have been outlined a number of times (Platz 1986; Pereira Junior 2007; Ahmed, Zeeshan, and Dandekar 2014), often emphasizing neglected (or difficult) aspects such as requirements, testing, and design (Cort et al. 1985). As early as 1986, Platz (1986) observed the need to balance SE measures with creative freedom.

SS has a number of scientific needs: *Correctness*, a "core value of science" (Howison and Herbsleb 2010, 3), stands as its most important quality and primary concern, beyond usability (Kelly and Sanders 2008a). *Reproducibility* is also essential for the scientific method (Cimiano and Sagerer 2015), allowing for validation and peer review (Recio-Garcia, Diaz-Agudo, and González-Calero 2013), and is often addressed through portability and process automation. Additionally, SS must handle *complex data* (Keffe 2010, 8) at great levels of *precision* (Hatton and Roberts 1994). This complexity often requires *interoperability* with third-party software and external resources (Fdez-Riverola et al. 2012), from other scientific software, industrial, and robotic systems (Picón et al. 2006), to productivity software such as spreadsheet editors (Trlica 1997; McKiney 2003). Finally, *testing* and validation can be challenging, since comparing results to real-world data is unfeasible (Segal and Morris 2008; Heaton and Carver 2015) and code testing requires scientific domain knowledge or the help of a scientist. (Kelly, Thorsteinson, and Hook 2011).

SS can be developed for desktop computers, workstations (Hannay et al. 2009), and portable devices (Hughes et al. 2004; Clark 2014). Sometimes, SS demands *High Performance Computing* (HPC) achieved through resources such as supercomputers (Hatton and Roberts 1994; Hannay et al. 2009), mainframes (Chen and Fu 1996), clusters (Cohen et al. 2013), grids (Choi et al. 2006; Frank et al. 2007), and cloud services (Hou et al. 2010; Church et al. 2012; Mendez, Villamiazr, and Castro 2013). HPC resources can be structurally complex (Kovalchuk et al. 2012), and are usually accessed through gateways via desktop or web tools (Ernst et al. 2003; Kolberg, Courivaud, and Özbek 2007; Gomes et al. 2015). Giving access to top computational power to users of low-end systems has been a concern for decades, and ensuring similar user experience for all users was once a much bigger challenge (Atkins and Phillips 1986; Phillips et al. 1986). In that respect, web and cloud-based services democratize HPC (Afgan et al. 2015), allowing scientists to focus on scientific problems, not computational ones (Bastos, Moreira, and Gomes 2013).

Skeptical on new technologies, SS developers are likely to adopt technologies that are compatible with old ones and tend to start things from scratch instead of adopting preexisting frameworks (Basili et al. 2008). Late adoption of technological trends could be attributed it to "high complexity and narrowly defined market segments" (Clark 2014, 237). This tendency reflects on how software is used: establishing new practices can be challenging due to preexisting work culture (Morris and Segal 2012). In such case, benefits should be clear from the start, and users should not be forced to change how they work (De Roure and Goble 2009). SS can be developed in various languages such as Fortran, Binary, Assembler, C, C++, Java, Python, PHP, MatLab, etc. (Nguyen-Hoan, Flint, and Sankaranarayana 2010); Ahmed and Zeeshan 2014). It can be developed in multiple languages (Basili et al. 2008) or assisted by domain-specific frameworks (Glez-Peña et al. 2010; Fdez-Riverola et al. 2012). Allowing scientists to do their own programming can be necessary (Wang, Tran and Andrade 2010). In this case, visual programming (VP) and domain-specific languages (DSLs) can be alternatives to low-level programming (Jones and Scaffidi 2011). Simpler and less error-prone than low-level languages (Hinsen 2013), DSLs are available as closed-source or open-source solutions (Papadimitriou et al. 2009). VP, on the other hand, allows scientists to construct scientific models and equations by dragging and connecting components as in a CAD/CAM environment resembling data flow diagrams (Keller and Rimmon 1992; Rijnders, Spoelder and Groen 1993). An example of such environment is LabVIEW, which enables lab automation and simulation combining data-flow, building blocks, virtual instruments, and typed programming language, (Kodosky, MacCrisken, and Rymar 1991; Wang, Tran and Andrade 2010). VP usability makes scientists more productive—unless computational performance is affected (Chambers 2013). Eventually, loss of performance can be overcome: visual elements could be rewritten in DSL or recompiled as machine language (Bilmes 1996). Another possibility is teaching users to improve their programming through the software itself (Chambers 2013). As an additional benefit, allowing users to develop and incorporate their own solutions can add value to the software (De Roure and Goble 2009).

Work in SS, especially when done through networked resources, can be divided in three phases: modeling, simulation, and result analysis (Kovalchuk et al. 2012, 2)—which are not necessarily attained through a single software. First, during the *modeling* phase, the scientific problem is represented in mathematical terms (Li 2011, 42), designed, and coded to represent the system during simulation (Daniluk 2012).

Once ready, models can be used in the *simulation*, where data is entered, submitted, and processed by a series of computational tasks. Dependencies and connections between these individual tasks, from start to finish, compose a *workflow* (Woollard et al. 2008). Workflows can be automated through scripting, visual editors, or specialized frameworks (Vigder et al. 2008; De Roure and Goble 2009; Silva 2010). They can involve complex processing, possibly requiring data to go sequentially through a series of different software, possibly taking days to compute. To minimize the risk of failure, it has been suggested that workflows should be broken down in smaller steps and use checkpoints systems (Gross et al. 2008; Nguyên, Trifan, and Désidéri 2011). In some cases, workflows can be saved as templates, exported, and shared for reuse (Bergmann, Demuth, and Sander 2011).

Finally, after simulation, output data should be ready for the *result analysis* phase. That is the culmination of the work in SS, since "the purpose of computing is insight, not numbers" (Hamming apud Kelly and Sanders 2008a, 3). Result analysis often relies on data visualization and manipulation—topics which will be explored in the next subsection. Scientific findings derived from analysis could be used to advance research further or disclosed in publications.

How Are User Interfaces for Scientific Software Designed?

HCI is often regarded as "the most ignored and unattended phase of scientific software solution development" (Ahmed, Zeeshan, and Dandekar 2014, 6), in many ways reflecting the "professional end-user developer" mindset. Developer's proximity to the domain can be a complication, as they assume that users with similar backgrounds or programming skills will understand the software, thus neglecting testing or documentation (Kelly and Sanders 2008b; Sanders 2008; Nguyen-Hoan, Flint, and Sankaranarayana 2010). Mainly concerned with the computational engine (Sanders 2008), developers might believe that "almost any user interface will be tolerated" (MacLeod, Johnson, and Matheson 1992, 415). Indeed, SS usability can be challenging right from installation and setup (Geimer, Hoste, and McLay 2014), potentially obfuscating functionalities (Papadimitriou et al. 2011). Additionally, creating GUIs requires specialized skills, and is possibly "the last thing a scientist wants to deal with" (Lundstrom and Klimeck 2006, 497). Developers are not the only responsible group for usability issues: sometimes, stakeholders financing the software might misunderstand user's needs (Morris and Segal 2012), or resist investing in design research, since its influence on insight is difficult to demonstrate (De Matos et al. 2013).

Neglecting usability presents a major risk, especially when software grows in complexity (Kendall et al. 2007), or is developed for external use (Sanders 2008). Applications could be rejected due to difficult user interfaces and lack of visual output (Ahmed and Zeeshan 2014). Conversely, user-friendly interfaces can be a major reason for adoption (Joppa et al. 2013), even overcoming "apparent lack of performance" (Manjunatha et al. 2011, 4). Sometimes, documentation is seen as a substitute for usability. However, SS documentation is often incomplete, generated based on user's demand and feedback, or created by users themselves (Pawlik et al. 2012). Also, it can be unpractical to access. Interestingly, a developer expressed his desire of keeping his software's manual "to the slim size of an average video game manual" (Sanders 2008, 69).

Aware of such risks, some developers adopt good practices such as: continuously testing user interfaces; elaborating storyboards; observing field work; testing software in controlled environments; adding help systems (Sanders and Kelly 2008); and planning usability cases (La Rue et al. 2014). The perceived importance of usability in SS seems to be increasing in some areas (Eliceiri et al. 2012). Recent approaches to SS development put interface design as a priority by valuing the interface designer's perspective (Mohammad 2010). The Butterfly development model, for instance, pays attention to interface design from early development stages, aiming at ease of use and learning by understanding user psychology, scientific domains, work environment, and HCI principles and patterns (Ahmed, Zeeshan, Dandekar 2014). The recent proposition for a "tool science" discipline, dedicated to SS development and teaching, recommends the improvement of usability through case study research, comparison of similar tools, and gamification. It also advocates that software quality reflects development and usage aspects combined, including basic functionality, good development practices, efficiency and effectiveness, and fun in use (Wolff 2015).

In many cases, GUIs are not essential (Chen and Fu 1996; Jarvis et al. 2006): SS can start as libraries accessed through command line and, later, receive a GUI—sometimes through third-party software integration (Chancelier, Lapeyre, and Lelong 2014). In fact, users might find convenient to bypass GUIs and use text-based interfaces (Joshi et al. 1997) or enter data programmatically (Weerawarana et al. 1996). GUI code should be kept separate from scientific calculations (Kelly, Hook, and Sanders 2009), allowing for easier customization and reconfiguration (Bastos, Moreira, and Gomes 2013). Nevertheless, GUIs can make SS user-friendly (Murphy 1996; Belsky et al. 2002), especially to users without computing background (Cohen et al. 2013) or in the case of complex software (Versek 2013). GUIs can mediate user's "visual and cognitive processes and the computer's numerous low-level calculations" (Foulser and Gropp 1990, 22), providing "a high-level abstraction of the underlying computational facilities" (Ramakrishnan and Rice 1996, 2). A particular class of SS that embraces the importance of GUIs is the Problem Solving Environment (PSE). PSEs are "computer system[s] that provides all the computational facilities to solve a target class of problems […] by communicating in the user's own terms" (Gallopoulos, Houstis, and Rice 1999, 7). In simpler terms, "PSE = user interface + libraries + knowledge base + integration" (Rice and Boisvert 1996, 47). PSEs do not require specialized programming expertise by the user: problems are formulated, simulated, solved, and displayed with the assistance of high-level user interfaces (Houstis and Rice 2002).

SS offers specific HCI challenges depending on which stage of the scientific process is being supported. The *modeling* phase, for instance, might require combined knowledge on scientific domain, mathematical modeling, and programming—needs that can foster interesting solutions: modeling could be constructed through flow-diagram interfaces (Julvez, Matcovschi, and Pastravanu 2014; Bunus 2006) and informed by a display of domain-specific knowledge base (Keller, Michal, and Das 1994). Additionally, GUIs could provide clearer understanding of scientific concepts behind programming by reinterpreting selected sections of code as scientific terms, along with an appropriate glossary (Stewart 2001).

The *simulation* phase could require tools for construction and execution of workflows, which could be created, configured, submitted, and monitored through GUIs (Bergmann, Demuth, and Sander 2011) for ease of use and increased productivity (Dong and Wild 2008; Vigder et al. 2008); De Roure and Goble 2009). As with models, workflows are often edited through data-flow diagrams—in this case, connecting sequences of computational tasks and services through which data will be transformed.

The *result analysis* phase is often highly visual, demanding GUIs for visualization and manipulation of output data. That could take shape as visual representations of the object under study, such as crystal structures (Belsky et al. 2002), geographic terrain (Feibush, Gagvani, and Williams 2000), or medical imaging (Bitter et al. 2007). In other cases, it could involve methods for data visualization such as charts, 2D and 3D graphs (Willson, Whitham, and Anderson 1992; Murali, Dutta, and Biswas 1993; Curtis, Kinnaird, and Xing 2011; Ellis et al. 2013). Finding ways of representing knowledge is important (Ramakrishnan and Rice 1996): they can enhance and facilitate insight (Kornbluh 1993), since "the human brain works better with pictures than strings and characters" (Ntombela et al. 2005, 247). Occasionally, however, search commands and formulas can be more effective and convenient than image-based approaches when analyzing large datasets (Springmeyer 1993).

Result analysis depends on how data is viewed, navigated, and configured by the user. Desirable features depend on context, and can include single or multiple views of data tables (Fischer et al. 2010); descriptive text; 3D, 2D and mixed representations of the subject (Bitter et al. 2007; Verigan 2007); static, dynamic, or interactive charts (Murali, Dutta, and Biswas 1993); control schemes for 3D and 2D viewport navigation (Bitter et al. 2007); 3D object examination and manipulation (Hu and Lill 2014); facilities for observing evolution over time (Springmeyer 1993, Bitter et al. 2007, Eliceiri et al. 2012, Terranova and Magni 2012); 3D scenes featuring complex geometry and materials (Feibush, Gagvani, and Williams 2000); Rigged 3D humanoid figures (Fourquet, Hue, and Chiron 2007); Image manipulation tools, access to 3D visualization properties (MacLeod, Johnson and Matheson 1992; Fischer et al. 2010). The ability of exporting images for the generation of reports can be an advantage (Springmeyer 1993).

Regarding the division of the scientific process (modeling, simulation, and analysis), supporting multiple stages could influence GUI construction (Bunus 2006). The PN Toolbox, for instance, is a Matlab extension featuring two GUI modes: drawing (for modeling) and exploring (for simulation and analysis). A toggle switch triggers subtle changes to the GUI, enabling appropriate tools to the selected phase, and disabling those which are not (Julvez, Matcovschi, and Pastravanu 2014). Another illustrative case is "Biok," a "programmable graphical application for biologists" where graphs, protein sequences, and 3D visualizations can have their content and methods modified, via programming, by users (Letondal 2006, 15).

SS can have specific HCI requirements derived from its scientific nature. Although having an up-front design phase contemplating a user-friendly interface is considered a best practice (Baxter et al. 2006), UI could have to adapt to *emerging requirements*, frequent changes in source code (Lande 2008), and software growth. This need for flexibility can lead to creative solutions for the quick addition of functionalities (MacLeod, Johnson and Matheson 1992). The complexity of data in SS might require ways of *entering and analyzing a high number of parameters* (Fdez-Riverola et al. 2012). The need for correctness could demand input methods tailored for *precision* (MacLeod, Johnson and Matheson 1992). Also, the production of adequate design artifacts could require *collaboration* between designers and scientists (Chen, Zhang, and Vogeley 2009).

Professional needs might also influence HCI design: SS might allow for different work modes depending on *user specialization* (Gertz, Stewart, and Khosla 1994; Javahery, Seffah and Radhakrishnan 2006). It could feature a dedicated toolbar for frequently used commands (Julvez, Matcovschi, and Pastravanu 2014). The use artificial intelligence to generate contextualized UI seems to be unusual, although it has been an area of investigation for some time (Brouwer-Janse 1990). Depending on the nature of work, visualization should support critical analysis of data under time pressure (Aragon et al. 2008), or provide clear indication of system malfunctioning (Morais et al. 2014). In some cases, for the sake of familiarity, GUIs might attempt to emulate functions *and* looks of physical instruments (Foster 1998).

Specialized frameworks can accelerate GUI development and help research groups to achieve better usability (Glez-Peña et al. 2010; Fdez-Riverola et al. 2012). In some cases, GUI solutions are designed with customization and expansion in mind, as a way to stimulate use and further development by other researchers (La Rue et al. 2014). Conversely, the adoption of open source solutions might allow the customization of preexisting UI to better suit requirements (Gorton et al. 2012).

Since SS is more often used in personal computers (Hannay et al. 2009), GUIs are typically based on the Windows-Icon-Mouse-Pointer (WIMP) paradigm—usually supporting simultaneous windows display (McFaddin and Rice 1992) and drag-and-drop functionalities (Manjunatha et al. 2011). The web is also a popular platform: web-based GUIs have been used since, at least, two decades ago (Chen and Fu 1996). In the past, transitioning from desktop to web applications meant tradeoffs and compromises in UI (Takatsuka and Gahegan 2001). Today, web interfaces can replicate environments as diverse as command line prompts (Choi et al. 2006), standalone desktop software (Yamazaki et al. 2011), websites and wiki platforms (Gorton et al. 2012), and "app stores" (Skidmore et al. 2011)—which helps developers to integrate different products (Turk 2015). Web interfaces can be expandable (Brookes et al. 2015) and adaptable to different screen sizes—allowing for better mobile experiences (Yamazaki et al. 2011). Web-based apps can also eliminate installation and compatibility issues, since the software runs on a server. Sometimes, web interfaces can be dynamically built (Wauer et al. 2004).

Finally, SS might demand particular technological devices and capabilities. High-quality graphics could be necessary for readability and immersive visualization (Feibush, Gagvani, and Williams 2000; Kovalchuk et al. 2012). In some cases, allowing users to balance visual quality and performance is advisable (Fischer et al. 2010). The use of SS could require equipment beyond traditional desktop computers, including joysticks, speech control, stereographic displays (Feibush, Gagvani, and Williams 2000); multi-touch screens, mobile devices, motion capture equipment (Fourquet, Hue, and Chiron 2007); real-time surveillance cameras (Gertz, Stewart, and Khosla 1994); tablet PCs, trackballs, 3D force-feedback controllers (Keefe 2010) and so on. Combining different platforms and technologies could be necessary: simulations could be diagrammed in web tools, executed in HPC resources, and experienced through stereoscopic VR (Kovalchuk et al. 2012). Hybrid interfaces, combining desktop PCs and immersive environments, could allow for 2D or 3D operations, depending on context (De Carvalho et al. 2009).

Which Guidelines and Case Studies Can Be Used as Reference?

In spite of the lack of attention to HCI attributed to SS, we could identify a number of guidelines and case studies that should be acknowledged for their informative research on usability design and research methods. Cherri Pancake identifies ten usability objectives divided into four dimensions: Ease of learning; Ease of use; Usefulness; and Throughput (efficiency). Her work emphasizes the need for user-centered, consistent, intuitive, and minimalistic design capable of preventing and fixing user errors, streamlining operations, and increasing productivity. She suggests a four-step model for user-centered design consisting of: Identifying software requirements based on user needs; Understanding how users work within their environment; Designing incrementally; and Performing user testing (Pancake 1996).

Daniel Keefe (2010, 8) highlights two important HCI needs in SS: handling "complex data" and "complex analysis tasks defined by specialized, motivated users." Keefe also emphasizes the importance of tool evaluation from scientific domain experts; the need to prioritize accuracy over speed of use; and the presence of interaction requirements such as selection, navigation, and annotation.

Ahmed, Zeeshan, and Dandekar (2014) argue that interface design should be guided by HCI principles of experimentation, contextualization, iteration, and empirical measurement. Their study proposes a comprehensive design process including the production of mockups, brainstorming, and prototyping informed by HCI design patterns (Ahmed, Zeeshan, and Dandekar 2014). Attention to scientific domain, work environment, and user's IT background are also emphasized.

Case studies unanimously stress the need for understanding the scientific work and its tasks. Rebecca Springmeyer (1993) has applied the "designer-as-apprentice" methodology for designing the MDC tool for data analysis. Her methods included field observation and user interviews to design around the way scientists worked.

The need for ethnographic research was also reinforced by developers of "Making Tea," a pervasive, Tablet PC-based version of the lab book (Hughes et al. 2004). Their research emphasizes the importance of consulting field experts for understanding their work best practices. The end product was evaluated under the *Process, Outcome, Affect* model, in which users describe their performance, results, and if they felt "empowered" by the new system (Hughes et al. 2004). A separate study on electronic laboratory notebooks was performed by Talbott et al. (2005), who stressed the need for collaboration, annotation, and access to metadata.

Methodologies and principles for user-centered design are also discussed by Javahery, Seffah and Radhakrishnan (2006). Again, the need for understanding how scientists operate (including field observation of work with other tools) is emphasized as a fundamental step, followed by prototyping, usability studies, and heuristic evaluation.

Design research methods for the OMERO imaging software were extensively documented (Macaulay et al. 2009; Sloan et al. 2009; Loynton et al. 2009) and included testing sessions, design workshops, demonstrations, surveys, design research, heuristics evaluation, and training material. The study also emphasizes the need to understand how scientists work and how labs function, find ways of setting priorities and manage user's expectations.

The Enzyme Portal (De Matos et al. 2013), developed through user-centered design, employed the following methods: Requirements eliciting; personae creation; user interviews; workflow analysis; workshops with experts; analysis of workshops; paper prototyping and testing; technical specification; and interactive prototyping. Identified challenges included: balancing needs of both computational and lab-based communities; measuring insight levels; establishing standards; presenting data and metadata; and finding individuals with combined knowledge on HCI, computing, and scientific domain. Overall, investigated material points out the need for designing around how SS is developed and used, offering insight on how to identify and address those needs through a careful design process.

How Can Scientific Software Be Gamified?

The idea of making scientific software more similar to games is hardly recent: back in 2000, Houstis and Rice predicted that, by this current decade, PSEs would resemble simulators and games (Houstis and Rice 2000), taking advantage of immersive environments, "abstract worlds and spaces with new rules and topologies" (Rice 1996, 5). It has been reported, over a decade ago, that new scientists and students were missing videogame-like features in SS (Javahery et al. 2004). Indeed, videogames should become a major influence on how interfaces are designed (Isbister 2011). Paul Brown (1996) has calculated the necessary time for mainstream adoption of new paradigms to be around forty years—approximately the amount of time since domestic video game systems have been released. Over two decades ago, PSEs were already taking advantage of multimedia and interactive tools for 3D visualization, also supporting simultaneous users in collaborative mode (Anupam and Bajaj 1993). In the early 1990s, multimedia capabilities for data visualization, such as 3D graphics, sound, and animation, were celebrated. There was an anticipation for virtual reality environments where users could "step inside their data" by using technologies "driven by mammoth commercial markets for games and entertainment" (Kornbluh 1993, 74). According to Sloan, McCorkle, and Bryden (2013), the evolution of the PSE is to become an Integrated Computational Environment featuring plug and play interaction and customizable real-time visualization tools (Sloan, McCorkle, and Bryden 2013)—arguably, qualities present in games. Moreover, the experimental quality of SS gives it a playful character—which can be illustrated by scientific applications named "playgrounds" (Larkin et al. 2009), or by simulated spaces denominated "playboxes" (Feibush, Gagvani, and Williams 2000, 38).

Gamification is frequently designed around motivational affordances characteristic of games. These include points, leaderboards, goals, levels, challenges, etc. (Hamari, Koivisto, and Sarsa 2014). This approach, combined with videogame-like virtual environment, has reportedly increased enjoyment, perceived ease of use, and *flow experience* (Herzig, Strahringer and Ameling 2012)—a mental state of optimal focus and engagement derived from ideal balance between challenges and skills (Csikszentmihalyi 1997). Video games often elicit flow through attention to feedback, contextual learning, and adjustment to player skills (Morris et al. 2013).

Beyond structural qualities and motivational affordances, games have a user-centric quality of teaching necessary skills for users to achieve goals and engage with the system. This is investigated by Sebastian Deterding (2013), who proposes a design lens and elaborates a gamification method for "[translating] game design insights for interaction design" (Deterding 2015, 329). Drawing inspiration from games is also a central issue to Dana Maria Popa, who argues that gamification of software should take advantage of "cross references from games and game design process…for designing better emotional experiences" (Popa 2013a, 8). Popa's guidelines recommends that gamified software should "facilitate optimal user experience," "give clear feedback," sustain "productivity or efficiency," "provide a safe play space," and "respect ethical goals" (Popa 2013a, 18). To better understand user requirements, Popa has also developed a variation of the persona method focused on emotional response to gamified experiences (Popa 2013b).

Regarding STEM applications, the state-of-the-art in gamification seems to be represented by engineering software—especially CAD and BIM—where game mechanics, aesthetics, and technologies have been applied to improve usability (Kosmadoudi et al. 2013). Such systems draw inspiration from videogames to tailor experiences that are more responsive, intuitive, and compelling than the ones made possible by conventional tools (Boeykens 2011). Such experiences take advantage of aesthetical (Aydin and Schnabel 2014), four-dimensional, immersive capabilities of game design and technologies (Moloney 2015). Parametric BIM models have been transformed into gamified environments, in some cases featuring Virtual and Augmented Reality technologies developed through videogame engines (Keenaghan and Horvath

2014). This immersive quality is not the only benefit: Autodesk Research team has applied a sense of structured play to GamiCAD, a gamified tutorial system for AutoCAD, by adding clear goals, feedback, sense of progression, guidance, time pressure, rewards, fictional setting, and visual stimulation (Li, Grossman, and Fitzmaurice 2012). Researchers from the Chinese University of Hong Kong developed ModRule, a software that explores collaborative aspects of gamification to bring closer architects and inhabitants in mass housing production (Lo et al. 2014). Researchers from the University of Calgary have been investigating how games can be used to teach and learn engineering. At a particular stage of their research, undergraduate students have created fully functional games about circuit design techniques (Marasco, Behjat, and Rosehart 2015). These games are, essentially, scientific software—as they allow for simulations modeled around actual scientific requisites.

The use of games in science education has been increasing, either through gamified learning or the use of scientifically themed games. These are considered useful practices, since games can elicit motivation and cognitive skills, fostering scientific thinking and learning (Morris et al. 2013). In some cases, educational games can be supported by data obtained through scientific software (Garcia Esquirol 2015).

There are propositions for gamifying SS development: Daniel Katz (2015) proposes gamification as a means to stimulate the building of scientific software communities, which could assist trends such as crowdsourced documentation (Pawlik et al. 2015). Other possible applications include test case generation and software verification (Mao et al. 2015). Requirement elicitation, critical in SS development, could benefit from approaches such as IThink, a gamified application for that end (Fernandes et al. 2012).

Although usually not designed for specialists, Citizen Science (CS) has become a very successful and prolific venue for gamified science, and could inform SS in many ways. Gamified citizen science is part of an emerging culture of massive collaborative scientific initiatives that make use of crowdsourced skills—ranging from common human skills to domain-specific ones—to handle tasks. These tasks can vary regarding how well-structured and independent from each other they are. In this context, gamification can transform monotonous tasks into compelling activities (Franzoni and Sauermann 2014). CS has been used to enlighten, educate, and collect scientific data from the general public (MacDonald et al. 2015) or, occasionally, specialists (Good et al. 2012). Crowdsourced problem-solving skills can be used to replace limited computational power (Cooper et al. 2010), or improve software accuracy (Mason, Michalakidis, and Krause 2012). In that sense, Cooper observed that videogames serve well citizen science by "combining what humans are good at with what computers are good at" (Cooper 2015, 490). User motivation in CS games can be elicited by game design elements (Bowser, Hansen, and Preece 2013), fictional settings (Prestopnik and Tang 2015), socialization (Bowser, Hansen, and Preece 2013), fun and amusement (Greenhill et al. 2014), discovery and education (Bowser et al. 2014), altruism (Schrope 2013), and previous interest in science (Iacovides et al. 2013). Success in CS projects can be measured for its contribution to science—e.g., publication rate, academic impact—and for its public engagement (Simmons et al. 2015). Through CS games, players can learn advanced topics while generating data that will expand that knowledge globally (Devlin et al. 2014).

Applying competitive point-based systems to CS is somewhat controversial: it could motivate some users (Bowser, Hansen, and Preece 2013) while having a de-motivational effect in others (Eveleigh et al. 2013). In this case, compelling experiences could be more effectively created through fictional settings (Prestopnik and Tang 2015), refined and interesting aesthetics (Kappen, Johannsmeier, and Nacke 2013), or explorative freeform play (Ponti, Hillman, and Stankovic 2015). Some studies link the amount and quality of collected data to the level of competitiveness (Preist, Massung, and Coyle 2014), while other sources claim that data quality is not negatively affected by neither point-based or story-based approaches (Prestopnik, Crowston, and Wang 2014). That is an important debate, as low quality of collected data is a risk in CS (Sandbrook, Adams, and Monteferri 2015).

Guidelines and case studies from gamified CS raise points that can be relevant to SS. Bowser, Hansen, and Preece (2013) state that gamified CS should support both casual and expert users, and that data quality should not be compromised. Jennett and Cox enlist a series of desirable attributes in CS projects: (1) clear presentation; (2) clear text (avoiding technicalities); (3) supportive learning material; (4) functionalities that help users to complete tasks; (5) attention to users level of expertise; (6) reminders of the importance of users contribution; (7) feedback on progress; and (8) engage learning on the tasks, the science behind it, and the community around it (Jennett and Cox 2014). Some of these guidelines could be applied to SS (especially numbers 4 and 7, but also 1, 3, and 5).

Cooper et al. (2010) discuss the design process behind Foldit, a game where hundreds of thousands players contributed to solve a very complex problem regarding protein folding. For this project, game designers were constantly informed by two groups: scientists who explained the underlying science behind the game, and players who discussed gameplay. Their study elaborates design challenges regarding visualization and interaction that could be pertinent to SS: *Visualizations* should make rules of represented systems visible and clear, manage the complexity of scientific models, and make the game approachable by non-experts (but customizable for experts). *Interactions* should be intuitive and fun, respect constraints of the scientific model, and allow for enough exploration. Moreover, *scores* should serve as feedback, indicating that players are moving towards a valid solution, and *introductory levels* should teach how the game works (Cooper et al. 2010). Arguably, the challenge of developing “an accessible interface to complex structures and problems” (Cooper, Khatib, and Baker 2013) could be transported to SS.

Lastly, since SS is used professionally, gamification should look at work-related gains and challenges. Oprescu, Jones, and Katsikitis (2014) have delineated ten principles for gamifying workplaces, listing expected benefits such as increased engagement; development of capabilities; increased satisfaction; and enhanced productivity. Their work claims that younger generations might value gamified workplaces more than older ones. A similar remark has been done two decades before by Webster and Martocchio (1993), who observed that younger employees were more receptive to playful approaches to work. Gamified work can raise objections: gamification practices have been accused of being exploitative (Bogost 2015), especially when underlying objectives of gamification proponents are not aligned with the workers’ (Bigham, Bernstein, and Adar 2014). In that case, is recommended that workers do not lose the connection with the real purpose of their work—as opposed to gamified motivational elements (Kim 2015).

Discussion

Structuring a design lens based on our findings requires such information to be condensed, reorganized, and articulated as a brief introduction to the subject and a set of questions. To assist with the process, we have gone through the findings, identifying challenges, issues, and opportunities, to organize them as shown in Table 2.

Table 2: Identified Challenges, Issues, and Opportunities

Development	Project size; team size; development methodologies; timescale; software lifecycle; technologies; expandability; team collaboration.
User-base	User-base size and necessities; multiple types and levels of user specialization; computer literacy; user expectations; empowerment through participation in development; community building.
Professional	Work practices, conditions, culture, ethics, safety norms, conventions, and best practices; evaluation from science-domain expert.
Scientific/Academic	Correctness; reproducible; data complexity and quality; changeable requirements; rigor.
Software	Expandable; incremental; portable; configurable; performance-oriented; easy to install; interoperability; automatable; target system requirements.
General UI	User-friendly; easy to learn; customizable; adequate to platform; consistent; flexible; minimalistic; incremental.
Specialized Usability	Complex data input/output and monitoring; precision; annotation tools; access to programming and knowledge-base; prevention and recovery from errors; report generation; metadata; GUI bypassing.
Modeling, Simulation and Result Analysis	Productive modeling and workflow composition; insightful result analysis through visualizations, navigation, and manipulation of data in two, three or four dimensions. Adequate contextualization and integration between modeling, simulation, and result analysis phases.
Gamification and Playful Design	Games as source of inspiration; better presentation and aesthetics; interactivity; technologies; game-like structures; goals; feedback; guidance; progression; flow; fun; exploration; adequate motivational design elements (score, points, etc.).

Table 3: Formatted Lens

The Lens of the Lab	
Scientific software should augment insight, productivity, and knowledge. It should facilitate and integrate supported stages of scientific work (modeling, simulation and result analysis), and generate output for publication, sharing, or further research. When designing for scientific software, consider the questions:	
	How can the interface represent the scientific matter, reinforce the way it works and support the theory behind it? How can it present and explore complex data at high levels of precision? How can it prevent and fix errors?
	Is the user interface intuitive, consistent and uncluttered? Is it flexible enough to allow for incremental expansion and customization? Is it adequate to the platforms it was designed for, and to other software it should be integrated to?
	How do scientists work? How is the work environment, culture, ethics, conventions, current practices and best practices? What do users need and expect? How can design embrace different levels of scientific specialization, computer literacy, and programming skills? How can it promote and attract collaboration or community building?
	How can games inform and inspire the software aesthetics and interactivity? Which game design elements could provide structure, goals, feedback, guidance, progression, flow, fun and experimentation? Would competition and point-based systems motivate or demotivate?
	Is implementation feasible regarding scope, planning, timescale, technologies, human resources, and software lifecycle?

Finally, we adapt that content to the design lens formatting, as shown in Table 3. Applying the lens to a project should be a straightforward process, basically consisting of either (a) reflecting upon the questions during design stage or (b) consulting appropriate stakeholders about the issues at hand.

Conclusion

Development and use of scientific software are very particular, often intertwined, activities, influenced by diverse technical, professional, and scientific needs. Usability in SS is, comprehensibly, regarded as deficient. However, it has also been approached very professionally by designers and researchers—a trend that should be increased by efficient development methodologies. The visual, interactive, structured, and experimental qualities of SS seem to qualify it as an opportune and natural venue for gamification and playful design—widely and successfully applied to citizen science, from which design lessons could be taken. By restructuring our literature review findings, we proposed a design lens to support gamification and usability design for SS. This lens should be further examined and tested as an extension of the current work, as well as complimented by additional research on game design applied to scientific software.

Acknowledgement

The authors would like to thank Marcelo Gattass, Alberto Raposo, Eduardo Thadeu Courseuil, Felipe de Carvalho, and Pablo Carneiro Elias, from Tecgraf/PUC-Rio, for their comments and remarks on scientific software usage, development, and trends. This study is part of a PhD research made possible by PUC-Rio.

REFERENCES

Ackroyd, Karen S., Steve H. Kinder, Geoff R. Mant, Mike C. Miller, Christine A. Ramsdale, and Paul C. Stephenson. 2008. "Scientific Software Development at a Research Facility." *IEEE Software* 25 (4): 44–51. doi:10.1109/MS.2008.93.

Afgan, Enis, Konstantinos Krampis, Nuwan Goonasekera, Karolj Skala, and James Taylor. 2015. "Building and Provisioning Bioinformatics Environments on Public and Private Clouds." In *2015 38th International Convention on Information and Communication Technology, Electronics, and Microelectronics (MIPRO)*, 223–28. Piscataway: IEEE.

Ahalt, Stan, Larry Band, Laura Christopherson, Ray Idaszak, Chris Lenhardt, Barbara Minsker, Margaret Palmer, Mary Shelley, Michael Tiemann, and Ann Zimmerman. 2014. "Water Science Software Institute: Agile and Open Source Scientific Software Development." *Computing in Science and Engineering* 16 (3): 18–26.

Ahmed, Zeeshan, and Saman Zeeshan. 2014. "Cultivating Software Solutions Development in the Scientific Academia." *Recent Patents on Computer Science* 7 (1): 54–66.

Ahmed, Zeeshan, Saman Zeeshan, and Thomas Dandekar. 2014. "Developing Sustainable Software Solutions for Bioinformatics by the 'Butterfly' Paradigm." [version 2; referees: 2 approved]. F1000Research 2014, 3:71 (doi: 10.12688/f1000research.3681.2) Anupam, Vinod, and Chandrajit L. Bajaj. 1993. "Collaborative Multimedia Scientific Design in SHASTRA." In *Proceedings of the First ACM International Conference on Multimedia*, 447–56. New York: ACM.

Aragon, Cecilia R., Sarah S. Poon, Gregory S. Aldering, Rollin C. Thomas, and Robert Quimby. 2008. "Using Visual Analytics to Maintain Situation Awareness in Astrophysics." In

IEEE Symposium on Visual Analytics Science and Technology, 27–34. Piscataway: IEEE.

Atkins, Daniel E., and Richard L. Phillips. 1986. "A User Perspective on Computer Workstation Integration." *IEEE Circuits and Devices Magazine* 2 (4): 22–31.

Aydin, Serdar, and Marc Aurel Schnabel. 2014. "A Survey on the Visual Communication Skills of BIM Tools." In *Rethinking Comprehensive Design: Speculative Counterculture, Proceedings of the 19th International Conference on Computer-aided Architectural Design Research in Asia CAADRIA 2014*, edited by N. Gu, S. Watanabe, H. Erhan, M. Hank Haeusler, W. Huang, and R. Sosa, 337–46. Kyoto: Kyoto Institute of Technology.

Basili, Victor R., Jeffrey C. Carver, Daniela Cruzes, Lorin M. Hochstein, Jeffrey K. Hollingsworth, Forrest Shull, and Marvin V. Zelkowitz. 2008. "Understanding the High-performance-computing Community: A Software Engineer's Perspective." *IEEE Software* 4: 29–36.

Bastos, Bruno F., Vinicius Macedo Moreira, and Antônio Tadeu A. Gomes. 2013. "Rapid Prototyping of Science Gateways in the Brazilian National HPC Network." In Proceedings of the International Workshop on Science Gateways, Zurich, Switzerland, 2013. Accessed 21 July, 2016. http://ceur-ws.org/Vol-993/paper8.pdf.

Baxter, Susan M., Steven W. Day, Jacquelyn S. Fetrow, and Stephanie J. Reisinger. 2006. "Scientific Software Development Is Not an Oxymoron." *PLoS Computational Biology* 2 (9): e87.

Belsky, Alec, Mariette Hellenbrandt, Vicky Lynn Karen, and Peter Luksch. 2002. "New Developments in the Inorganic Crystal Structure Database (ICSD): Accessibility in Support of Materials Research and Design." *Acta Crystallographica Section B: Structural Science* 58 (3): 364–69.

Bergmann, Sandra, Bastian Demuth, and Volker Sander. 2011. "A Web Framework for Workflow Submission and Monitoring via UNICORE 6 Based on Distributable Scientific Workflow Templates." *Schriften Des Forschungszentrums Jülich IAS Series* 9: 45.

Bigham, Jeffrey P., Michael S. Bernstein, and Eytan Adar. 2014. *Human-computer Interaction and Collective Intelligence*. Cambridge: MIT Press.

Bilmes, Jeff. 1996. "User-friendly Neural-net Design." *IEEE Spectrum* 33 (2): 63.

Bitter, Ingmar, Robert Van Uitert, Ivo Wolf, Luis Ibanez, and J-M Kuhnigk. 2007. "Comparison of Four Freely Available Frameworks for Image Processing and Visualization That Use ITK." *IEEE Transactions on Visualization and Computer Graphics* 13 (3): 483–93.

Boeykens, Stefan. 2011. "Using 3D Design Software, BIM, and Game Engines for Architectural Historical Reconstruction." Liege: CAADFutures.

Bogost, Ian. 2015. "Why Gamification Is Bullshit 2." In *The Gameful World: Approaches, Issues, Applications*, edited by Steffen P. Walz and Sebastian Deterding, 65. Cambridge: MIT Press.

Bowser, A., D. Hansen, and J. Preece. 2013. "Gamifying Citizen Science: Lessons and Future Directions." Paper presented at CHI 2013 Conference, Paris, France, April 28.

Bowser, Anne, Derek Hansen, Yurong He, Carol Boston, Matthew Reid, Logan Gunnell, and Jennifer Preece. 2013. "Using Gamification to Inspire New Citizen Science Volunteers." In *Gamification 2013: Proceedings of the First International Conference on Gameful Design, Research, and Applications*, 18–25. New York: ACM.

Bowser, Anne, Derek Hansen, Jennifer Preece, Yurong He, Carol Boston, and Jen Hammock. 2014. "Gamifying Citizen Science: A Study of Two User Groups." In *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, 137–40. New York: ACM.

Brebner, Paul C. 1998. “Software Process Improvement by Example (SPIE).” In *Software Engineering: Education and Practice, 1998. Proceedings. 1998 International Conference*, 88–95. Piscataway: IEEE.

Brookes, Emre H., Nadeem Anjum, Joseph E. Curtis, Suresh Marru, Raminder Singh, and Marlon Pierce. 2015. “The GenApp Framework Integrated with Airavata for Managed Compute Resource Submissions.” *Concurrency and Computation: Practice and Experience* 27 (16): 4292–4303. doi:10.1002/cpe.3519.

Brouwer-Janse, Maddy D. 1990. “AI Technologies for User Interfaces: Knowledge-based Front-ends.” In *IEE Colloquium on AI in the User Interface*, 2/1–2/3. Stevenage : IET.

Brown, Paul. 1996. “New Media: An Emergent Paradigm.” *Periphery* 29: 13–15.

Bunus, Peter. 2006. “A Simulation and Decision Framework for Selection of Numerical Solvers in Scientific Computing.” In *Proceedings of the 39th Annual Symposium on Simulation*, 178–87. Washington, DC: IEEE Computer Society.

Caillois, Roger. 2001. *Man, Play, and Games*. Urbana: University of Illinois Press.

Chambers, Christopher. 2013. “Helping End Users Find and Fix Performance Issues in Visual Dataflow Code.” In *2013 IEEE Symposium on Visual Languages and Human-centric Computing*, 169–70. Piscataway: IEEE.

Chancelier, Jean-Philippe, Bernard Lapeyre, and Jérôme Lelong. 2014. “Using Premia and Nsp for Constructing a Risk Management Benchmark for Testing Parallel Architecture.” *Concurrency and Computation: Practice and Experience* 26 (9): 1654–65. doi: 10.1002/cpe.2893.

Chen, Chaomei, Jian Zhang, and Michael S Vogeley. 2009. “Reflections on the Interdisciplinary Collaborative Design of Mapping the Universe.” In *Human-computer Interaction. Interacting in Various Application Domains: 13th International Conference, HCI International 2009, San Diego, CA, USA, July 19-24, 2009, Proceedings, Part IV*, edited by Julie A. Jacko, 693–702. Berlin: Springer.

Chen, Y. W., and T. Y. Fu. 1996. “The Use of World Wide Web Browsers to Create Graphical User Interfaces (GUIs) for Crystallographic Software: A GUI Implementation for X-PLOR.” *Journal of Applied Crystallography* 29 (2): 202–03.

Choi, Young Jun, Takashige Oroguchi, Yoshinori Kato, Makoto Takeda, and Yoshio Tago. 2006. “Labgrid: Integrated Problem Solving Environment System for High Throughput Computing.” In *2006 Second IEEE International Conference on E-Science and Grid Computing*, 103–103. Piscataway: IEEE.

Church, Philip, Adam Wong, Michael Brock, and Andrzej Goscinski. 2012. “Toward Exposing and Accessing HPC Applications in a SaaS Cloud.” In 2012 IEEE 19th International Conference on Web Services (ICWS), 692–99. Piscataway: IEEE.

Cimiano, Philipp, and Ing Gerhard Sagerer. 2015. “Continuous Quality Control for Research Data to Ensure Reproducibility: An Institutional Approach.”

Clark, Alex M. 2014. “Cheminformatics: Mobile Workflows and Data Sources” In *The Future of the History of Chemical Information*, edited by Leah R McEwen and Robert E. Buntrock, 237–53. Washington, DC: American Chemical Society.

Cohen, Johanne, David Moxey, Chris Cantwell, Pavel Burovskiy, John Darlington, and Spencer J Sherwin. 2013. “Nekkloud: A Software Environment for High-order Finite Element Analysis on Clusters and Clouds.” In *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, 1–5. Piscataway: IEEE.

Cooper, Seth. 2015. “Massively Multiplayer Research: Gamification And (Citizen) Science” In *The Gameful World: Approaches, Issues, Applications*, edited by Steffen P. Walz and Sebastian Deterding, 487-500. Cambridge: MIT Press.

Cooper, Seth, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, David Salesin, David Baker, and Zoran Popović. 2010. “The Challenge of Designing Scientific Discovery Games.” In *Proceedings of the*

Fifth International Conference on the Foundations of Digital Games, 40–47. New York: ACM.

Cooper, Seth, Firas Khatib, and David Baker. 2013. "Increasing Public Involvement in Structural Biology." *Structure* 21 (9): 1482–84. doi:10.1016/j.str.2013.08.009.

Cort, G, J. A. Goldstone, R. O. Nelson, R. V. Poore, L. Miller, and D. M. Barrus. 1985. "A Development Methodology for Scientific Software." *IEEE Transactions on Nuclear Science* 32 (4): 1439–43.

Crabtree, Carlton, A. Gunes Koru, Carolyn Seaman, and Hakan Erdogmus. 2009. "An Empirical Characterization of Scientific Software Development Projects According to the Boehm and Turner Model: A Progress Report." In *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering,* 22–27. Washington, DC: IEEE.

Csikszentmihalyi, Mihaly. 1997. *Finding Flow: The Psychology of Engagement with Everyday Life*. New York: Basic Books.

Curtis, Ross E., Peter Kinnaird, and Eric P. Xing. 2011. "GenAMap: Visualization Strategies for Structured Association Mapping." In *2011 IEEE Symposium on Biological Data Visualization*, 87–94. Piscataway: IEEE.

Daniluk, Andrzej. 2012. "Visual Modeling for Scientific Software Architecture Design: A Practical Approach." *Computer Physics Communications* 183 (2): 213–30.

De Carvalho, Felipe Gomes, Alberto Raposo, and Marcelo Gattass. 2009. "Designing a Hybrid User Interface: A Case Study on an Oil and Gas Application." In *Proceedings of the 8th International Conference on Virtual Reality Continuum and Its Applications in Industry*, 191–96. New York: ACM.

De Matos, Paula, Jennifer A. Cham, Hong Cao, Rafael Alcántara, Francis Rowland, Rodrigo Lopez, and Christoph Steinbeck. 2013. "The Enzyme Portal: A Case Study in Applying User-centered Design Methods in Bioinformatics." *BMC Bioinformatics* 14 (1): 103.

De Roure, David, and Carole Goble. 2009. "Software Design for Empowering Scientists." *IEEE Software* 26 (1): 88–95.

Deterding, Sebastian. 2013. "Skill Atoms as Design Lenses for User-centered Gameful Design." In *Workshop Papers CHI 2013*. New York: ACM.

———. 2015. "The Lens of Intrinsic Skill Atoms: A Method for Gameful Design." *Human-computer Interaction* 30 (3–4): 294–335.

Deterding, Sebastian, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. "From Game Design Elements to Gamefulness: Defining Gamification." In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, 9–15. New York: ACM.

Devlin, Sam, Peter Cowling, Daniel Kudenko, Nikolaos Goumagias, Alberto Nucciareli, Ignazio Cabras, Kiran Jude Fernandes, and Feng Li. 2014. "Game Intelligence." In *2014 IEEE Conference on Computational Intelligence and Games*, 1–8. Piscataway: IEEE.

Dong, Xiao, and David Wild. 2008. "An Automatic Drug Discovery Workflow Generation Tool Using Semantic Web Technologies." In *IEEE Fourth International Conference on eScience, 2008*, 652–57.

Dongarra, Jack, Gene H. Golub, Eric Grosse, Cleve Moler, and Keith Moore. 2008. "Netlib and NA-Net: Building a Scientific Computing Community." *IEEE Annals of the History of Computing* 1 (2): 30–41.

Eliceiri, Kevin W, Michael R Berthold, Ilya G Goldberg, Luis Ibáñez, Bangalore S Manjunath, Maryann E Martone, Robert F Murphy, et al. 2012. "Biological Imaging Software Tools." Nature Methods 9 (7). Nature Publishing Group: 697–710.

Ellis, Heidi J. C., Gerard Weatherby, Ronald J. Nowling, Jay Vyas, Matt Fenwick, and Michael Gryk. 2013. "A Pipeline Software Architecture for NMR Spectrum Data Translation." *Computing in Science and Engineering* 15 (1): 76–83.

Ernst, Thilo, Tom Rother, Franz Schreier, L. Wauer, and Wolfgang Balzer. 2003. "DLR's VirtualLab: Scientific Software Just a Mouse Click Away." *Computing in Science and Engineering* 5 (1): 70–79.

Eveleigh, Alexandra, Charlene Jennett, Stuart Lynn, and Anna L. Cox. 2013. "I Want to Be a Captain! I Want to Be a Captain!: Gamification in the Old Weather Citizen Science Project." In *Gamification 2013: Proceedings of the First International Conference on Gameful Design, Research, and Applications*, 79–82. New York: ACM.

Fdez-Riverola, F., D. Glez-Peña, H. López-Fernández, M. Reboiro-Jato, and J. R. Méndez. 2012. "A JAVA Application Framework for Scientific Software Development." *Software: Practice and Experience* 42 (8): 1015–36. doi:10.1002/spe.1108.

Feibush, Eliot, Nikhil Gagvani, and Daniel Williams. 2000. "Visualization for Situational Awareness." *IEEE Computer Graphics and Applications* 20 (5): 38–45.

Fernandes, João, Diogo Duarte, Claudia Ribeiro, Carla Farinha, João Madeiras Pereira, and Miguel Mira da Silva. 2012. "iThink: A Game-based Approach Towards Improving Collaboration and Participation in Requirement Elicitation." *Procedia Computer Science* 15: 66–77.

Fischer, Felix, M. Alper Selver, Walter Hillen, and Cüneyt Güzeli\cs. 2010. "Integrating Segmentation Methods from Different Tools into a Visualization Program Using an Object-based Plug-in Interface." *IEEE Transaction on Information Technology in Biomedicine* 14 (4): 923–34.

Foster, Kenneth R. 1998. "Software Tools [Technology 1998 Analysis and Forecast]." IEEE Spectrum 35 (1): 52–56.

Foulser, David E., and William D. Gropp. 1990. "CLAM and CLAMShell: A System for Building User Interfaces." In *The Second International Conference on Expert Systems for Numerical Computing*, edited by John R. Rice, Robert Vichnevetsky and Elias N. Houstis, 22–25.Report no. 90-963. Department of Computer Science, Purdue University. Accessed July 20, 2016. http://docs.lib.purdue.edu/cstech/817/.

Fourquet, J-Y, V. Hue, and P. Chiron. 2007. "Olarge: On Kinematic Schemes and Regularization for Automatic Generation of Human Motion and Ergonomic Evaluation of Workplaces." In *IECON 2007 33rd Annual Conference of IEEE Industrial Electronics*, 2835–40. Piscataway: IEEE.

Frank, Alexander, Rainer Stotzka, Thomas Jejkal, Volker Hartmann, Michael Sutter, and Hartmut Gemmeke. 2007. "GridIJ-a Dynamic Grid Service Architecture for Scientific Image Processing." In 2007 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, 375–84. Piscataway: IEEE.

Franzoni, Chiara, and Henry Sauermann. 2014. "Crowd Science: The Organization of Scientific Research in Open Collaborative Projects.*" Research Policy* 43 (1): 1–20.

Gallopoulos, Stratis, Elias N. Houstis, and John R. Rice. 1992. "Future Research Directions in Problem Solving Environments for Computational Science." Paper presented at a Workshop on Research Directions in Integrating Numerical Analysis, Symbolic Computing, Computational Geometry, and Artificial Intelligence for Computational Science, Washington, DC, April 11–12.

Garcia Esquirol, Óscar. 2015. "Futuro de La Enseñanza Médica: Inteligencia Artificial Y Big Data." *FEM: Revista de La Fundación Educación Médica* 18: s60–s61.

Geimer, Markus, Kenneth Hoste, and Robert McLay. 2014. "Modern Scientific Software Management Using EasyBuild and Lmod." In *Proceedings of the First International Workshop on HPC User Support Tools*, 41–51. Piscataway: IEEE.

Gertz, Mathew E., David B. Stewart, and Pradeep K. Khosla. 1994. "A Human Machine Interface for Distributed Virtual Laboratories." *IEEE Robotics and Automation Magazine* 1 (4): 5–13.

Glez-Peña, Daniel, Miguel Reboiro-Jato, Paulo Maia, Miguel Rocha, Fernando Díaz, and Florentino Fdez-Riverola. 2010. "AIBench: A Rapid Application Development Framework for Translational Research in Biomedicine." *Computer Methods and Programs in Biomedicine* 98 (2): 191–203.

Gomes, Antonio Tadeu A., Bruno F. Bastos, Vivian Medeiros, and Vinicius M. Moreira. 2015. "Experiences of the Brazilian National High-performance Computing Network on the Rapid Prototyping of Science Gateways." *Concurrency and Computation: Practice and Experience* 27 (2): 271–89. doi:10.1002/cpe.3258.

Good, Benjamin M., Salvatore Loguercio, Max Nanis, and Andrew Su,. 2012. "Genegames. Org: High-throughput Access to Biological Knowledge and Reasoning through Online Games." In *2012 IEEE Second International Conference on Healthcare Informatics, Imaging and Systems Biology*, 145. Piscataway: IEEE.

Gorton, Ian. 2013. "Cyberinfrastructures: Bridging the Divide between Scientific Research and Software Engineering." In *Computer* 47 (8): 48–55. Piscataway: IEEE.

Gorton, Ian, Chandrika Sivaramakrishnan, Gary Black, Signe White, Sumit Purohit, Carina Lansing, Michael Madison, Karen Schuchardt, and Yan Liu. 2012. "Velo: A Knowledge-management Framework for Modeling and Simulation." *Computing in Science and Engineering* 14 (2): 12–23.

Greenhill, Anita, Kate Holmes, Chris Lintott, Brooke Simmons, Karen Masters, Joe Cox, and Gary Graham. 2014. "Playing with Science: Gamised Aspects of Gamification Found on the Online Citizen Science Project—Zooniverse." In *GAMEON 2014*, 15–24. London: EUROSIS.

Gross, Lutz, Hans Mühlhaus, Elspeth Thorne, and Ken Steube. 2008. "A New Design of Scientific Software Using Python and XML." *Pure and Applied Geophysics* 165 (3–4): 653–70.

Hamari, Juho, Jonna Koivisto, and Harri Sarsa. 2014. "Does Gamification Work?—A Literature Review of Empirical Studies on Gamification." In *2014 47th Hawaii International Conference on System Sciences*, 3025–34. Piscataway: IEEE.

Hannay, Jo Erskine, Carolyn MacLeod, Janice Singer, Hans Petter Langtangen, Dietmar Pfahl, and Greg Wilson. 2009. "How Do Scientists Develop and Use Scientific Software?" In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, 1–8. Piscataway: IEEE. doi:10.1109/SECSE.2009.5069155.

Hatton, L., and A. Roberts. 1994. "How Accurate Is Scientific Software?" *IEEE Transactions on Software Engineering 20* (10): 785–97. doi:10.1109/32.328993.

Heaton, Dustin, and Jeffrey C. Carver. 2015. "Claims about the Use of Software Engineering Practices in Science: A Systematic Literature Review." *Information and Software Technology* 67: 207–19.

Herzig, Philipp, Susanne Strahringer, and Michael Ameling. 2012. "Gamification of ERP Systems—Exploring Gamification Effects on User Acceptance Constructs." In *Proceedings of the 2012 MKWI Multikonferenz Wirtschaftsinformatik*, edited by Dirk Christian Mattfield and Susanne Robra-Bissantz, 793–804. Braunschweig: Institut für Wirtschaftsinformatik.

Hettrick, Simon. 2014. "It's Impossible to Conduct Research without Software, Say 7 out of 10 UK Researchers.." Accessed July 20, 2016. www.software.ac.uk/blog/2014-12-04-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers.

Hinsen, Konrad. 2013. "A Glimpse of the Future of Scientific Programming." *Computing in Science and Engineering* 15 (1): 84–88.

Holthouse, Mark, and Stuart G. Greenberg. 1978. "Software Technology for Scientific and Engineering Applications." In *1978 IEEE Computer Society's Second International Computer Software and Applications Conference*, 814–18. Piscataway: IEEE.

Hou, Zhengxiong, Xingshe Zhou, Jianhua Gu, Yunlan Wang, and Tianhai Zhao. 2010. "ASAAS: Application Software as a Service for High Performance Cloud Computing." In *2010 12th IEEE International Conference on High Performance Computing and Communications*, 156–63. Piscataway: IEEE.

Houstis, Elias N., and John R. Rice. 2002. "On the Future of Problem Solving Environments." *Computational Science, Mathematics, and Software*. Report no. 00-009. Department of Computer Science, Purdue University. Accessed July 20, 2016. http://docs.lib.purdue.edu/cstech/1487/.

Howison, James, and James Herbsleb. 2010. "Socio-technical Logics of Correctness in the Scientific Software Development Ecosystem." Paper presented at the *2010 ACM Conference on Computer Supported Cooperative Work*, Savannah, Georgia, February 6–10.

———. 2011. "Scientific Software Production: Incentives and Collaboration." In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, 513–22. New York: ACM.

Hughes, Gareth V., Hugo R. Mills, Graham Smith, Terry R. Payne, and Jeremy Frey. 2004. "Breaking the Book: Translating the Chemistry Lab Book into a Pervasive Computing Lab Environment." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 25–32. New York: ACM.

Hu, Bingjie, and Markus A. Lill. 2014. "PharmDock: A Pharmacophore-based Docking Program." *Journal of Cheminformatics* 6 (1): 14.

Iacovides, Ioanna, Charlene Jennett, Cassandra Cornish-Trestrail, and Anna L. Cox. 2013. "Do Games Attract or Sustain Engagement in Citizen Science?: A Study of Volunteer Motivations." In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 1101–6. New York: ACM.

Isbister, Katherine. 2011. "Emotion and Motion: Games as Inspiration for Shaping the Future of Interface." *Interactions* 18 (5): 24–27.

Jarvis, Roger M., David Broadhurst, Helen Johnson, Noel M. O'Boyle, and Royston Goodacre. 2006. "PYCHEM: A Multivariate Analysis Package for Python." *Bioinformatics* 22 (20): 2565–66.

Javahery, Homa, Ahmed Seffah, and Thiruvengadam Radhakrishnan. 2004. "Beyond Power: Making Bioinformatics Tools User-centered." *Communications of the ACM* 47 (11): 58–63.

Jennett, Charlene, and Anna L. Cox. 2014. "Eight Guidelines for Designing Virtual Citizen Science Projects." In Citizen + X: Volunteer-Based Crowdsourcing in Science, Public Health, and Government: Papers from the 2014 HCOMP Workshop , 16-17. Palo Alto, CA: The AAAI Press.

Jones, Michael, and Christopher Scaffidi. 2011. "Obstacles and Opportunities with Using Visual and Domain-Specific Languages in Scientific Programming." In *2011 IEEE Symposium on Visual Languages and Human-centric Computing*, 9–16. Piscataway: IEEE.

Joppa, Lucas N., Greg McInerny, Richard Harper, Lara Salido, Kenji Takeda, Kenton O'Hara, David Gavaghan, and Stephen Emmott. 2013. "Troubling Trends in Scientific Software Use." *Science* 340 (6134): 814–15.

Joshi, Anupam, Naren Ramakrishnan, Tzvetan Drashansky, Elias N. Houstis, John R. Rice, Sanjiva Weerawarana, and L. H. Tsoukalas. 1997. "Agent Based Systems to Support Multi-disciplinary Problem Solving Environments." Report no. 97-031. Department of Computer Science, Purdue University. Accessed July 20, 2016. http://docs.lib.purdue.edu/cstech/1368/.

Julvez, Jorge, Mihaela H. Matcovschi, and Octavian Pastravanu. 2014. "MATLAB Tools for the Analysis of Petri Net Models." In *2014 IEEE Emerging Technology and Factory Automation*, 1–12. Piscataway: IEEE.

Kappen, Dennis L., Jens Johannsmeier, and Lennart E. Nacke. 2013. "Deconstructing 'Gamified' Task-management Applications." In *Gamification 2013: Proceedings of the First International Conference on Gameful Design, Research, and Applications*, 139–42. New York: ACM.

Katz, Daniel S. 2015. "Building Scientific Software Communities." Presentation at HEP Software Foundation Workshop, Menlo Park, CA, January 20-22, 2015. Accessed July 27, 2016. http://pt.slideshare.net/danielskatz/software-communities

Keefe, Daniel F. 2010. "Integrating Visualization and Interaction Research to Improve Scientific Workflows." *IEEE Computer Graphics and Applications* 30 (2): 8–13.

Keenaghan, Garrett, and Imre Horvath. 2014. "State of the Art of Using Virtual Reality Technologies in Built Environment Education." In *TMCE 2014: Proceedings of the 10th International Symposium on Tools and Methods of Competitive Engineering*, edited by I. Horváth, Z. Rusák, 935-948. Delft: Delft University of Technology .

Keller, Richard M, and Michal Rimon. 1992. "A Knowledge-based Software Development Environment for Scientific Model-building." In *Proceedings of the Seventh Knowledge-based Software Engineering Conferenc*e, 192–201. Piscataway: IEEE.

Keller, Richard M., Michal Rimon, and Aseem Das. 1994. "A Knowledge-based Prototyping Environment for Construction of Scientific Modeling Software." *Automated Software Engineering* 1 (1): 79–128.

Kelly, Diane. 2007. "A Software Chasm: Software Engineering and Scientific Computing." *IEEE Software* 24 (6): 120–119. doi:10.1109/MS.2007.155.

———. 2009. "Determining Factors That Affect Long-Term Evolution in Scientific Application Software." *Journal of Systems and Software* 82 (5): 851–61.

———. 2013. "An Analysis of Process Characteristics for Developing Scientific Software." In *Innovative Strategies and Approaches for End-User Computing Advancements*, edited by Ashish Dwivedi and Steve Clarke. Hershey: IGI Global. services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-2059-9.

———. 2015. "Scientific Software Development Viewed as Knowledge Acquisition: Towards Understanding the Development of Risk-averse Scientific Software." *Journal of Systems and Software* 109: 50–61.

Kelly, Diane , and E. M. Skordaki. 2015. "A Medical Computing Lab and a Model of Learning." In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering*, 1561–66. Piscataway: IEEE.

Kelly, Diane, Daniel Hook, and Rebecca Sanders. 2009. "Five Recommended Practices for Computational Scientists Who Write Software." *Computing in Science and Engineering* 11 (5): 48–53.

Kelly, Diane, and Rebecca Sanders. 2008a. "Assessing the Quality of Scientific Software." In *First International Workshop on Software Engineering for Computational Science and Engineering*, Leipzig, Germany, May 2008. Accessed on July 21 2016. http://www.se4science.org/workshops/secse08/Papers/Kelly.pdf.

———. 2008b. "The Challenge of Testing Scientific Software." Paper presented at CAST 2008: Beyond the Boundaries, Toronto, Canada, July 14–16.

Kelly, Diane, and Spencer, Smith. 2010. "3rd CASCON Workshop on Software Engineering for Science." In *Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research*, 420–22. Indianapolis: IBM.

Kelly, Diane, Stefan Thorsteinson, and Daniel Hook. 2011. "Scientific Software Testing: Analysis with Four Dimensions." *IEEE Software* 28 (3): 84–90.

Kendall, Richard P., Douglass E. Post, Jeffrey C. Carver, Dale B. Henderson, and David A. Fisher. 2007. *A Proposed Taxonomy for Software Development Risks for High-performance Computing (HPC) Scientific/Engineering Applications* (CMU/SEI-2006-

TN-039). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2007. http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=8013

Killcoyne, Sarah, and John Boyle. 2009. "Managing Chaos: Lessons Learned Developing Software in the Life Sciences." *Computing in Science and Engineering* 11 (6): 20–29.

Kim, Tae Wan. 2015. "Gamification Ethics: Exploitation and Manipulation." Paper presented at CHI 2015, Seoul, South Korea, April 18–23.

Kodosky, Jeffrey, Jack MacCrisken, and Gary Rymar. 1991. "Visual Programming Using Structured Data Flow." In *Proceedings of the 1991 IEEE Workshop on Visual Languages*, 34–39. Piscataway: IEEE.

Kolberg, Sigbjørn, Daniel Courivaud, and Mehmet Efe Özbek. 2007. "LMS and Interactivity-technical Issues for Remote Laboratories." In *IEEE 18th International Symposium on Personal, Indoor, and Mobile Radio Communications*, 1–4. Piscataway: IEEE.

Kornbluh, Ken. 1993. "Engineering Software-Seeing Data in Action." *IEEE Spectrum* 30 (11): 60–64.

Kosmadoudi, Zoe, Theodore Lim, James Ritchie, Sandy Louchart, Ying Liu, and Raymond Sung. 2013. "Engineering Design Using Game-enhanced CAD: The Potential to Augment the User Experience with Game Elements." *Computer-aided Design* 45 (3): 777–95.

Koteska, Bojana, and Anastas Mishev. 2013. "Software Engineering Practices and Principles to Increase Quality of Scientific Applications." In *ICT Innovations 2012*, 245–54. City of publication: Springer.

Kovalchuk, Sergey V., Pavel A. Smirnov, Sergey S. Kosukhin, and Alexander V. Boukhanovsky. 2012. "Virtual Simulation Objects Concept as a Framework for System-level Simulation." In *Proceedings of 2012 IEEE e-Science Conference*, 1–8. Piscataway: IEEE. doi:10.1109/eScience.2012.6404413.

La Rue, Megan, Chi Shen, Alan Doizer, and Mathew Beck. 2014. "Ehanced GUI Environment for Pymatgen in Material Science." In *2014 5th IEEE International Conference on Software Engineering and Service Science*, 1186–90. Piscataway: IEEE.

Lande, Daniel Ross. 2008. "Implementation of an XML-based User Interface with Applications in Ice Sheet Modeling." Master's thesis, University of Montana.

Larkin, Narasimhan K., Sean Raffuse, Daniel Pryden, Alan Healy, Kevin Unger, Tara Strand, and Robert Solomon. 2009. *Conversion of the BlueSky Framework into Collaborative Web Service Architecture and Creation of a Smoke Modeling Application*. Research Project Reports, Paper no. 143. U.S. Joint Fire Science Program. Accessed July 21, 2016. http://digitalcommons.unl.edu/jfspresearch/143.

Letondal, Catherine. 2006. "Participatory Programming: Developing Programmable Bioinformatics Tools for End-users." In *End User Development*, edited by Henry Lieberman, Fabio Paternò, and Volker Wulf, 207–42. Dordrecht: Springer.

Li, Wei, Tovi Grossman, and George Fitzmaurice. 2012. "Gamicad: A Gamified Tutorial System for First Time Autocad Users." In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, 103–12. New York: ACM.

Li, Yang. 2011. "Reengineering a Scientific Software and Lessons Learned." In *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering*, 41–45. New York: ACM.

Lo, Tian Tian, Marc Aurel Schnabel, Serdar Aydin, and Kaixia Shi. 2014. "ModRule: Using Gamification for Collaborative Mass-housing Design Process." In Across: Architectural Research through to Practice: 48th International Conference of the Architectural Science Association, edited by F. Madeo and M. A. Schnabel, 733–43. Genova: Genova University Press.

Loynton, Scott, David Sloan, Jean-Marie Burel, and Catriona Macaulay. 2009. "Towards a Project Community Approach to Academic Scientific Software Development." In *2009*

5th IEEE International Conference on e-Science Workshops, 120–24. Piscataway: IEEE.

Lundstrom, Mark, and Gerhard Klimeck. 2006. "The NCN: Science, Simulation, and Cyber Services." In *2006 IEEE Conference on Emerging Technologies-nanoelectronics*, 496–500. Piscataway: IEEE.

Macaulay, Catriona, David Sloan, Xinyi Jiang, Paula Forbes, Scott Loynton, Jason R. Swedlow, and Peter Gregor. 2009. "Usability and User-centered Design in Scientific Software Development." *IEEE Software* 26 (1): 96–102. doi:10.1109/MS.2009.27.

MacDonald, E. A., N. A. Case, J. H. Clayton, M. K. Hall, Matt Heavner, Nicolas Lalone, K. G. Patel, and Andrea Tapia. 2015. "Aurorasaurus: A Citizen Science Platform for Viewing and Reporting the Aurora." *Space Weather* 13 (9): 548–59. doi:10.1002/2015SW001214.

MacLeod, Robert S., Christopher R. Johnson, and Mike A. Matheson. 1992. "Visualization of Cardiac Bioelectricity: A Case Study." In *Proceedings of the 1992 IEEE Conference on Visualization*, 411–18. Piscataway: IEEE.

Madsen, Astrid, Morten Svendsen, Rasmus Harr, and Ulrik Hauen-Limkilde. 2007. "Toyplay and Goalplay." Master's thesis, IT University. raum.pbworks.com/f/Toyplay+&+Goalplay+MSc+Thesis.pdf.

Manjunatha, Ashwin, Ajith H. Ranabahu, Paul E. Anderson, and Amit P. Sheth. 2011. "Identifying and Implementing the Underlying Operators for Nuclear Magnetic Resonance Based Metabolomics Data Analysis." Paper presented at the Third International Conference on Bioinformatics and Computational Biology, New Orleans, Louisiana, March 23–25.

Mao, Ke, Licia Capra, Mark Harman, and Yue Jia. 2015. "A Survey of the Use of Crowdsourcing in Software Engineering." Technical Report RN/15/01, Department of Computer Science, University College London, Accessed July 21, 2016. http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/research/Research_Notes/rn_15_01.pdf.

Marasco, Emily Ann, Laleh Behjat, and William Rosehart. 2015. "Integration of Gamification and Creativity in Engineering Design." Paper presented at 122nd ASEE Annual Conference and Exposition, Seattle, Washington, June 14–17. www.asee.org/file_server/papers/attachment/file/0005/7389/Marasco_ASEE2015_Final.pdf.

Marinovici, Cristina, Harold Kirkham, and Kevin Glass. 2014. "The Hidden Job Requirements for a Software Engineer." In *2014 47th Hawaii International Conference on System Sciences*, 4979–84. Piscataway: IEEE.

Mason, Aaron D., Georgios Michalakidis, and Paul J. Krause. 2012. "Tiger Nation: Empowering Citizen Scientists." In *2012 6th IEEE International Conference on Digital Ecosystems Technologies*, 1–5. Piscataway: IEEE.

McFaddin, H. Scott, and John R. Rice. 1992. "Architecture of the RELAX Problem Solving Environment." Report no. 92-081. Department of Computer Science, Purdue University. Accessed July 21, 2016. http://docs.lib.purdue.edu/cstech/1001/.

McGonigal, Jane. 2011. *Reality Is Broken: Why Games Make Us Better and How They Can Change the World*. New York: Penguin Group.

McKiney, K. M. 2003. "Thinking Inside the Box." *Scientific Computing and Instrumentation*. Accessed July 21, 2016. https://www.scientificcomputing.com/article/2003/05/thinking-inside-box

Mendez, Diego, Mario Villamiazr, and Hector Castro. 2013. "e-Clouds: Scientific Computing as a Service." In *2013 Seventh International Conference on Complex, Intelligent, and Software Intensive Systems*, 481–86. Piscataway: IEEE.

Mielke, Angela M., Sean M. Brennan, Mark C. Smith, David C. Torney, Arthur B. Maccabe, and Josh F. Karlin. 2005. "Independent Sensor Networks." *IEEE Instrumentation and Measurement Magazine* 8 (2): 33–37.

Mills, James K, Phillip Baines, Thomas Chang, Steven Chew, Trevor Jones, Stephen Lam, and Adi Rabadi. 1995. "Development of a Robot Control Test Platform." *IEEE Robotics and Automation Magazine* 2 (4): 21–28.

Mohammad, Atif Farid. 2010. "A New Perspective in Scientific Software Development." In Innovations and Advances in Computer Sciences and Engineering, 129–34. Springer.

Moloney, Jules. 2015. "Videogame Technology Re-purposed: Towards Interdisciplinary Design Environments for Engineering and Architecture." *Procedia Technology* 20: 212–18.

Morais, Hugo, Pieter Vancraeyveld, Birger Pedersen, Allan Henning, Morten Lind, H. Johannsson, and Jacob Ostergaard. 2014. "SOSPO-SP: Secure Operation of Sustainable Power Systems Simulation Platform for Real-time System State Evaluation and Control." *IEEE Transactions on Industrial Informatics* 10 (4): 2318–29.

Morris, Bradley J., Steve Croker, Corinne Zimmerman, Devin Gill, and Connie Romig. 2013. "Gaming Science: The 'Gamification' of Scientific Thinking." *Frontiers in Psychology* 4. doi:10.3389/fpsyg.2013.00607.

Morris, Chris, and Judith Segal. 2012. "Lessons Learned from a Scientific Software Development Project." *IEEE Software* 29 (4): 9–12.

Murali, P., D. Dutta, and R. N. Biswas. 1993. "An Integrated Framework for Quality Scientific Software Development." In *ACM SIGPLAN Fortran Forum* 12: 19–25.

Murphy, Michael J. 1996. "Utility of Coupling Nonlinear Optimization Methods with Numerical Modeling Software." Paper presented at the International Conference on Shock Waves in Condensed Matter, St. Petersburg, Russia, September 2–6.

Nanthaamornphong, Aziz, and Jeffrey C. Carver. 2015. "Test-driven Development in Scientific Software: A Survey." *Software Quality Journal*: 1–30.

Nguyên, Toàn, Laurentiu Trifan, and Jean-Antoine Désidéri. 2011. "Resilient Workflows for Cooperative Design." In *2011 15th International Conference on Computer Supported Cooperative Work in Design*, 69–75. Piscataway: IEEE.

Nguyen-Hoan, Luke, Shayne Flint, and Ramesh Sankaranarayana. 2010. "A Survey of Scientific Software Development." In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 1. New York: ACM. doi:10.1145/1852786.1852802.

Ntombela, M., K. K. Kaberere, K. A. Folly, and A. I. Petroianu. 2005. "An Investigation into the Capabilities of MATLAB Power System Toolbox for Small Signal Stability Analysis in Power Systems." In *2005 IEEE Power Engineering Society Inaugural Conference and Exposition in Africa*, 242–48. Piscataway: IEEE.

Oprescu, Florin, Christian Jones, and Mary Katsikitis. 2014. "I PLAY AT WORK—Ten Principles for Transforming Work Processes through Gamification." *Frontiers in Psychology* 5. doi: 10.3389/fpsyg.2014.00014.

Ovaska, Päivi, Matti Rossi, and Kari Smolander. 2005. "Filtering, Negotiating and Shifting in the Understanding of Information System Requirements." *Scandinavian Journal of Information Systems* 17 (1): 7.

Pancake, Cherri M. 1996. "'Improving the Usability of Numerical Software through User-centered Design." Technical Report 96-60-11, Oregon State University, Corvallis, OR, USA. Accessed July 21, 2016. https://web.engr.oregonstate.edu/~pancake/papers/ImpUsab.pdf

Papadimitriou, Spiros, K. Terzidis, Seferina Mavroudi, and S. Likothanassis. 2011. "Exploiting Java Scientific Libraries with the Scala Language within the ScalaLab Environment." *IET Software* 5 (6): 543–51.

Papadimitriou, Stergios, Konstantinos Terzidis, Seferina Mavroudi, and Spiridon Likothanassis. 2009. "Scientific Scripting for the Java Platform with jLab." *Computing in Science and Engineering* 11 (4): 50–60.

Pawlik, Aleksandra, Judith Segal, and Marian Petre. 2012. "Documentation Practices in Scientific Software Development." In *2012 5th International Workshop on Cooperative and Human Aspects of Software Engineering*, 113–19. Piscataway: IEEE.

Pawlik, Aleksandra, Judith Segal, Helen Sharp, and Marian Petre. 2015. "Crowdsourcing Scientific Software Documentation: A Case Study of the NumPy Documentation Project." *Computing in Science and Engineering* 17 (1): 28–36.

Pereira Junior, Manoel. 2007. "Concepção de Um Processo de Desenvolvimenro Específico Para Software Científico." PhD diss., Centro Federal de Educação Tecnológica de Minas Gerais. www.mmc.cefetmg.br/info/downloads/D027-ManoelPereiraJunior.pdf.

Phillips, Richard L., Daniel E. Atkins, Nancy Benovich, and Brian D. Schipper. 1986. "A Bridge from Full-function to Reduced-function Workstations." *IEEE Computer Graphics and Applications* 6 (5): 53–57.

Picón, Artzai, Arantza Bereciartua, José Angel Gutiérrez, and José Pérez. 2006. "3D High Precision Tube Bevel Measurement Using Laser Based Rotating Scanner." In *2006 IEEE Conference on Emerging Technologies and Factory Automation*, 1190–97. Piscataway: IEEE.

Pinto, Gustavo Da Rocha B., Júlia Célia Mercedes Strauch, Jano Moreira De Souza, Jonice Oliveira, Leonardo F. Cardoso, Lúcio Rogério Botelho, Lutieta Guerreiro Martorano, Emerson Cordeiro Morais, Manuel Antonio de Castro Jr., and Sergio Palma da Justa Medeiros. 2002. "A Framework to Support Scientific Knowledge Management: A Case Study in Agro-meteorology." In *7th International Conference on Computer Supported Cooperative Work in Design*, 320–24. Piscataway: IEEE.

Platz, Jochen. 1986. "Project Management in the Development of Scientific Software." *Computer Physics Communications* 41 (2): 217–25.

Ponti, Marisa, Thomas Hillman, and Igor Stankovic. 2015. "Science and Gamification: The Odd Couple?" In *Proceedings of the 2015 Annual Symposium on Computer-human Interaction in Play*, 679–84. New York: ACM.

Popa, Dana Maria. 2013a. "Design Case: Gamification of ERP—A User Centered Design Approach." Paper presented at the CHI 2013 Workshop: Designing Gamification: Creating Gameful and Playful experiences. Paris, France, April 28. Accessed July 21, 2016. http://gamification-research.org/wp-content/uploads/2013/03/Popa.pdf.

———. 2013b. "Industry Design Case: Introducing Gamification Persona Tool." In *Proceedings of the* CHI 2013 Workshop: Designing Gamification: Creating Gameful and Playful experiences, 50-55. Accessed July 21, 2016. http://gamification-research.org/wp-content/uploads/2013/03/CHI2013_Designing_Gamification_Workshop.pdf.

Prego, Juan José Gude, and Luis Vázquez Seisdedos. 2011. "Tailor-made Small Simulator for a Drum Boiler Control Based on Linear Techniques." In *2011 IEEE 16th Conference on Emerging Technologies and Factory Automation*, 1–4. Piscataway: IEEE.

Preist, Chris, Elaine Massung, and David Coyle. 2014. "Competing or Aiming to Be Average?: Normification as a Means of Engaging Digital Volunteers." In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, 1222–33. New York: ACM.

Prestopnik, Nathan, Kevin Crowston, and Jun Wang. 2014. "Exploring Data Quality in Games with a Purpose." In *iConference 2014 Proceedings*, 213–228. Illinois: iSchools. doi:10.9776/14010 .

Prestopnik, Nathan R., and Jian Tang. 2015. "Points, Stories, Worlds, and Diegesis: Comparing Player Experiences in Two Citizen Science Games." *Computers in Human Behavior* 52: 492–506.

Ramakrishnan, N., and John R. Rice. 1996. “Knowledge Discovery in Computational Science: A Case Study in Algorithm Selection.” Report no. 96-081. Department of Computer Science, Purdue University. Accessed July 21, 2016. http://docs.lib.purdue.edu/cstech/1335/.

Recio-Garcia, Juan A., Belén Diaz-Agudo, and Pedro Antonio González-Calero. 2013. “The COLIBRI Open Platform for the Reproducibility of CBR Applications.” In *Case-based Reasoning Research and Development*, 255–69. Berlin: Springer.

Rice, John R. 1996. “Scalable Scientific Software Libraries and Problem Solving Environments.” Report no. 96-001. Department of Computer Science, Purdue University. Accessed July 21, 2016. http://docs.lib.purdue.edu/cstech/1257/.

Rice, John R., and Ronald F. Boisvert. 1996. “From Scientific Software Libraries to Problem-solving Environments.” *Computing in Science and Engineering* 3: 44–53.

Rijnders, Frank M., Hans J. W. Spoelder, and Frans C. A. Groen. 1993. “Distributed Visual Programming Environment: Applications within Data-acquisition.” In *1993 Instrumentation and Measurement Technology Conference, Conference Record.*, 690–93. Piscataway: IEEE.

Sandbrook, Chris, William M. Adams, and Bruno Monteferri. 2015. “Digital Games and Biodiversity Conservation.” *Conservation Letters* 8 (2): 118–24. doi:10.1111/conl.12113.

Sanders, Rebecca. 2008. “The Development and Use of Scientific Software.” Master’s thesis, Queen’s University of Kingston.

Sanders, Rebecca, and Diane Kelly. 2008. “Dealing with Risk in Scientific Software Development.” *IEEE Software* 25 (4): 21–28. doi:10.1109/MS.2008.84.

Schell, Jesse. 2015. *The Art of Game Design: A Book of Lenses*. 2nd ed. Boca Raton: CRC Press.

Schneidewind, Lydia, Stephan Hörold, Cindy Mayas, Heidi Krömker, Sascha Falke, and Tony Pucklitsch. 2012. “How Personas Support Requirements Engineering.” In *Proceedings of the First International Workshop on Usability and Accessibility Focused Requirements Engineering*, 1–5. Piscataway: IEEE.

Schrope, Mark. 2013. “Solving Tough Problems with Games.” *Proceedings of the National Academy of Sciences* 110 (18): 7104–6.

Scott, B. 2010. “Designing with Lenses.” Accessed July 21, 2016. http://www.uxbooth.com/articles/designing-with-lenses/.

Segal, Judith. 2007. “Some Problems of Professional End User Developers.” In *IEEE Symposium on Visual Languages and Human-Centric Computing*, 111–18. Piscataway: IEEE. doi:10.1109/VLHCC.2007.17.

———. 2009. “Some Challenges Facing Software Engineers Developing Software for Scientists.” In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, 9–14. Washington, DC: IEEE Computer Sociey.

Segal, Judith, and Chris Morris. 2008. “Developing Scientific Software.” *IEEE Software* 25 (4): 18–20. doi:10.1109/MS.2008.85.

———. 2011. “Developing Software for a Scientific Community: Some Challenges and Solutions.” In *Handbook of Research on Computational Science and Engineering: Theory and Practice*, edited by Joanna Leng and Wes Sharrock, 177–96. Hershey: IGI Global.

Silva, Laryssa Aparecida Machado da. 2010. “Composer-science: Um Framework Para a Composição de Workflows Científicos.” Master’s thesis, Federal University of Juiz de Fora.

Simmons, Brooke, Chris Lintott, Karen Masters, Anita Greenhill, Gary Graham, and Kate Holmes. 2015. “Defining and Measuring Success in Online Citizen Science: A Case Study.” *Computing in Science and Engineering* 17: 28.

Skidmore, Edwin, Seung-jin Kim, Sangeeta Kuchimanchi, Sriramu Singaram, Nirav Merchant, and Dan Stanzione. 2011. “iPlant Atmosphere: A Gateway to Cloud Infrastructure for the Plant Sciences.” In *Proceedings of the 2011 ACM Workshop on Gateway Computing Environments*, 59–64. New York: ACM.

Sletholt, Magnus Thorstein, Jo Erskine Hannay, Dietmar Pfahl, and Hans Petter Langtangen. 2012. “What Do We Know about Scientific Software Development’s Agile Practices?” *Computing in Science and Engineering* 14 (2): 24–37.

Sloan, Benjamin M., Douglas S. McCorkle, and Kenneth M. Bryden. 2013. “An Overview of Computational Environments for Engineering.” In *51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 285. Reston: American Institute of Aeronautics and Astronautics.

Sloan, David, Catriona Macaulay, Paula Forbes, and Scott Loynton. 2009. “User Research in a Scientific Software Development Project.” In *Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology*, 423–29. Swinton: British Computer Society.

Spencer, Matt. 2015. “Brittleness and Bureaucracy: Software as a Material for Science.” *Perspectives on Science* 23 (4): 466–484.

Springmeyer, Rebecca R. 1993. “Applying Observations of Work Activity in Designing Prototype Data Analysis Tools.” In Visualization, 1993. Visualization’93, Proceedings., IEEE Conference on, 228–35.

Stewart, Mark E. M. 2001. “Automated Analysis of Scientific and Engineering Semantics.” In *Proceedings of the 2001 9th International Workshop on Program Comprehension*, 113–14. Piscataway: IEEE.

Takatsuka, Masahiro, and Mark N. Gahegan. 2001. “Exploratory Geospatial Analysis Using GeoVISTA Studio: From a Desktop to the Web.” In *Proceedings of the 2001 Second International Conference on Web Information Systems Engineering* 2: 92–101. Piscataway: IEEE.

Talbott, Tara, Michael Peterson, Jens Schwidder, and James D. Myers. 2005. “Adapting the Electronic Laboratory Notebook for the Semantic Era.” In *Proceedings of the 2005 International Symposium on Collaborative Technologies and Systems*, 136–43. Piscataway: IEEE.

Taweel, Adel, Brendan Delaney, Theodoros N. Arvanitis, and Lei Zhao. 2009. “Communication, Knowledge and Co-ordination Management in Globally Distributed Software Development: Informed by a Scientific Software Engineering Case Study.” In *2009 Fourth IEEE International Conference on Global Software Engineering*, 370–75. Piscataway: IEEE.

Terranova, Nadia, and Paolo Magni. 2012. “TGI-simulator: A Visual Tool to Support the Preclinical Phase of the Drug Discovery Process by Assessing in Silico the Effect of an Anticancer Drug.” *Computer Methods and Programs in Biomedicine* 105 (2): 162–74.

Trlica, Cary. 1997. “Software Tools [Technology Analysis and Forecast].” *IEEE Spectrum* 34 (1): 60–64.

Turk, Matthew. 2014. “Fostering Collaborative Computational Science.” *Computing in Science and Engineering* 16 (2): 68–71.

———. 2015 “Vertical Integration.” In *Computing in Science and Engineering* 17: 64–66.

Vanschoren, Joaquin, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. 2014. “OpenML: Networked Science in Machine Learning.” *ACM SIGKDD Explorations Newsletter* 15 (2): 49–60.

Verigan, Adam. 2007. “Improving Pediatric Cardiology Consultation Methods by Introducing Digital Interactive 3-D Heart Models: A Proof of Concept Study.” Master’s thesis, University of South Florida.

Versek, Craig William. 2013. “Charge Transport Studies of Proton and Ion Conducting Materials.” PhD diss., University of Massachusetts.

Vigder, Mark, Norman G. Vinson, Janice Singer, Darlene Stewart, and Keith Mews. 2008. “Supporting the Everyday Work of Scientists: Automating Scientific Workflows.” *IEEE Software* 25 (4): 52–58.

Wang, Guoqiang, Trung N. Tran, and Hugo Andrade. 2010. “A Graphical Programming and Design Environment for FPGA-Based Hardware.” In *2010 International Conference on Field-programmable Technology*, 337–40. Piscataway: IEEE.

Wauer, Jochen, Karsten Schmidt, Tom Rother, Thilo Ernst, and Michael Hess. 2004. “Two Software Tools for Plane-wave Scattering on Nonspherical Particles in the German Aerospace Center’s Virtual Laboratory.” *Applied Optics* 43 (35): 6371–79.

Webster, Jane, and Joseph J. Martocchio. 1993. “Turning Work into Play: Implications for Microcomputer Software Training.” *Journal of Management* 19 (1): 127–46.

Weerawarana, Sanjiva, Elias N. Houstis, John R. Rice, Ann C. Catlin, Margaret G. Gaitatzes, Shahani Markus, and Tzveten T Drashansky. 1996. “The Purdue PSE Kernel: Towards a Kernel for Building PSEs.” Report no. 96-082. Department of Computer Science, Purdue University. Accessed July 21, 2016. http://docs.lib.purdue.edu/cstech/1336/.

Willson, Bryan, Jeff Whitham, and Charles Anderson. 1992. “Estimating Ignition Timing from Engine Cylinder Pressure with Neural Networks.” In *Proceedings of the Intelligent Vehicles’ 92 Symposium*, 108–13. Piscataway: IEEE.

Wilson, Greg, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, Paul Wilson. 2014. “Best Practices for Scientific Computing.” *PLoS Biology* 12 (1): e1001745. doi:10.1371/journal.pbio.1001745.

Wolff, Christian. 2015. “The Case for Teaching ‘Tool Science.’” In *2015 IEEE Global Engineering Education Conference*, 932–38. Piscataway: IEEE.

Woollard, David, Nenad Medvidovic, Yolanda Gil, and Chris A. Mattmann. 2008. “Scientific Software as Workflows: From Discovery to Distribution.” *IEEE Software* 25 (4): 37–43. doi:10.1109/MS.2008.92.

Yamazaki, Tadashi, Hidetoshi Ikeno, Yoshihiro Okumura, Shunji Satoh, Yoshimi Kamiyama, Yutaka Hirata, Keiichiro Inagaki, Akito Ishihara, Takayuki Kannon, and Shiro Usui. 2011. “Reprint of: Simulation Platform: A Cloud-based Online Simulation Environment.” *Neural Networks* (9): 927–32. doi: 10.1016/j.neunet.2011.08.007.

ABOUT THE AUTHORS

Francisco Queiroz: Lecturer and PhD Student, Department of Arts and Design, PUC-Rio, Rio De Janeiro, Brazil

Rejane Spitz: Professor, Department of Arts and Design, PUC-Rio, Rio De Janeiro, Brazil

The International Journal of Design Management and Professional Practice is one of six thematically focused journals in the family of journals that support the Design Principles and Practices knowledge community—its journals, book series, conference and online community. It is a section of Design Principles and Practices: An International Journal.

The International Journal of Design Management and Professional Practice explores the organization of design, design work, and design as a professional practice. As well as papers of a traditional scholarly type, this journal invites presentations of practice—including case studies documenting professional practice, accompanied by exegeses analyzing organizational purposes, processes, and effects.

The International Journal of Design Management and Professional Practice is a peer-reviewed scholarly journal.

ISSN 2325-162X