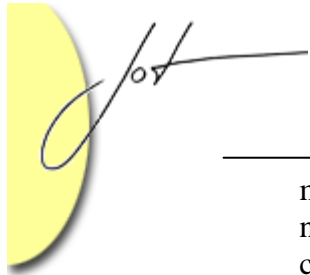


## Contents

	Page
<b>Editorial</b>	5
<hr/>	
<b>COLUMNS</b>	
<hr/>	
<b>Strategic Software Engineering</b>	
<b>Strategic Software Engineering</b> <i>By John McGregor</i>	7
<p>Typically a company seeks to achieve a unique position that will give them a competitive advantage. In software development we see many ways to achieve advantage. Apache servers are successful because they are high quality, free and their source is available for modification. Eclipse is successful because it is high quality, free, and its architecture makes it sufficiently flexible for a wide variety of uses. The Cummins Engine product line and the GM Power Train product line are successful because they use a product line strategy that facilitates planned reuse of software thereby reducing expenses and increasing quality and productivity.</p>	
<b>Java at Large</b>	
<b>The Discrete Fourier Transform, Part 1</b> <i>By Douglas Lyon</i>	17
<p>The DFT is typically held as too slow for direct computation. However, for small windows of time, on even modest machines and voice-grade single-channel 8-bit audio, we find that the computation can be fast enough for real-time processing.</p>	
<b>Business Objects</b>	
<b>Get Your Head In The Clouds!</b> <i>By Mahesh Dodani</i>	27
<p>Cloud computing provides the enterprise the capabilities to lower the cost of delivering IT services optimized through flexible delivery</p>	



models, while at the same time making IT more responsive to business needs and allowing greater visibility of IT usage to support billing and chargeback. Cloud computing can be a key catalyst to transform the enterprise to be able to innovate to gain competitive advantage in a fast changing environment by providing reliable IT services that can quickly and flexibly adapt to meet business requirements.

## The OOP Scene

### **Cloud Computing – Benefits and Challenges!** 37

*By Dave Thomas*

Recently I've been somewhat surprised by the lack of understanding of the impact of Cloud Computing. Cloud Computing enables Next Generation IT and provides the long promised alternative to mainframes and surrounding tiers. The recent rushed announcement of the Open Cloud Manifesto confirms that even those late to the party see it as a major sea change in the industry.

## Guest Column

### **Bad smells in design and design patterns** 43

*By Cédric Bouhours, Hervé Leblanc, and Christian Percebois.*

We have wittingly chosen to represent a design pattern as a class diagram (inspired from the structure section of the GOF catalog), only with pattern participants as class name and inter-classes relations (association and inheritance links). Indeed, since we are at the design level, we are interested in the architecture of the pattern (classes and relations between them).

---

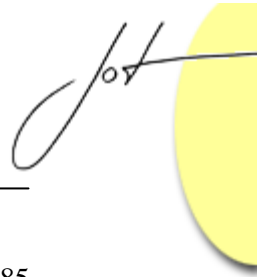
## REFEREED ARTICLES

---

### **A Novel Approach to Generating Test Cases from UML Activity Diagrams** 65

*By Debasish Kundu and Debasis Samanta*

In this work, we propose a novel approach for generating test cases using UML 2:0 activity diagrams. In our approach, we consider a coverage criteria called *activity path* coverage criteria. Generated test suite following *activity path* coverage criteria in our approach aims to cover more faults like synchronization fault, fault in a loop than the existing works.



- Towards the Integration of UML- and textual Use Case Modeling** 85  
*By Veit Hoffmann, Horst Lichter, Alexander Nyßen, and Andreas Walter*

According to the UML, a use case represents a variety of scenarios that can result from the interaction between a system and its environment. Its description has to cover all possible scenarios. Since it is hard and often impossible to name and describe each of them in isolation, a use case is often described in an incremental fashion: One scenario is described completely and explicitly, and all other scenarios are described implicitly in terms of their differences to the first one.

- Guidelines for Enabling the Extraction of Aspects from Existing Object-Oriented Code** 101  
*By Marco Tulio Valente, Marcelo Nassau Malta, and Samuel Domingues*

However, one of the principal obstacles for applying aspect-oriented refactoring techniques is the fact that aspect languages { such as AspectJ [12] } only allow aspects to advise well-defined points in the execution of object-oriented systems, called join points.

- Automated State-Based Unit Testing for Aspect-Oriented Programs: A Supporting Framework** 121  
*By Mourad Badri, Linda Badri, and Maxime Bourque-Fortin*

Interactions between aspects and classes are a new source for faults. Existing object-oriented testing techniques are not adequate for testing aspect-oriented programs. As a consequence, new testing techniques must be developed.

---

## OUTLOOK

---

- A brief outlook to the next issue** 127