



Original software publication

The eclipse integrated computational environment

Jay Jay Billings^{a,b,*}, Andrew R. Bennett^{a,c}, Jordan Deyton^{a,d}, Kasper Gammeltoft^{a,e},
Jonah Graham^f, Dasha Gorin^{a,g}, Hari Krishnan^h, Menghan Liⁱ, Alexander J. McCaskey^a,
Taylor Patterson^{a,j}, Robert Smith^a, Gregory R. Watson^a, Anna Wojtowicz^{a,k}

^a Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA

^b The Bredeben Center for Interdisciplinary Research and Graduate Education, University of Tennessee, 444 Greve Hall, 821 Volunteer Blvd. Knoxville, TN 37996-3394, United States

^c University of Washington, Seattle, WA 98105, United States

^d General Electric Company, 3200 North Grandview Blvd Waukesha, WI 53188-1678, United States

^e Georgia Institute of Technology North Avenue, Atlanta, GA 30332, United States

^f Kichwa Coders Ltd., 1 Plomer Green Avenue, Downley, High, Wycombe HP135 LN, United Kingdom

^g Northwestern University, 633 Clark Street Evanston, IL 60208, United States

^h Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, CA 94720, United States

ⁱ Department of Computer Science and Department of Biological Sciences, Purdue University, West Lafayette, IN 47906, United States

^j Acato Information Management, LLC, 114 Union Valley Rd., Oak Ridge, TN 37830, United States

^k Colorado State University, Fort Collins, CO 80523, United States



ARTICLE INFO

Article history:

Received 11 June 2017

Received in revised form 13 July 2018

Accepted 16 July 2018

Keywords:

Workflows

Workflow management

Supercomputing

Usability

Eclipse

ABSTRACT

Problems in modeling and simulation require significantly different workflow management technologies from standard grid-based workflow management systems. Computational scientists typically interact with simulation software in a feedback-driven way where solutions and workflows are developed iteratively and simultaneously. This work describes common modeling and simulation activities and how combinations of these activities form unique workflows. We present the Eclipse Integrated Computational Environment as a workflow management system and development environment for the modeling and simulation community. Examples of the Environment's applicability to problems in energy science, general multiphysics simulations, quantum computing, and other areas are presented along with its impact on the community at large.

© 2018 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Code metadata

Current code version	'next'
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_SOFTX-D-17-00043
Legal code license	Eclipse public license 1.0
Code versioning system used	Git
Software code languages, tools, and services used	Java, OSGi, Eclipse RCP, and Maven
Compilation requirements, operating environments and dependencies	Java 1.8 or greater, Maven, and an internet connection for dependencies
If available, link to developer documentation/manual	https://wiki.eclipse.org/ICE
Support email for questions	ice-dev@eclipse.org

Software metadata

Current code version	2.2.1
Permanent link to executables of this version	https://www.eclipse.org/downloads/download.php?file=/ice/builds/2.2.1/
Legal software license	Eclipse public license 1.0
Computing platforms/operating systems	Windows (32/64-bit), Mac OS/X, Linux (32/64-bit)
Installation requirements and dependencies	Java 1.8 or greater
If available, link to user manual. If formally published, include a reference to the publication in the reference list	https://wiki.eclipse.org/ICE
Support email for questions	ice-users@eclipse.org

* Corresponding author at: Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37830, USA.

E-mail address: billingsjj@ornl.gov (J.J. Billings).

@jayjaybillings (J.J. Billings).

Notice of copyright

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

1. Motivation and significance

In previous work, Billings et al. interviewed modeling and simulation subject matter experts to compile a list of requirements for implementing and using these kinds of applications. In the process, they discovered that many of the difficulties inherent in using high-performance modeling and simulation software fall into five distinct categories [1]. These activities, detailed in Section 1.1, include (1) creating input, (2) executing jobs, (3) analyzing results, (4) managing data, and (5) modifying code. There are many tools that address these problems individually, but the same research found that the excess number and specialization of these tools also contribute to the learning curve.

Previous efforts to address these five issues have resulted in general-purpose scientific workflow tools like Kepler [2] or myopic tools that satisfy only a single set of requirements for a single piece of software or a single platform. These are opposite extremes, but a middle-of-the-road solution is also possible. A workflow engine could be developed that limits its scope to high-performance computing (HPC) and to the set of possible workflows associated with the five previously mentioned activities. With only minor additional development, a rich application programming interface (API) could be exposed so that highly customized solutions could still be made based on this limited workflow engine.

It is not clear which, if any, of these solutions is better than the others, and practical requirements will ultimately dictate the path of a project's progress. This work considers a middle ground solution and presents the Eclipse Integrated Computational Environment (ICE) as proof that it is possible to create such a system. Specifically, the work described here shows that

- modeling and simulation activities can be described in a succinct workflow model (see “Workflow Model”);
- an architecture for such a workflow system can satisfy the model of workflows in an extensible way (see “Software Architecture”); and
- such a system is applicable to a suite of problems in energy science, including virtual battery simulations and additive manufacturing, among others (see “Illustrative Examples”).

This section concludes with an introduction to the ICE workflow model. Section 2 details the software from an architectural perspective, and Section 3 provides a set of comprehensive examples. A presentation of the impact is included in Section 4, and details on obtaining sample code are provided in Section 5.

1.1. Workflow model

ICE's workflow model is based on making it easier for scientists to create input, launch jobs, analyze results, manage data, and modify code. Many scientists would most likely find these activities difficult for all codes with which they lack experience, whereas with their own codes – or those with which they are most familiar – these tasks may be so simple that they are taken for granted.

Any particular combination of these activities across one or more scientific software package or code results in a unique workflow. Such a workflow is normally, but not always, requested by a human user and orchestrated by a workflow management system.

The most obvious workflow for any individual simulation code or collection of codes is to string the activities together, where the user's workflow is to create the input, launch the job, perform some analysis, and manage the data—possibly modifying the code in the process. However, there are many other combinations, including re-running jobs with conditions or modifications or analyzing someone else's data.¹

Creating input is the process of describing the physical model or state of a system that will be simulated. This could include creating an input file(s) or making calls to an external process to configure a running program. In most situations, a computational scientist will modify existing input or create new input from a template. “Input” generally includes run time parameters for the simulation framework (e.g., tolerances); configuration options (e.g., data locations, output locations, module configurations); properties of the materials to be simulated; and a discretization of the simulation space (e.g., mesh, grid, particle distribution). The collection of all required input can be quite large and may go by many names, including “input set”, “input package”, “problem”, or, simply, “input”. Often, the set of input files will be described in a “main” input file that acts as a kind of manifest to describe – and provide links to – all necessary information for a given problem.

In this work, it should be assumed – unless otherwise noted – that “input” refers to the entire set of input, not to a single file.

Executing jobs, or “running the workflow” in this context, is the process of performing calculations using a simulation code or framework based on known variables from the input. These are typically run locally for small jobs or for development. Large simulations, on the other hand, typically require a large amount of hardware resources. These resources are usually off-site (i.e., physically unavailable to the user) and are accessed remotely through Secure Shell (SSH) connections or similar protocols. Remote execution requires moving the input in advance of the execution and copying or moving the output to the user's machine. In many cases, though, the output is too large to move to the user's local machine.

Local and remote jobs are often monitored to ascertain a job's status. This monitoring could be a simple check as to whether or not the execution has completed, or it could involve monitoring the output of individual quantities to examine the calculation state. The latter is often used to detect calculation errors that will result in incorrect results. If such problems are found, the job is typically canceled (“killed”) to save compute resources and is then re-run later.

Local jobs in ICE are executed using standard Java system calls. Remote jobs are launched only through SSH connections on remote machines. This includes direct SSH command execution on clusters and proxy connections through a pilot service on large Leadership-class supercomputers. Services such as Globus GRAM and Bosco are not supported, but the SSH command execution includes extensive support for numerous batch and queuing systems. The Eclipse Parallel Tools Platform (PTP) is used to create all remote SSH connections, regardless of the target machine [3].

In this work, it should be assumed – unless otherwise noted – that “executing a job” includes monitoring that job in one or more ways, possibly including real-time updates to visualizations. It is also important to note that executing a job is not the same as executing a workflow. Executing jobs specifically refers to launching simulations, whereas executing a workflow could be something different such as generating input or post-processing results.

¹ The authors have identified many unique combinations that define workflow “classes”. When possible, every effort is made to give the classes colloquial names such as “The Re-Run” or “The Graduate Student”.

Download English Version:

<https://daneshyari.com/en/article/6964938>

Download Persian Version:

<https://daneshyari.com/article/6964938>

[Daneshyari.com](https://daneshyari.com)