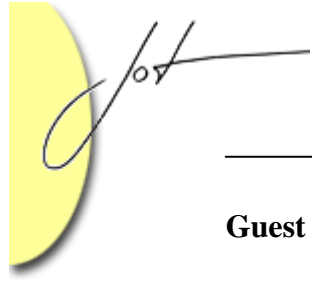


Contents

	Page
Editorial	5
<hr/> COLUMNS <hr/>	
Strategic Software Engineering	
Intentionality <i>By John McGregor</i>	7
<p>With the latest statistics claiming that 15- 50% of projects are cancelled before they deliver anything, perhaps what we have always gotten is not what we have always wanted. Intentionality is a property of a decision. A decision that is the result of explicit examination of relevant factors is an intentional decision. It takes longer to make an intentional decision so many managers don't make the effort. They go with the default, business as usual, and they get what they have always gotten.</p>	
Java at Large	
The Discrete Fourier Transform, Part 5: Spectrogram <i>By Douglas Lyon</i>	15
<p>There are several methods available for improving our ability to identify guitar notes. For example, each note has a unique harmonic signature. Presently, we only take the strongest harmonic. However, if we look at several harmonics, we could take advantage of individual note harmonic signatures.</p>	
Business Objects	
The Practice of "Architecting" Cloud Solutions <i>By Mahesh Dodani</i>	25
<p>Creating a cloud service and publishing it in the service catalog involves both the cloud service developer as well as the service provider. The service developer creates service templates that will be offered to the service consumers. The type of service templates varies according to type of cloud offering. Examples of service templates include a collection of simple or composite images, storage volumes, virtual desktops, and web-conferences. Service templates are typically published in a service catalog.</p>	



Guest column

- Covariantly Adjusting Co-Types in Timor** 35
By Leslie Keedy, Gisela Menger, and Christian Heinlein

We introduce a safe form of *automatic* covariant adjustment of the parameters of co-types, to reflect the different expanded types which occur in the subtype hierarchy of the initial expanded type. This potentially spares the programmer from writing such methods explicitly and helps to ensure consistency between co-types which are related via an adjustment hierarchy.

Guest column

- A Modern, Compact Implementation of the Parameterized Factory Design Pattern** 57
By Harold Fortuin

A modern OO language update of the Parameterized Factory Creational Pattern is presented. It is compact since it folds the Factory method into an abstract base class, and imposes a package structure on its concrete subclasses. The design is demonstrated in Java, but is also applicable to C# and other modern object-oriented languages.

Educator's Corner

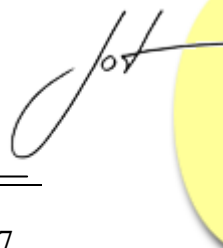
- Darwin's World Simulation in C#: An Interpreter** 65
By Richard Wiener

In Darwin's world the user creates robot-like graphical creatures with behavior defined by a simple programming language. These creatures migrate around a small two-dimensional grid, each according to its simple program created by the user. The GUI application that controls and displays the location of these creatures must interpret the program instructions supplied in a simple text file for each creature type in real-time and display the movement and behavior of these creatures.

REFEREED ARTICLES

- A Framework for Adding Design by Contract™ to the .NET Object-Oriented Programming Languages** 81
By Jennifer Pandolfo and Cui Zhang

Even though Design by Contract (DBC) has been supported by Eiffel since 1985, other programming languages that offer built-in DBC support are still rare. Redundant efforts have taken place to implement the support of DBC for different object-oriented programming languages. This paper presents the design and implementation of a framework for extending object-oriented programming languages to support DBC.



How AspectJ is Used: An Analysis of Eleven AspectJ Programs 117
By Sven Apel

To see how programmers use AspectJ, metrics are presented that distinguish the use of aspects in terms of the classifications discussed in the last section. For each metric, we count the number of lines of code (LOC) for different categories and determine the fraction of the program's source for each category.

A Program Transformation Technique to Support AOP within C++ Template 143
By Suman Roychoudhury, Jeff Gray, Jing Zhang, Purushotham Bangalore, and Anthony Skjellum

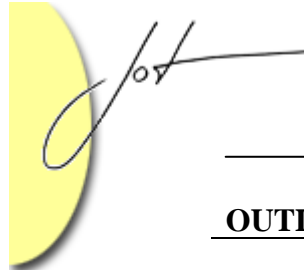
The combination of aspects and generics is a topic in need of further investigation. This paper enumerates the general challenges of uniting aspects with C++ templates. It also underlines the need for new language constructs to extend AOP support to C++ templates and provides an initial solution to realize this goal.

An Aspect-Oriented Approach for the Development of Complex Simulation Software 161
By Tudor B. Ionescu, Andreas Piater, Walter Scheuermann, and Eckart Laurien

Aspect-oriented programming is used to implement cross-cutting concerns like distribution, workflow engine integration, persistence, or fault tolerance whereas standard object-oriented programming is used for implementing the core functionality of the simulation application. We provide a proof of concept for this approach by describing the implementing two concerns specific to simulation software, namely distribution and workflow engine integration.

On Differencing Object-Oriented Formal Specifications 183
By Fathi Taibi, Md. Jahangir Alam, and Junaidi Abdullah

A differencing approach for OO formal specifications is proposed in this paper. It comprises two parts: the first part consists of comparing specifications to identify their matching elements. The second part uses the matching results to produce deltas differentiating them. These deltas are formally modeled as a set of primitive operation with traceability information allowing the reversal of their effect. The differencing approach is empirically validated.



OUTLOOK

A brief outlook to the next issue

199