7-2004

Plone and Content Management

George K. Thiruvathukal
Loyola University Chicago, gkt@cs.luc.edu

Konstantin Läufer
Loyola University Chicago, klaeufer@gmail.com

Recommended Citation

George K. Thiruvathukal, Konstantin L?ufer, "Plone and Content Management," Computing in Science and Engineering, vol. 6, no. 4, pp. 88-95, July/Aug. 2004, doi:10.1109/MCSE.2004.19

Editors: Paul F. Dubois, paul@pfdubois.com
George K. Thiruvathukal, gkt@cs.luc.edu

PLONE AND CONTENT MANAGEMENT

By George K. Thiruvathukal and Konstantin Läufer

WITH THE ADVENT OF THE WEB, TIM BERNERS-LEE AND HIS TEAM AT CERN CREATED A FRAMEWORK THAT SPARKED A REVOLUTION. THIS REVOLUTION AND THE DOT-COM era that followed it are well known, so we won't rehash them here, but we do want to reflect on why Berners-Lee and his team created the Web in the first place. The idea was to look beyond FTP and establish a more sophisticated protocol for disseminating scientific information. Although nobody would argue that HTTP and HTML are anything but groundbreaking, current search technology makes it abundantly clear that we need better ways of organizing Web content.

Perhaps nowhere other than in academic and scientific circles is the need for better approaches to managing individual, community, and research content more apparent. Most Web sites are maintained by authors who cobble together HTML pages, many of which bear that evil "under construction" emblem or aren't kept up to date. Worse, most maintainers don't have time to invest in learning HTML properly, despite the wealth of authoring tools that aim to simplify the task of making consistent and attractive Web pages.

Content management systems (CMSs) have been the subject of commercial and open-source community interest for some time. A CMS is the only way to support distributed content development, let alone facilitate content maintenance. CMSs take off where most Web authoring tools give up—for example, as good as Dreamweaver is as an HTML authoring tool, it's inadequate for maintaining even small portal sites managed by a few developers. The model of publishing remains FTP (not exactly known for its security), which offers no direct support for version control.

In this column, we're going to look at Plone (www.plone.org), which we feel is one of the best CMSs available today. Even better, it's distributed under a free open-source license: the cost of getting started is only limited to the time you have available to set up the software on a server. Plone is written in Python and uses the Zope application server infrastructure; it runs on most modern operating systems. We've even set it up at Loyola University Chicago in the Department of Computer Science. Besides being two faculty members who rely on Plone to host all our Web content, we have recently customized Plone for our department's public Web site (www.cs.luc.edu). The end result is a site on which content can be maintained entirely over the Web at zero cost to our department. Everything you see on our Web pages was generated by a unique combination of plaintext authoring, clever plug-ins, and a site-wide style.

Plone

Plone lets you organize your public (Web) life: by using it, students, colleagues, and outside collaborators and researchers can find your papers and teaching materials. However, the word *plone* doesn't signify anything. We even checked the dictionary to see if it was one of those words we forgot to study for the SATs or GREs. All indications are that it's not a word—at least, not in English—so is it a noun? A verb? Some other part of speech? In our own work, we tend to use it like this:

- Have you ploned today? Did you plone home?
- Your page appears to be a plone of my page.
- If you were prehistoric, you'd probably be called a plonosaurus.

Before we get into trouble with more bad puns on words that aren't even words yet, let's dive into Plone and explore its vocabulary and salient features.

Typed Content

A key aspect of Plone is support for the notion of *typed* default content. With XML, we see this notion in the form of document type definitions (DTDs). The same idea applies to Plone, just in a more user-friendly manner: Plone users can do almost all their work without thinking about markup, but they can still use XML to customize content appearance. Other CMSs support similar principles, but in Plone, the notion of typed content is similar to what we see in operating systems, with the added twist of rich metadata that is content-type-specific.

 Copublished by the IEEE CS and the AIP 1521-9615/04/$20.00 © 2004 IEEE

Case Study: A Multisite Server for an Academic Department

Let's look at the Plone-based Web server we set up in the Department of Computer Science at Loyola University Chicago.

Requirements

The requirements for this setup were an inexpensive commodity hardware and operating system; multiple sites on one server, including public content, departmental intranet, individual faculty members' sites, and project sites; customization of the visual design of each site to match the PR department's requirements for public sites and faculty members' preferences for individual and project sites; and integration with static legacy content, including a public home page with a different layout than the rest of the site as well as faculty members' existing static pages and scripts.

Solution

We came up with a flexible yet easy-to-manage solution that leverages the Zope and Plone technologies described in the main article. For under US$2,500, we put together a dedicated single-processor server with a five-disk SATA RAID with one terabyte of usable capacity. This server runs Gentoo Linux, which offers excellent optimization capabilities and configuration management. We are currently running a single instance of Zope 2.6.1 with Plone 1.0.1; an upgrade to Plone 2.0 is planned. Each of the various sites mentioned is realized as a separate Plone site living within the Zope instance. This allows each site to be customized separately with respect to visual design, logos, and so on. The resulting site URLs are www.cs.luc.edu:8080/site1, www.cs.luc.edu:8080/site2, and so forth.

We're running Apache 2.0.47 as a front end. This lets us integrate the various kinds of content as required. Using Apache's name-based virtual hosts and the ModProxy module in conjunction with a Zope "virtual host monster," we can transparently assign one or more different host names to each site, making the sites look like separate servers. The virtual hosts can be found at site1.cs.luc.edu, site2.cs.luc.edu, and so forth, mapping to the site URLs given earlier.

In addition, the Apache ModProxy and ModAlias modules let us have a static (X)HTML home page whose layout is as required by our PR department; we can transparently expose the original faculty home pages that still reside on our old server. Simply specifying a custom logo and some custom cascading style sheets customizes most of the Plone sites, as does changing the layout of the various page components into columns. This type of customization requires no programming and, optionally, only limited knowledge of (X)HTML.

Our public site has to satisfy the PR department's requirements, including showing a different banner image for each section of the site. We were able to address this requirement by writing a small Python script that we dropped into the site's styles module.

Here are Plone's standard content types:

- *Folders* let you collect other content, including nested folders.
- *Documents* can be written in plaintext, HTML, or a variant of plaintext known as structured text. Documents are rendered according to cascading style sheets (CSSs).
- *Links* let users create a symbolic name for any external URL to be referenced—for example, cise can be created as an internal name for whenever we want to reach www.computer.org/cise.
- *Files* allow a symbolic name to be associated with documents not written in plaintext or markup that need to be incorporated into portal sites.
- *Events* are content with a date, typically associated with a calendar. Whenever an event is created in Plone, it can be published on a calendar.
- *Images* are graphical content that can be incorporated in documents simply by referencing the image.
- *News items* are similar to documents but are intended for informational sites on which news events come and go. Whenever news items are published, they'll appear as news (provided the news still is news).

Plone is not limited to the content types described here—for example, plugins are tremendously powerful and a key ingredient to Plone's success—but we won't cover such concepts here. Suffice it to say that plugins are to Plone what the extensions framework is to Python: new components can be introduced to support new content, in some cases without even having to restart the Zope server.

Content Properties

In addition to managing various typed content, each content item can have one or more associated properties. Properties let us go beyond the typical Web site configuration and introduce dynamism. Here are some of Plone's properties:

- *Allow discussion* permits a given content item to allow Web-based threaded discussion. This is usually relevant to document content, but it can be used for any type of content in Plone.
- *Keywords* let users enter keywords facilitating search for a particular item. Contrasted with HTML, they allow metadata to be associated with any content, including content not written in HTML.

Cafe Dubois

"How much pain have cost us the evils which have never happened?"

–Thomas Jefferson, letter to John Adams, 8 April 1816

Take Two Aspirin

I have a coffee cup I bought in the souvenir shop at Monticello, and it has the saying above from Jefferson that I have personally found very useful to remember. (An interesting Web site about what Jefferson did *not* say is "Spurious Jefferson Quotes" at www.geocities.com/CapitolHill/7970/jefpco13.htm.) In my conversations with groups that do scientific programming, I often find people concerned with evils that are never going to happen. Indeed, it seems it's human nature to always be worrying about the wrong thing. People try to make the software process prevent evil when all it can really do is prevent accidents and aid communication.

If someone on your project gives in to fascist impulses and wants to impose a set of Draconian procedures that will allegedly prevent mistakes, remind him or her of this difference between evil and accident. We should never mistake incompetence for bad intentions. Making it easy to do the right thing and hard to do the wrong thing is about all we can usefully do. Actually, the evil part often happens when trying to cover up incompetence.

For example, a friend of mine—I'll call him Rob—was devastated when his wife of more than 38 years ran away with his best friend. Rob, who is a gentle and generous soul, went into a depression. Rob is diabetic and, in his depressed state, did not eat regularly; his foot started hurting as it sometimes does if his diet gets out of whack.

After a week, Rob decided that he just wasn't doing very well and that he should try to get some help. So he went to the local emergency room and was ushered into what they call on *ER* a "Psych Consult."

After a few preliminary questions, the shrink asked Rob, "Are you having any pain?" Rob replied, "Well, yes, I have a stabbing pain in my foot."

Wrong answer. The doctor pressed the secret button and in came some burly orderlies who took Rob by force to the maximum-security ward where they put him on suicide watch for three days. The doctor wrote in his chart that Rob had stabbed himself in the foot.

Later, someone wrote in the chart that the "wound" was clean and did not require attention.

Programming News

Previously we've talked about VPython (vpython.org), a Python add-on designed to help physics students do visualizations without learning a lot of programming. VPython now supports 3D: add one line to your program, put on your 3D glasses, and you're in business.

Bruce Eckel, author of *Thinking in C++*, *Thinking in Java*, and so on, gave a talk at PyCon 2004 and mentioned a new language, D. You can read about D at www.digitalmars.com/d/. I thought it looked interesting.

Recently, I needed to rid one of our home machines of all the adware and other offensive stuff that my kids had managed to acquire, so I went looking for SpyBot—Search and Destroy. This is a donation-supported program by Patrick M. Kolla that is easy to install and use, and, if you have a PayPal account, it's easy to send in your donation. But I did encounter something alarming in the process: evildoers are gaming Google, trying to misdirect your search for this program into commercial alternatives or worse. So here's the right URL: www.safer-networking.org.

- *Effective date* is when the published content item becomes available; it's particularly useful for news and community-style portals.
- *Expiration date* is when the published content item ceases to become available.
- *Format* is MIME encoding for a content item. When content is created in Plone, the correct MIME encoding is usually deduced, but Plone lets you set MIME type explicitly whenever it's incorrectly deduced. (This solves an important problem on many Web sites where the server incorrectly deduces the MIME type and causes browsers to display content as ASCII/Unicode text.)
- *Others* include language, copyright, and contributors, all of which should be self-explanatory.

All these properties are useful, but some go well beyond

what we find in static HTML and Web content management. Contrasted to sites where content is maintained manually and permission bits set content visibility (often incorrectly), Plone makes the entire process as simple as filling in dates.

Publication States

A notable advantage of Plone is that it goes beyond the traditional notion of Web sites where all content lives in a file system. It distinguishes between three states of existence for all types of content:

- *Visible* content can be accessed if a user knows the URI to reach it. To be visible, each item in the URI's path must be visible or published.
- *Published* content includes the definition of visible content plus the ability to actually see a link to the content from within another content item. The distinction between visible and published content is not obvious without a few clarifying examples, which we'll discuss shortly.
- *Private* content can't be accessed by anyone except users in the portal with sufficient privileges. Plone's security model—as well as that of most CMSs—is deliberately simple compared to standard Unix- or Windows-style security.

In Plone, the notion of visible and published content seems largely academic until the notion of content management becomes more familiar. Consider an event content item: if an event is created and placed in the visible state, visitors to the portal who are not content managers will not see the item on the calendar. The calendar itself is a plugin that aggregates event content and highlights dates, but for events to show up on the calendar, we must publish the event formally, especially if it's to be visible to general site visitors.

The distinction becomes apparent when using Plone to make a personal or community site. Site visitors will not see visible items at first glance, but those who maintain the portal will see all items (private, visible, or published).

The notion of states allows for truly private content to remain private. However, we stress that this model is not the equivalent of user/group security (as with Unix) or access control lists (as with Windows). It is designed to distinguish only between maintainers and nonmaintainers, and results in a useful simplification that lets maintainers focus on real content and not the innards of Unix security that plague so many maintainers of static HTML sites.

Authoring Content with Structured Text

We usually create content in Plone by creating document content items, which we can write in plaintext, HTML, or structured text. To create a page, we go to a folder in the Contents View and use a drop-down list to select the type of item (document) to be added. A form is then displayed to specify the document's attributes (name, title, and content).

Structured text was conceived as a simplification to HTML and was part of another common collaboration tool known as a Wiki. Ward Cunningham (one of the fathers of so-called object-oriented analysis) created the Wiki concept, which is designed to support straightforward content authoring. This idea is not altogether new: LaTeX and HTML both aimed to provide a straightforward markup that was more or less plaintext in nature. However, structured text abandons the idea of formal markup altogether, instead focusing on clever uses of indentation and regular sequences of characters to provide special textual emphasis, such as italics. We can't go into extreme detail on this here, so let's look at an example. Figure 1 shows an example of structured text source, illustrating some common text elements. Figure 2 shows how a browser, depending on the exact CSS styles applied, might display this text.

We've found we can use structured text for just about anything that we want to make into a page within the CMS. When utter sophistication is required (a rare but sometimes necessary evil), or when we have a lot of legacy content, we can always upload regular HTML. When doing so, we recommend using vanilla (X)HTML, thereby allowing the CSS to make your content look consistent. For highly collaborative sites that have multiple authors, structured text is a clear winner. It's easy to read and write, and it lets authors who couldn't care less about HTML and all its idiosyncrasies focus on actually writing.

Customization and Personalization

The idea of using a CMS to maintain a portal sounds great, but many Plone and CMS users get discouraged once they see the default look (skin). It makes them reminisce about the days of HTML when they could do whatever they wanted, simply by clever coding.

Plone allows for virtually unlimited customization—completely separate from content authoring—through several mechanisms.

CSS

CSSs are to Web publishing what paragraph styles are to Word files or document classes are to LaTeX. Using a CSS lets you determine how something in HTML is formatted without having to apply explicit formatting. Most CMSs rely on a CSS to format rendered content, simply by linking the content to the CSS (using HTML syntax for CSS linking).

```
  My Heading

    This text is part of a paragraph. This text belongs to the same paragraph. So does
this text. We know that My Heading is a heading, because any paragraph indented with
whitespace denotes a new level, resulting in an actual heading being generated when
Plone transforms the structured text to HTML.

    To make another paragraph, a blank line is left between paragraphs, similar to what
is done in LaTeX.

    Unlike LaTeX, however, I do not need to learn markup rules, which sometimes trip up
even the most seasoned users. Of course, LaTeX is still the greatest system ever, but
it might not be the best choice for authoring simple Web content.

  Another Heading

    Making a link to "the CISE site":http://computer.org/cise is straightforward. When
Plone processes the structured text, the text in quotes will become the link label. The
text encountered until whitespace is seen will be the hypertext reference.

  Other things, like descriptions, are straightforward:

  Hydrogen -- The first element.

  Helium -- The second element.

  Lithium -- The third element.

  Basically, structured text is designed for writing up pages quickly by having an ex-
tremely light syntax. Here is how we write *bold* and _underlined_ text. You can even
do 'typewriter' text, similar to LaTeX, by surrounding the text with single quotes.

  Of course, structured text may not be for everyone. So you can always write HTML and
upload it to your Plone site. If you avoid explicit styling and formatting, the Plone
system will automatically style the text for you with attractive results.
```

Figure 1. Plone example. Structured text source of some sample text. Figure 2 shows it via a browser.

Plone goes one step further by providing Web-based forms for specifying the CSS, thus allowing users to specify how each HTML tag will be formatted without requiring them to learn anything about the actual details. For the record, CSS is a complex beast compared to HTML, and most Web authors don't even bother trying to figure out how it works. You can actually upload a CSS created with a tool such as Dreamweaver to a Plone site to create the ultimate personalized appearance.

XML

Plone differs from static authoring tools such as Dreamweaver in its use of XML: Plone uses it to support the notion of page templates, which is best thought of as a mix of HTML and XML. One of XML's advantages is that we can use it do many of the tasks found in higher-level scripting languages such as Python. For example, the Loyola Computer Science Department Web site needed a graphical banner for each major department category. Based on the top-level folder's name, the correct graphical banner is displayed. We were able to support such a feature thanks to Plone's powerful page template (and XML) functionality. The same task, even in professional tools such as Dreamweaver, is cumbersome at best, often requiring separate templates for each different header to be displayed.

Python

Plone is written in Python, so one of the world's greatest programming languages is at your fingertips for custom func-

My Heading

This text is part of a paragraph. This text belongs to the same paragraph. So does this text. We know that My Heading is a heading, because any paragraph indented with whitespace denotes a new level, resulting in an actual heading being generated when Plone transforms the structured text to HTML.

To make another paragraph, a blank line is left between paragraphs, similar to what is done in LaTeX.

Unlike LaTeX, however, I do not need to learn markup rules, which sometimes trip up even the most seasoned users. Of course, LaTeX is still the greatest system ever, but it might not be the best choice for authoring simple Web content.

Another Heading

Making a link to the CISE site is straightforward. When Plone processes the structured text, the text in quotes will become the link label. The text encountered until whitespace is seen will be the hypertext reference.

Other things, like descriptions, are straightforward:

Hydrogen
: The first element.

Helium
: The second element.

Lithium
: The third element.

Basically, structured text is designed for writing up pages quickly by having an extremely light syntax. Here is how we write *bold* and underlined text. You can even do `typewriter` text, similar to LaTeX, by surrounding the text with single quotes.

Of course, structured text may not be for everyone. So you can always write HTML and upload it to your Plone site. If you avoid explicit styling and formatting, the Plone system will automatically style the text for you with attractive results.

Figure 2. Example text. This is how the example text in Figure 1 would look like in a browser window.

tionality. We can use Python on two levels: to develop a server-side plugin for new content and to provide additional functions within the XML-provided page-template mechanism. The example we described earlier (with the changing banners) required an auxiliary Python function to get the top-level folder name of an arbitrary folder path to select the banner to be displayed. For example, if the currently selected path is /admission/graduate, the admission.gif banner is to be displayed. The same is true for when the path is /admission/undergraduate. To support this capability, we added a function to map the path to an image. `getPath('/admission/undergraduate')` returns `'admission'`.

It's possible to maintain all your content without a CMS and achieve total personalization while having some of Plone's benefits—for example, you could use PHP Hypertext Preprocessor (PHP) to achieve much of the dynamic behavior found in Plone. However, Plone's tremendous power is achieved by allowing all the content to be maintained from within the Plone software itself. You're never put in the position of having to copy files to the remote server, then ensuring all permissions bits are set correctly, and then ensuring consistent appearance and navigation.

Hosting Plone

A major question that invariably arises is this: How do I get started? We can answer this in one of two ways, depending on whether you just want to experiment with Plone or if you actually want to host serious content in a production environment. We're going to assume the latter.

Hardware

We recommend addressing hardware first. For our department, we decided to go for broke and build a serious box for hosting our Web environment. We weren't just looking to

Other Types of Content Management Systems

Let's review some of the other types of content management systems (CMSs) and related systems, with an emphasis on open-source systems.

Slash and Slashclones

Slash (the software used by Slashdot), PHP-Nuke, and their numerous derivatives (sometimes referred to as Slashclones) are intended as community news portals and weblogs. They usually come with a variety of modules that provide functionality such as discussions, calendars, and polls, making it easy to set up a community site from scratch. However, they often support only a fixed set of content types and layout options, and any major customization requires hacking around a sometimes-unwieldy code base.

Wikis

The original WikiWikiWeb and its numerous clones, which now exist in almost any imaginable technology, are Web server applications that let users create and edit collections of Web pages through a browser.

Actually, the structured text markup language used by Plone is based on the kinds of simplified markup languages used to author Wiki pages. Wikis can be a very effective knowledge-capturing tool, but they vary considerably in terms of the features they provide, such as markup capabilities, customization, versioning, searching, permissions, and so forth.

Zope and CMF

Technically, Zope is an application server written in Python, and CMF (Content Management Framework) is a Zope product, a plugin that runs on top of a Zope installation. CMF supports different content types, such as articles, news items, links, and documents, and lets programmers define new content types. The Plone CMS, in turn, is a Zope product that leverages the Zope CMF.

Various other Zope products are available and can provide much of the functionality that comes out of the box with a typical Slashclone. There is even a Wiki clone and a Slashclone available as Zope products, allowing Wiki and community news content to coexist with Plone sites.

Java-Based Systems

In case you're looking for a Java-based CMS, there are several reasonable open-source choices. OpenCMS is full-featured but depends too heavily on MS Internet Explorer for WYSIWYG editing. Cofax (Content Object Factory) is geared toward news content management and aggregation. There is even a Slashclone in the Java world, JBoss Nukes, which was launched by the JBoss group as a more scalable, robust alternative to the Perl and PHP-based versions.

Other Open-Source and Commercial Systems

A plethora of open-source and commercial CMSs are available now, varying greatly in terms of features, supported platforms, technology, and, for commercial systems, price. In some cases, these systems support specific aspects of content management required in certain industries—for example, role-based workflow.

Where to Look for More Information

thocar.free.fr/ChoosingACmsKDBSoftware.html: A detailed comparison of several open-source CMSs based on a list of required features and feedback from advanced users and developers. This site helped us choose Plone for the content management needs of our department.

www.quakernet.org.uk/sites/ituse/CMSs.htm: A white paper on CMSs in the context of the Web strategy for a nonprofit organization. This paper provides background and compares various open-source and commercial systems.

www.commonsgroup.com/ideas/fulltext.shtml?x=212: A systematic high-level comparison of several CMSs including Plone. This paper also includes a detailed feature chart.

www.cmsinfo.org: A community of CMS users and developers. It focuses mainly on news and information on open-source CMSs.

www.oscom.org: The official site of the International Association for Open Source Content Management. The site includes a matrix of open-source CMS products.

www.cmsmatrix.org: Another CMS matrix that includes useful statistics. Incidentally, Plone is the frontrunner in most categories.

www.contentmanager.eu.com: A site for users and providers of CMSs, which includes background information and resources on choosing a CMS.

www.la-grange.net/cms: A list of open-source CMS products categorized by technology.

www.clueful.com.au/cmsdirectory: A metadirectory of sites related to Web content management.

www.opensourcecms.com: This site lets you try out some CMSs online without installing them yourself. However, it specializes in systems with a PHP front end and a MySQL database back end.

host the department's Web site but to provide an infrastructure that faculty, students, research groups, and collaborators could use to exchange ideas. The machine we built provides one terabyte of RAID-5 storage (with hot-swappable

drives). We took advantage of a RAID controller from a company called 3ware (www.3ware.com), which provides a line of controllers for building scalable storage solutions from commodity ATA or Serial ATA hard drives. We chose 3ware for its excellent Linux support.

Linux

For us, the only rational choice is Linux. Of course, Plone runs on all platforms that support Python, including Windows and Macintosh OS X. We chose Gentoo Linux (www.gentoolinux.org), which is rapidly evolving as a Linux distribution of choice. The ongoing management of the environment is achieved by updating local metadata to learn about new packages. Gentoo Linux differs from the major Linux distributions (such as Red Hat) by optionally building everything from source code. What's great about Gentoo is the availability of most major open-source packages, including Plone. Once your Gentoo system is up and running, all you need to do is make a few configuration changes, and you're ready to begin Plone hosting!

Apache

The Apache Web server plays a key role in properly hosting Plone. You'll want to run Apache, especially if you want to take advantage of Secure Sockets Layer (SSL) and virtual hosting. Virtual hosting plays a major role in our environment—for example, www.thiruvathukal.com and www.cs.luc.edu are hosted on the same system. The Plone software doesn't provide these useful capabilities, even though it does feature an embedded Web server to serve all content that it manages.

Content management is a powerful idea, but it's also easy to see why many people don't even bother maintaining their Web pages. The notion of static HTML authoring doesn't scale well and requires a tremendous amount of expertise to execute even minimally. Plone is one of the best-of-breed solutions, and we think you'll agree, but in the meantime, we're going to get back to some of our human ploning experiments.

George K. Thiruvathukal is an associate professor of computer science at Loyola University Chicago. He also is president and CEO of Nimkathana Corporation, which does research and development in high-performance cluster computing, data mining, handheld/embedded software, and distributed systems. He wrote two books with Prentice Hall covering concurrent, parallel, distributed programming patterns and techniques in Java and Web programming in Python. Contact him through www.cs.luc.edu/gkt.

Konstantin Läufer is an associate professor of computer science at Loyola University Chicago. He is also director of architecture and application services at Nimkathana Corporation. His research interests include programming languages, software architecture and frameworks, concurrent and distributed systems, and mobile computing. He received a PhD in computer science from the Courant Institute at New York University. Contact him through www.cs.luc.edu/laufer.

How to Reach *CiSE*

Writers

For detailed information on submitting articles, write to cise@computer.org or visit www.computer.org/cise/author.htm.

Letters to the Editors

Send letters to Jenny Ferrero, Contact Editor, jferrero@computer.org. Please provide an email address or daytime phone number with your letter.

On the Web

Access www.computer.org/cise/ or http://cise.aip.org for information about CiSE.

Subscription Change of Address (IEEE/CS)

Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify CiSE.

Subscription Change of Address (AIP)

Send general subscription and refund inquiries to subs@aip.org.

Subscribe

Visit https://www.aip.org/forms/journal_catalog/order_form_fs.html or www.computer.org/subscribe/.

Missing or Damaged Copies

If you are missing an issue or you received a damaged copy (IEEE/CS), contact membership@computer.org. For AIP subscribers, contact kgentili@aip.org.

Reprints of Articles

For price information or to order reprints, send email to cise@computer.org or fax +1 714 821 4010.

Reprint Permission

To obtain permission to reprint an article, contact William Hagen, IEEE Copyrights and Trademarks Manager, at copyrights@ieee.org.